Integrating Tools to Support Software Measurement

Vinícius Soares Fonseca, Monalessa Perini Barcellos, Ricardo de Almeida Falbo

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

{vsfonseca, monalessa, falbo}@inf.ufes.br

Abstract. *Software measurement is a key practice to process improvement and project management. Given the nature of measurement activities, supporting tools are essential. Different tools can be combined to support the measurement process and provide necessary information for decision making. However, these tools are usually developed by different developers, at different points in time and without concern for integration. As a result, organizations have to deal with integration issues to allow tools communication and properly support the measurement process. In this paper we present a tool integration initiative performed following the Ontology-Based Approach for Measurement Systems Integration (OBA-MSI) aiming to support the measurement process in a software development organization. The integration involved three tools: Taiga, SonarQube and SoMeSPC.*

1. Introduction

Software measurement involves defining measures, collecting data for these measures and analyzing data aiming to support decision making [McGarry *et al.* 2002]. There are several standards (such as ISO/IEC 12207 [ISO/IEC 2008] and maturity models (such as CMMI [SEI 2010]) that address software process improvement and include measurement as an essential process. 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 that process [Maretto and Barcellos 2013].

Typically, organizations adopt different tools to support different processes. For example, schedule and budget tools are used to support project management, and development environments and version control systems are used to support coding and source code management. Despite these tools are not usually conceived to support software measurement, many times they store useful data related to the supported processes (e.g., number of defects, time and cost spent on activities, etc.).

To support properly the software measurement process, tools must be integrated, but this is not an easy task. The heterogeneity between systems is the major difficulty. In general, each tool runs independently and implements its own data and behavioral models, which are not shared between different tools, leading to conflicts [Izza 2009]. Semantic conflicts occur when applications use different meanings to the same information item, i.e., when information items seem to have the same meaning, but they do not. To reduce these conflicts, integration initiatives should address semantic issues. Ontologies can be used as an interlingua to map the concepts used by different applications, enabling data and services understanding [Calhau and Falbo 2010].

Considering that, we developed the *Ontology-Based Approach for Measurement Systems Integration* (OBA-MSI) [Fonseca *et al.* 2016], a systematic approach that uses the Reference Software Measurement Ontology (RSMO) [Barcellos *et al.* 2013] and the Software Measurement Task Ontology (SMTO) [Barcellos and Falbo 2013] to guide tool integration to support the software measurement process. In this paper we report the use of OBA-MSI in a tool integration initiative to support the measurement process in a software development organization. This paper is organized as follows: Section 2 addresses aspects related to software measurement and integration and introduces OBA-MSI; Section 3 describes the use of OBA-MSI in a tool integration initiative; Section 4 covers related works; and Section 5 presents our final considerations.

2. Background

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

For performing software measurement, initially, an organization must plan it. Based on its goals, the organization has to define which entities (processes, products and so on) are to be considered for software measurement, 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, storing and analyzing them. Data analysis provides information to decision making, supporting the identification of appropriate actions. Finally, the measurement process and its products should be evaluated to identify potential improvements [Barcellos *et al.* 2013].

The measurement process is complex, and we need tools to support it. Moreover, ideally, these tools must be integrated. However, integrating tools is not an easy task. Integration involves several dimensions [Izza 2009], among which we highlight two: layer and level. As for *layers*, integration can address one or several information system layers. *Data integration* deals with moving or federating data between multiple data stores. Integration at this layer assumes bypassing the application logic and manipulating data directly in the database, through its native interface. *Message* or *service integration* addresses messages exchange between the integrated applications. *Process integration* views enterprises as a set of interrelated processes and it is responsible for handling message flows, implementing rules and defining the overall process execution. It constitutes the most complex integration approach.

Regarding integration *levels*, four main levels can be distinguished: *hardware*, *platform*, *syntactical* and *semantic* levels. Hardware level covers differences in computer hardware, networks, etc. Platform level encompasses differences in operating system, database platform, etc. Syntactical level addresses the way the data model and operation signatures are written down. Semantic level deals with the intended meaning of the concepts in a data schema or operation signature. Each level depends on the previous one, so it is not possible to consider semantics if syntax is not considered yet.

In this paper, for semantically integrating measurement-related tools, we used an ontology-based approach called OBA-MSI (Ontology-Based Approach for Measurement System Integration) [Fonseca *et al.* 2016]. OBA-MSI considers a holist view of the software measurement process and uses software measurement ontologies to address semantic conflicts while integrating measurement-related tools. OBA-MSI considers the integration process as a software process and it is centered on the requirements elicitation and analysis phases. Requirements elicitation is conducted by using a goal-based approach that follows GQM (Goal Question Metric) [Basili *et al.* 1994] principles to guide software measurement aligned to organizational goals. Semantic integration occurs mainly in the analysis phase and it is carried out based on the Reference Software Measurement Ontology (RSMO) [Barcellos *et al.* 2013], which describes the conceptualization of the software measurement domain, and on the Software Measurement Task Ontology (SMTO) [Barcellos and Falbo 2013], which describes the main activities of the software measurement process, their inputs and outputs, being consistent with RSMO. Besides supporting tools integration, OBA-MSI helps organizations to define an appropriate measurement process or to improve an existing one. Figure 1 presents OBA-MSI process.

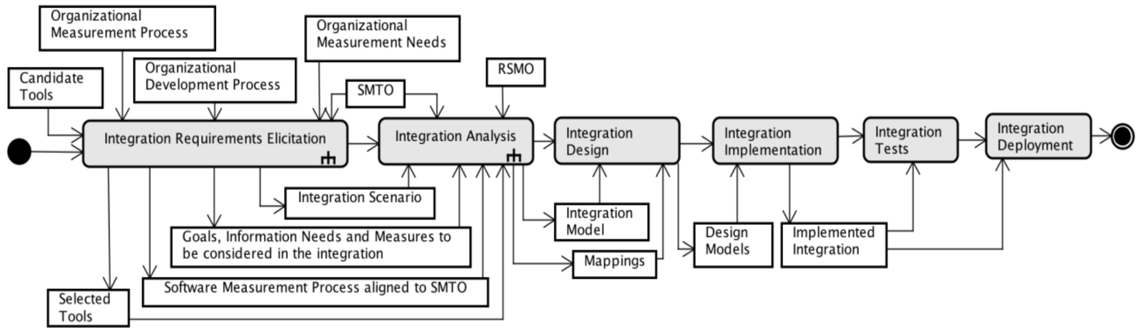

Organizational Measurement Process
Organizational Measurement Needs
Candidate Tools
Organizational Development Process
SMTO
RSMO
Integration Requirements Elicitation
Integration Analysis
Integration Design
Integration Implementation
Integration Tests
Integration Deployment
Integration Scenario
Goals, Information Needs and Measures to be considered in the integration
Software Measurement Process aligned to SMTO
Selected Tools
Integration Model
Mappings
Design Models
Implemented Integration

Figure 1. OBA-MSI process.

The OBA-MSI process starts with the ***Integration Requirements Elicitation*** phase, when the integration scenario is produced, indicating, among others, the measurement activities to be supported by the integration initiative and he tools to be integrated. OBA-MSI advocates that, to be properly supported by the integrated solution, the measurement process should be appropriately defined. Thus, it should be aligned to the measurement process established in SMTO. Figure 2 details this phase.

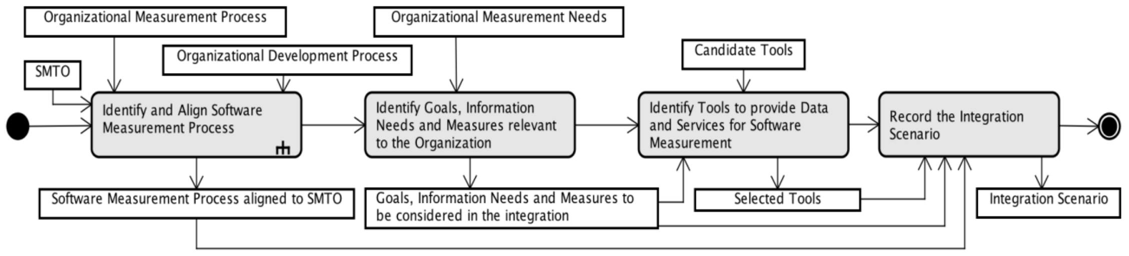

Organizational Measurement Process
Organizational Measurement Needs
Organizational Development Process
Candidate Tools
SMTO
Identify and Align Software Measurement Process
Identify Goals, Information Needs and Measures relevant to the Organization
Identify Tools to provide Data and Services for Software Measurement
Record the Integration Scenario
Software Measurement Process aligned to SMTO
Goals, Information Needs and Measures to be considered in the integration
Selected Tools
Integration Scenario

Figure 2. OBA-MSI – Integration Requirements Elicitation phase.

The ***Identify and Align Software Measurement Process*** activity deals with the alignment of the organizational software measurement process to SMTO to ensure that it includes all activities necessary for measurement to be properly carried out and that they are properly defined. For aligning the organizational software measurement process to SMTO, it is necessary to verify the existence of organizational software measurement process. Some organizations have a defined measurement process, while others carry out measurement activities implicitly during their development process. There are also organizations that do not perform software measurement and will start the practice from the tool integration initiative. An organization with a defined measurement process should align it to SMTO. If the organization does not have a defined measurement process, but performs measurement along its development process, the activities in which measurement takes place must be identified and, then, the measurement process must be defined and aligned to SMTO. Finally, if the organization does not have a defined measurement process and does not perform software measurement, it should define the software measurement process from SMTO, i.e., SMTO must be used as a basis to establish the organizational software measurement process.

Once the software measurement process is consistent with SMTO, it is necessary to ***Identify Goals, Information Needs and Measures relevant to the Organization*** to ensure the alignment between measurement and organization's goals for providing useful information. In this sense, in order to integrate tools and aggregate value to the organization, the goals should be established before starting measure. For this, organization's business goals relevant to measurement must be identified. From them, measurement goals may be derived, determining which information needs must be attained and the necessary measures for it. Since in OBA-MSI the identification of goals, information needs and measures should be aligned to the tools to be integrated, the established goals must be liable from measures that can be obtained from the tools. Therefore, this activity should be performed iteratively with the next one.

Identify Tools to provide Data and Services for Software Measurement is the activity in which the tools to be integrated are selected. Tools must be analyzed aiming at supporting goal monitoring by providing the required measures. Tools that the organization already uses, as well as others unused until then, should be considered. Among the tools used by the organization, it should be considered the ones that support measurement-related activities, even if they are not tools dedicated for this purpose.

Record the Integration Scenario consists in recording the results of the previous activities in the integration scenario, including: goals, information needs and measures to be addressed by the integration initiative, tools to be integrated, domains involved in the integration initiative (domains in which measurement will be applied), and activities of the measurement process that will be supported by the integrated solution.

Once defined the integration scenario, the ***Integration Analysis*** phase can start. In this phase, integration models modeling structural and behavioral aspects of the integration at conceptual level should be established considering semantic aspects and taking the integration scenario into account. Structural models are used to address data integration. Behavioral models, in turn, support service and process integration. First, the tools' conceptual models relevant to the integration must be retrieved, as well as the RSMO e STMO fragments to be used. Then, the ontologies are used as basis to vertical mappings (VMs), which relate tools' elements (e.g., concepts and relations) and ontologies' elements to assign meaning to the tools' elements. Once VMs are established, the integration model is built based on the ontologies and the tools' models in a way that each element of the integration model has a meaning. Next, horizontal mappings (HMs) between the elements of the tools and of the integration model are established to define how the tools will be seen in the integration solution and how the interaction between them will occur.

Finally, the ***Integration Design**, **Implementation**, **Tests** and **Deployment*** activities should be performed. There are several ways of building an integration solution, but OBA-MSI does not commit to any specific one.

3. Applying OBA-MSI in a Software Organization

OBA-MSI was used in an integration initiative to support the software measurement process at the Software Development Extension Laboratory (LEDS) [LEDS 2016], a software development organizational unit of a Brazilian Federal Institute. LEDS has 20 members and develops software to Government and Industry. It follows Scrum principles and adopts tools to support project management and software development. The LEDS' software manager reported the need of obtaining data for monitoring software projects and product quality. According to him, required data were scattered among tools, hindering access to them. Besides, getting additional data derived from data provided by the tools would be useful to support decision making.

As described in the previous section, the first OBA-MSI activity (***Identify and Align Software Measurement Process***) concerns the alignment of the organizational measurement process with SMTO. At the time OBA-MSI was applied, the organization did not have a measurement process formally defined. Thus, following OBA-MSI, we analyzed the organization's development process in order to identify measurement-related activities and we found some related to data collection. For instance, the development process had the *Estimate Project* activity, wherein values are estimated to team velocity, user stories size and number of story points. The process had also the *Perform Continuous Integration* activity, in which source code is integrated and data regarding the source code is collected and stored. In order to make the measurement activity explicit, we split the last activity into *Perform Continuous Integration*, which deals with the continuous integration itself, and *Measure Source Code,* being devoted to perform measurements regarding source code. Figure 3 presents a fragment of the development process with measurement related activities identified (in grey).

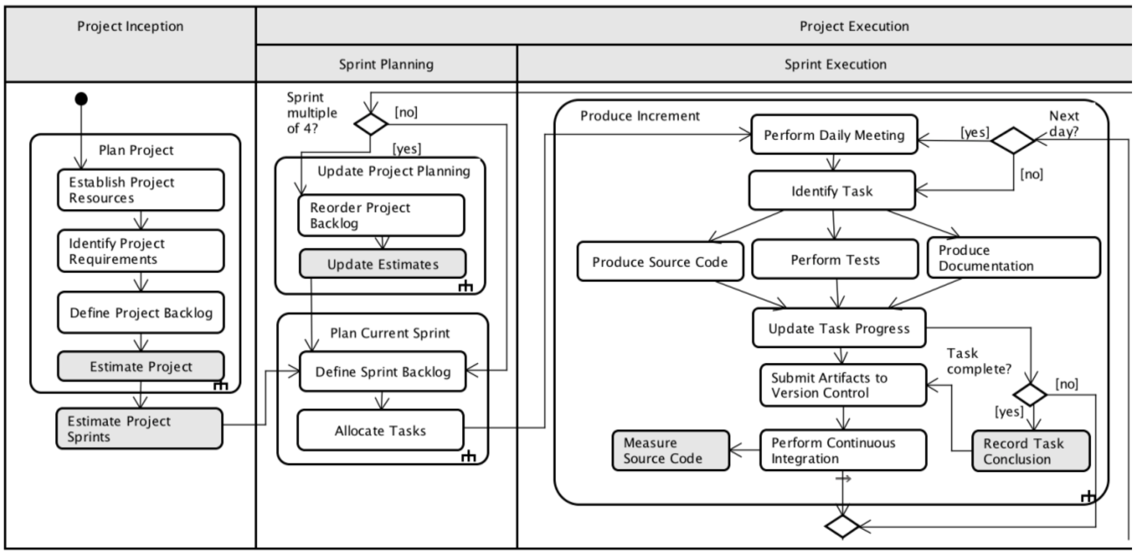


Figure 3. Fragment of the Development Process with measurement activities identified.

After identifying the measurement-related activities, we defined the organizational measurement process considering the identified activities and SMTO. As defined by OBA-MSI, we started by mapping the measurement-related activities with the SMTO activities. For instance, *Estimate Project* from the development process was mapped to *Perform Measurement,* since during project estimation, estimated values are attributed to team velocity, user stories size and number of story points. The mapping revealed that only activities related to the *Perform Measurements* SMTO activity were clearly identified in the LEDS' development process. However, although not defined in the development process, activities related to the other SMTO activities (*Plan Measurement* and *Analyze Measurements*) were also performed at LEDS. For instance, when starting a project, the manager planned targets to be achieved, such as the desired value interval to source code duplications rate in the project. At the end of a sprint or of the project, he checked if the established targets were achieved. Thus, taking the three main SMTO activities as basis, we detailed each one of them considering the identified measurement-related activities or SMTO sub-activities. Figure 4 shows the measurement process defined for LEDS. Activities from the development process are in grey; the ones from SMTO are in white.

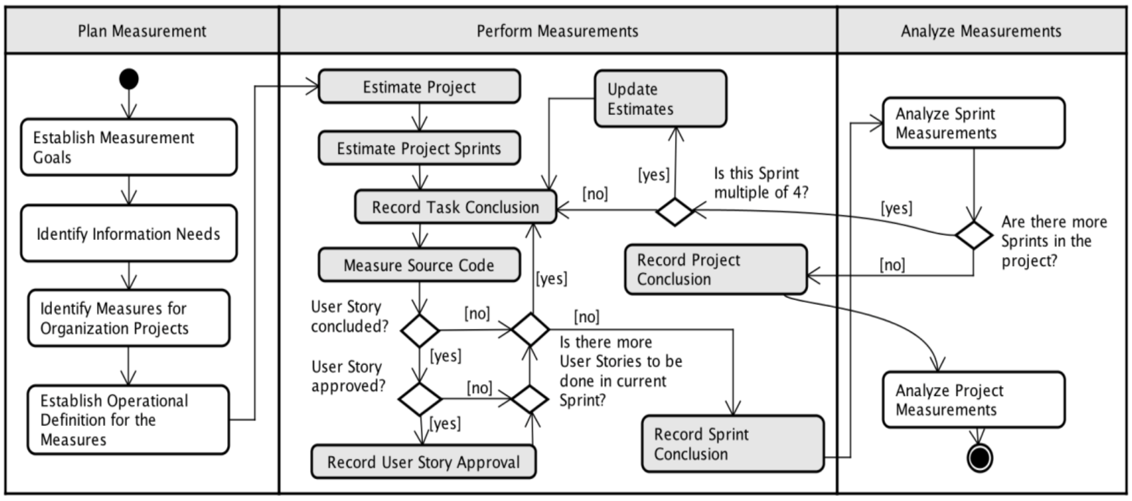
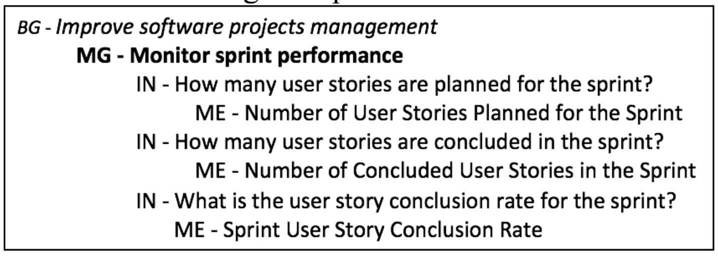

Plan Measurement
Perform Measurements
Analyze Measurements
Establish Measurement Goals
Identify Information Needs
Identify Measures for Organization Projects
Establish Operational Definition for the Measures
Estimate Project
Estimate Project Sprints
Record Task Conclusion
Measure Source Code
User Story concluded?
User Story approved?
Record User Story Approval
Is there more User Stories to be done in current Sprint?
Update Estimates
Is this Sprint multiple of 4?
Record Project Conclusion
Record Sprint Conclusion
[yes]
[no]
Analyze Sprint Measurements
Are there more Sprints in the project?
Analyze Project Measurements

Figure 4. Measurement Process defined from the LEDS Development Process and SMTO.

Although the defined measurement process is not the same as the process established by SMTO, they are aligned. Since there were not explicitly defined activities related to *Plan Measurement*, we defined them considering the SMTO *Plan Measurement* sub-activities. To address the *Perform Measurements* activity, the measurement activities identified from the LEDS' development process were used. In Figure 4, each activity in the *Perform Measurements* partition is a representation of the *Perform Measurements* SMTO activity considering a specific set of measures. To address *Analyze Measurements*, analogous to *Perform Measurements*, we included different activities to different set of measures. Thus, each activity in *Analyze Measurements* partition is a representation of the *Analyze Measurements* SMTO activity, dealing with measurements related to a sprint or to the project as a whole. It is worth noticing that the measurement-related activities identified from the development process and included in the measurement process contributes to establish a direct connection between the LEDS' measurement and development processes, making explicit in which points of the development process measurements should occur.

Aiming to establish the goals to be monitored and the measures to be addressed by the integration solution, the ***Identify Goals, Information Needs and Measures relevant to the Organization*** activity was performed. As defined in OBA-MSI, it was performed iteratively with the selection of the tools to be integrated, aiming to ensure that the identified measures could be obtained from the selected tools. Two business goals (BG) were identified: *Improve software project management* and *Improve source code quality*. From them, 7 measurement goals (MG), 34 information needs (IN) and 34 measures (ME) were identified. Figure 5 presents some of them.



BG - Improve software projects management

MG - Monitor sprint performance

IN - How many user stories are planned for the sprint?

ME - Number of User Stories Planned for the Sprint

IN - How many user stories are concluded in the sprint?

ME - Number of Concluded User Stories in the Sprint

IN - What is the user story conclusion rate for the sprint?

ME - Sprint User Story Conclusion Rate

Figure 5. Some of the identified Goals, Information Needs and Measures.

In the ***Identify Tools to provide Data and Services for Software Measurement*** activity, we followed guidelines provided by OBA-MSI to select the tools to be integrated. We identified the tools used by LEDS for supporting the measurement-related activities. Taiga (*http://taiga.io/*), a tool supporting agile project management, and SonarQube (*http://www.sonarqube.org/),* which supports source code quality evaluation, were selected. Since none of these tools was developed to support the measurement process, a specific software measurement tool had to be selected. We chose SoMeSPC (*http://github.com/nemo-ufes/SoMeSPC*), a tool supporting software measurement and statistical process control, as the specific measurement tool to be integrated. SoMeSPC was developed by using a Reference Software Measurement Architecture based on RSMO [Maretto and Barcellos 2013]. After selecting the tools, the integration scenario was established from the results of the activities performed so far and considering as integration domains measurement applied to agile project management and coding.

Next, we performed the ***Integration Analysis*** phase. We started by selecting the RSMO and SMTO fragments to be used. The entire SMTO, which is used in OBA-MSI to support integration at service and process layers, was selected, because the integration should address all activities of the software measurement process. As for RSMO, which is used to support integration at data layer, we selected the fragment relevant to the integration scenario. RSMO reuses concepts of the Software Process Ontology (SPO) [Briguente *et al.* 2011] and of the Software Organization Ontology (SOO) [Barcellos and Falbo 2009]. Since these ontologies do not address agile software processes aspects (Scrum), it was necessary to extend them and integrate the concepts related to agile process (concepts in grey in Figure 6) to the RSMO fragment to be used. Figure 6 presents part of the RSMO fragment used in the integration initiative.

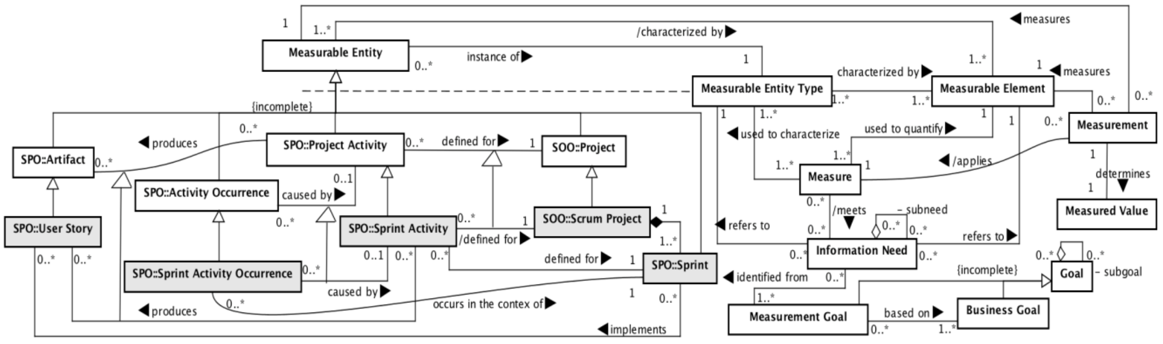


Figure 6. Part of the RSMO fragment used in the integration.

A *Measurable Entity* is anything that can be measured (e.g., a project) and can be classified according to types (*Measurable Entity Type*), such as Project. Measurable Entities are characterized by *Measurable Elements*, which are properties that can measured (e.g., size, cost). Measurable Elements are quantified by *Measures* (e.g., number of function points quantifies size). *Measurement* is the act of attribute a *Measured Value* to a Measure. Considering the integration scenario, some Measurable Entities are particularly relevant, since they represent the entities that will be measured in the integration solution. A *Scrum Project* is composed by *Sprints*. A *Sprint Activity* is a type of *Project Activity* defined for a Sprint. Sprint Activities can produce *User Stories*, a type of *Artifact*. User Stories produced in Sprint Activities are implemented in the same Sprint for which the Sprint Activity is defined. A *Sprint Activity Occurrence* is a type of *Activity Occurrence* that occurs in the context of a Sprint and is caused by a Sprint Activity. A *Business Goal* is a *Goal* that expresses the intention for which strategic actions are planned and performed. *Measurement Goals* express the intention for measurement actions and should be defined based on Business Goals. An *Information Need* describes the information necessary to monitor Measurement Goals and is met by Measures.

After selecting the ontologies fragments, we obtained the relevant fragments of the tools' conceptual structural models by analyzing the tools and their documentation. Figure 7 depicts a fragment of the Taiga's conceptual model used. A *Project* is composed by *Sprints* in which *User Stories* are implemented. *Tasks* are defined to implement User Stories and have *Status*, which can be *new*, *in progress* or *closed*. *Project Stats Detail* and *Sprint Stats Detail* store consolidated data about a Project and a Sprint, respectively.

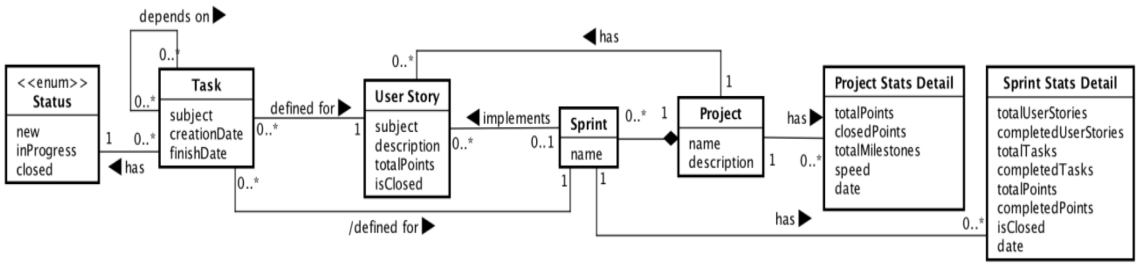


Figure 7. Fragment of Taiga conceptual model.

Figure 8 shows a fragment of the SonarQube's conceptual model used. A *Resource* represents entities that can be submitted to code analysis. A Resource has a *Qualifier* that indicates its type, such as Software Project, Software Module and Developer. A *Metric* is a measure and has a *Domain*, which refers to the property that a metric measures (e.g., complexity, size). A *Measure* is the measurement of a Resource by applying a Metric.

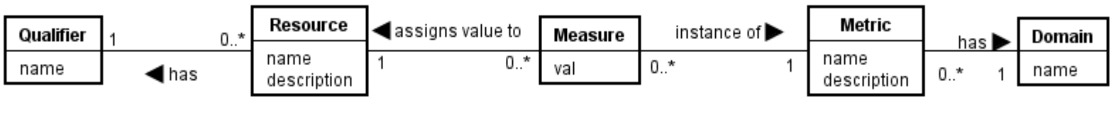


Figure 8 – Fragment of SonarQube conceptual model.

Figure 9 shows a fragment of the SoMeSPC's conceptual model used. For better visualization, attributes were omitted. Since SoMeSPC was developed from a Software Measurement Reference Architecture defined based on RSMO, its conceptual model is consistent with RSMO. In Figure 9, the only concept without a direct matching in

RSMO is *Numeric Value*, specialized from *Measured Value* and defined to allow recording numeric values collected to measures.

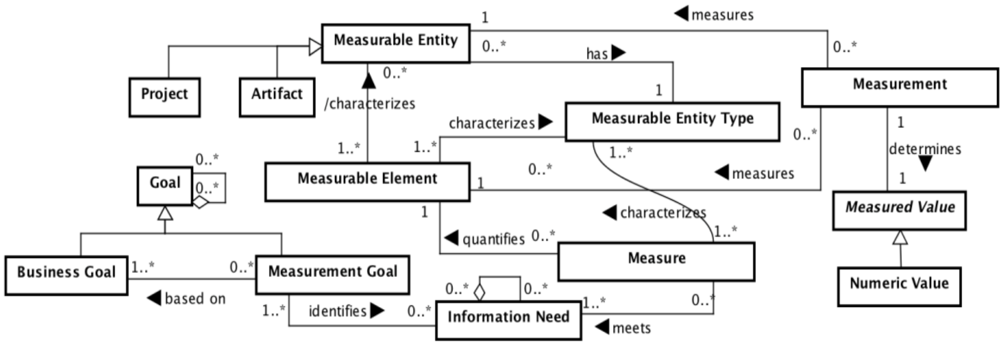


Figure 9. Fragment of SoMeSPC conceptual model.

Once the conceptual models are identified, it is necessary to perform vertical mappings (VMs) to assign semantics to the tools' elements by relating them to RSMO elements. VMs allows assigning semantics and also identifying the semantic equivalences among the tools' elements. VMs must be done to concepts and relationships. Table 1 presents some VMs among concepts.

Table 1. Examples of Vertical Mappings among Concepts.

RSMO	SoMeSPC	Taiga	SonarQube
Sprint	-	Sprint	-
User Story	-	User Story	-
Sprint Activity	-	Task, when status = new	-
Sprint Activity Occurrence	-	Task, when status = inProgress or closed	
Measurable Entity	Measurable Entity	-	Resource
Measurable Entity Type	Measurable Entity Type	-	Qualifier
Measurable Element	Measurable Element	-	Domain
Measure	Measure	-	Metric
Measurement	Measurement	-	Measure
Measured Value	Numeric Value	-	Measure.val

VMs can be direct, i.e., a concept in the ontology is mapped to a class in the tools' conceptual models (e.g., *Scrum Project* in RSMO is mapped to *Project* in Taiga). However, there are situations in which mappings are not direct. For instance, the RSMO concept *Sprint Activity Occurrence* is mapped to *Task* in Taiga only when the Task is *in progress* or *finished*. In Table 1 it is possible to notice that *Measure* and *Measured Value* concepts (RSMO) were not mapped to Taiga concepts. In fact, there are attributes in some Taiga's classes that are measures, i.e., the attributes are instances of the *Measure* concept. In Figure 7, *totalPoints* in *User Story*, and *totalUserStories, completedUserStories, totalTasks, completedTasks, iocaineDoses, totalPoints* and *completedPoints* in *Sprint Stats Detail* are examples of attributes mapped as instances of the *Measure* concept. Values assigned to these attributes are *Measured Values*. These attributes helped us to identify measures possible to be obtained from Taiga. For instance, from the attributes *totalUserStories* and *completedUserStories* in *Sprint Stats Detail* we identified the measures *Number of User Stories Planned for the Sprint* and *Number of Concluded User Stories in the Sprint*, and also *Sprint User Story Conclusion Rate*, obtained by *Number of Concluded User Stories in the Sprint/ Number of User Stories Planned for the Sprint.*

After VMs, we built the integration model. First, we included the RSMO concepts relevant to integration. Next, concepts and relationships present in the tools' conceptual models but not present in RSMO were added to the model. Thus, some multiplicities established in RSMO were adjusted, since they should be more restrictive in the integration model. Last, concepts and relationships required by the integration solution and that do not exist neither in RSMO nor in the tools were added. Figure 10 depicts a fragment of the integration structural model. *Sprint Stats Detail* and *Project Stats Detail* were included from Taiga, *Numeric Value* from SoMeSPC and the others from RSMO. Ellipses indicate cardinalities changed because Taiga is more restrictive than RSMO.

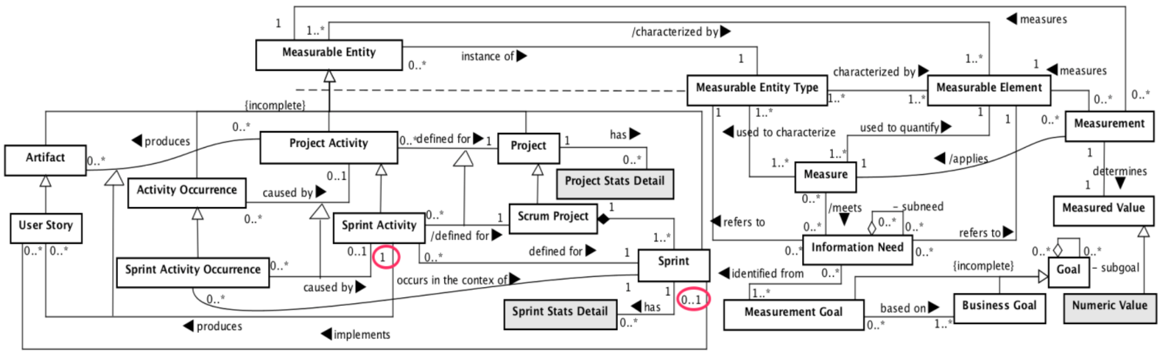


Figure 10. Fragment of the Integration Conceptual Model.

With the integration model in hands, horizontal mappings (HMs) were performed to address the concepts and relationships absent in the ontology and introduced in the integration model from the tools. Table 2 presents some of the HMs among concepts.

Table 2. Examples of Horizontal Mappings among Concepts.

Integration Model	SoMeSPC	Taiga	SonarQube
Numeric Value	Numeric Value	-	-
Sprint Stats Detail	-	Sprint Stats Detail	-

The integration conceptual model, obtained by using RSMO as interlingua to interoperability, enables the integration at data layer. However, service and process layers should also be addressed in an integration initiative [Calhau and Falbo 2010]. Thus, following OBA-MSI guidelines to address integration at service and process layers, we identified services available in the tools' APIs able to support SMTO measurement process activities. Table 3 presents, as examples, the mappings between sub-activities of *Perform Measurements* in SMTO and the tools' services able to support them. The use of services to tools communication addresses integration at the service layer, since services are used to exchange data.

Table 3. Mappings between APIs services and SMTO activities.

SMTO Activity (Perform Measurement sub-activities)	Service	Tool
Select Measurement Plan Item for Measurement	Project Measurement Plan Registration	SoMeSPC
Select Entity for Measurement	Project Measurement Plan Registration	SoMeSPC
	Project, Sprint, User Story and Task Registration	Taiga
	Source Code Registration	SonarQube
Collect Data	Measurement Registration	SoMeSPC
	Project, Sprint, User Story and Task Registration	Taiga
	Source Code Registration	SonarQube

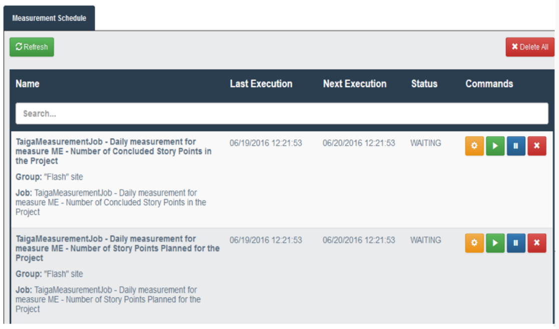
Considering the mappings between services and SMTO activities and the alignment of the organizational measurement process with SMTO (established when the measurement process was defined based on SMTO), the services mapped to SMTO activities can be mapped to the correspondent LEDS' measurement process activities, indicating which activities are supported by the tools' services. Thus, once tool integration is performed considering the identified services, an integrated support is obtained for the measurement process, covering also the integration at the process layer.

Integration design and ***implementation*** were done by the means of a mediator, which was developed based on the integration model and the identified services. The mediator consists of a Java web application developed as a SoMeSPC module extension. It is responsible for coordinating services to support all activities of the measurement process, considering a holistic view of it. It provides three main features:

(i) A wizard for guiding the definition of Project Measurement Plans, as a result of the Plan Measurement activity. User is guided step-by-step from goals selection (each measurement goal is associated to information needs and to measures identified during *Integration Requirements Elicitation* phase) to the establishment of operational definition of measures. For each measure included in the Project Measurement Plan, a job is created to automatically collect data for the measure, considering the periodicity indicated in its operational definition.

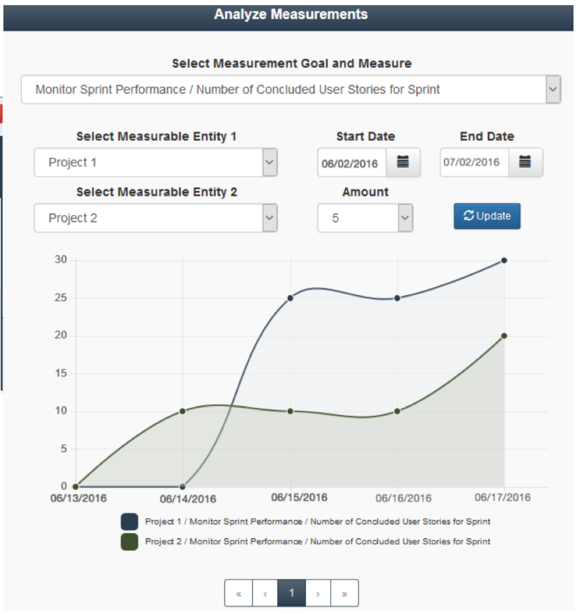
(ii) A panel to manage (execute, pause or delete) the created jobs to collect data from the integrated tools. Figure 11(a) illustrates a partial view of the panel.

(iii) A goal-based analysis feature that renders measured values into charts, supporting measurement analysis by providing useful information to monitor the established goals and to make decisions. Figure 11(b) illustrates selected data represented in a chart to support measurements analysis. In the figure, data related to the number of concluded user stories for sprints of two projects are plotted, providing information regarding the sprints performance.



Name	Last Execution	Next Execution	Status	Commands
TaigaMeasurementJob - Daily measurement for measure ME - Number of Concluded Story Points in the Project **Group:** "Flash" site **Job:** TaigaMeasurementJob - Daily measurement for measure ME - Number of Concluded Story Points in the Project	06/19/2016 12:21:53	06/20/2016 12:21:53	WAITING	
TaigaMeasurementJob - Daily measurement for measure ME - Number of Story Points Planned for the Project **Group:** "Flash" site **Job:** TaigaMeasurementJob - Daily measurement for measure ME - Number of Story Points Planned for the Project	06/19/2016 12:21:53	06/20/2016 12:21:53	WAITING	

(a)


Analyze Measurements

Select Measurement Goal and Measure: Monitor Sprint Performance / Number of Concluded User Stories for Sprint

Select Measurable Entity 1: Project 1

Start Date: 06/02/2016

End Date: 07/02/2016

Select Measurable Entity 2: Project 2

Amount: 5

Update
30
25
20
15
10
5
0
06/13/2016
06/14/2016
06/15/2016
06/16/2016
06/17/2016
Project 1 / Monitor Sprint Performance / Number of Concluded User Stories for Sprint
Project 2 / Monitor Sprint Performance / Number of Concluded User Stories for Sprint
« ‹ 1 › »

(b)

Figure 11. (a) Job Management Feature (b) Goal-based analysis feature.

The integrated solution was deployed and made available for the organization usage. A training involving software measurement theory and the functionalities provided by the integrated solution was given to the LEDS members. Some weeks later we applied a survey in order to collect feedback from the team and the manager. The survey goal was to evaluate if the integrated set of tools is able to support the software measurement process and if it added value to the organization. Three indicators were used to verify the goal achievement: (i) functionalities adequacy; (ii) functionalities usefulness; (iii) benefits obtained from the use of the integrated solution when compared to the isolated use of the tools. 7 LEDS members who used the integrated solution participated in the survey. For each functionality (Measurement Plan Definition, Job Management, and Goal-based Analysis) the participants were asked to indicate its usefulness and adequacy. All the functionalities were considered useful and adequate by the participants. The participants were also asked to indicate if the use of the integrated solution provides *more*, *much more*, *the same*, *less*, or *much less* benefits when compared to the isolated use of the tools. 4 participants answered that the use of the integrated solution provides *much more* benefits than the use of the tools in isolation and 3 participants answered that it provides *more* benefits, indicating that the integrated solution adds value to the organization. Finally, the participants were asked about their perception regarding the defined measurement process. They stated that make the measurement process explicit helped understand the process as a whole and measure what is really important to the organization.

After the survey, an interview was conducted with the LEDS' manager. In summary, the manager stated that the defined measurement process and the adopted approach are suitable for LEDS. He said that the integrated solution favors daily data analysis, since it consolidates data and provides information in a single source. Besides, new measures that are not directly provided by the isolated tools are provided by the integrated solution. The manager pointed out that, although the use of the integrated solution is recent, the benefits provided from it have been already perceived. For instance, according to the manager, by using the integrated solution it is easy to compare data related to different projects and teams and use them to identify problems and improve team performance.

5. Related Work

In the literature, there are some tool integration initiatives that support software measurement. We carried out a systematic investigation and identified 12 integration initiatives involving tools integration to support software measurement (see [Fonseca *et al.* 2015a, 2015b]). Most of the investigated initiatives integrates code-related tools. Consequently, most of the measures addressed in the initiatives are code-related measures. None of the found initiatives addresses integration at process layer. Only one of them ([Ghezzi and Gall 2011]) is concerned with semantic aspects. None of the investigated initiatives presented the method followed to perform the integration. Thus, we presumed that they have used ad-hoc approaches for integrating the tools.

In comparison with these 12 initiatives, the integration initiative carried out at LEDS presents some differences that must be highlighted: (i) Besides a code-related tool, tools supporting project management and software measurement were integrated. Thus, other measures than code-related ones are addressed. (ii) The integration covers data, service and process layers. (iii) Reference ontologies (i.e., ontologies with the purpose of making the best possible description of the domain in reality [Guizzardi 2005]) were used to deal with semantic issues. The initiative presented in [Ghezzi and Gall 2011] also use ontologies, but only lightweight ontologies, limited to computational properties concerns. (iv) A systematic approach (OBA-MSI) that guides integration through the use of ontologies to assign semantics to the elements involved in the integration was followed.

6. Final Considerations

Although organizations use several tools to support their processes, one of the problems faced by them is the lack of tools suitable for properly supporting their measurement process. As a consequence, organizations develop their own solutions that can range from electronic spreadsheets to systems devoted to support software measurement. However, inappropriate solutions can compromise data quality and usefulness and even the whole measurement process [Dumke and Ebert, 2007].

Integrating tools to support the measurement process can be a useful solution for many organizations, allowing for the integration of data and services from different sources in order to get useful information to decision making. Integration is a complex task that involves several concerns. A key factor for integration is the sharing of a common understanding between the tools regarding the meaning of the exchanged data and services. In other words, it is important to deal with integration not only at the syntactic level, but also at the semantic level. Among the instruments used to address semantics, ontologies have been acknowledged as an important means to address tool semantic integration. They can be used as an interlingua to map concepts and services used by the different applications.

Considering that, in this paper, we reported the use of OBA-MSI, an ontology-based approach for semantically integrating measurement-related tools, to carry out a tool integration initiative to support the software measurement process at LEDS, aiming to help software project monitoring and software quality improving. Tools supporting agile project management (Taiga), code analysis (SonarQube), and software measurement (SoMeSPC) were integrated by using a mediator.

The main difficulty we faced when applying OBA-MSI in the integration initiative at LEDS regards recovering Taiga's and SonarQube's conceptual models. Although there is documentation available at tool's websites, sometimes the available information was not enough to identify the relevant fragments for the integration. Software measurement knowledge was needed for us to decide which fragments of the tool's conceptual models should be used in the integration. In this sense, it is worth pointing out that, although OBA-MSI provides the steps and guidelines to be followed to integrate tools, it is necessary to make some decisions along the OBA-MSI process that depend on user knowledge of software measurement.

OBA-MSI addresses tool integration considering semantic aspects and alignment to organizational goals. This contributes to increase quality in tool integration initiatives. However, the concern with semantics can imply more effort to perform the integration initial phases than when integrating tools in an ad hoc manner. We advocate that the benefits provided from using OBA-MSI justify the necessary effort to apply it. Even so, aiming to decrease the demanded effort and ease the use of OBA-MSI, we have been working on improving the approach and providing some support to apply it.

Some learned lessons obtained from the initiative can be pointed out: (i) By applying OBA-MSI, the organizational measurement process was made explicit and integrated to the development process. This helped the team to understand the software measurement process, and also contributed to perform the measurement process as part of the development process instead of an isolated process. (ii) Measurement-related tools integration should be guided to organizational and project goals in order to provide useful information and assure that only what really matters is measured. Even Taiga and SonarQube being able to provide other measures, focus on the ones related to the LEDS goals contributed to the success of the initiative. (iii) The use of ontologies as interlingua to tool integration allowed proper integrating the tools, avoiding conflicts about the meaning of the integrated elements. (iv) The integrated set of tools provide more benefits than using them in isolation. The integrated set of tools allows following the measurement process as a whole, connecting measures to goals and using information provided from collected data to monitor goals achievement. Besides, new measures not provide by the tools can be obtained from the integrated solution. (v) Provide automatic measurements and features that guide the user in a step-by-step process made the measurement process easier to be understood, performed and institutionalized.

Currently, we are evaluating new tools to be added to the integrated set of tools and planning using it as a tool to aid software measurement teaching. The integration initiative at LEDS was conducted by one of the researchers involved in the OBA-MSI development. As future work, we plan to apply OBA-MSI to integrate tools without the researchers intervention.

Acknowledgement

This research is funded by the Brazilian Research Funding Agency CNPq (Processes 485368/2013-7 and 461777/2014-2). The authors acknowledge the LEDS for the opportunity of applying OBA-MSI and for the provided feedback.

References

Barcellos, M. P., Falbo, R. A. (2009). Using a Foundational Ontology for Reengineering a Software Enterprise Ontology. In Joint International Workshop on Metamodels (MOST 2009), p. 179-188.

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

Barcellos, M. P. and Falbo, R. de A. (2013). A software measurement task ontology. In *28th Annual ACM Symposium on Applied Computing (SAC 2013)*. ACM Press.

Basili, V. R., Caldiera, G. and Rombach, H. D. (1994). Goal Question Metric Approach. *Encyclopedia of Software Engineering*. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Briguente, A. C. O., Falbo, R. A. and Guizzardi, G. (2011). Using a Foundational Ontology for Reengineering a Software Process Ontology. In: *XXVI Brazilian Symposium on Data Base.*

Calhau, R. F. and Falbo, R. A. (2010). An Ontology-Based Approach for Semantic Integration. In *2010 14th IEEE International Enterprise Distributed Object Computing Conference*. IEEE.

Dumke, R. and Ebert, C. (2007). *Software Measurement: Establish - Extract - Evaluate - Execute*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Florac, W. a. and Carleton, A. D. (1997). *Measuring the software process: statistical process control for software process improvement*. Boston, USA: Addison Wesley.

Fonseca, V. S., Barcellos, M. P. and Falbo, R. D. A. (2016). An Ontology-Based Approach for Integrating Tools Supporting the Software Measurement Process. *Special Issue on Advances in Software Measurement and Measurement Programs of the Science*. Computer Programming Journal (under review).

Fonseca, V. S., Barcellos, M. P. and Falbo, R. de A. (2015a). Tools Integration for Supporting Software Measurement: A Systematic Literature Review. *iSYS - Information Systems Brazilian Journal*, v. 8, n. 4, p. 80–108.

Fonseca, V. S., Barcellos, M. P. and Falbo, R. de A. (2015b). Integration of Software Measurement Supporting Tools: A Mapping Study. In *27th International Conf.erence on Software Engineering and Knowledge Engineering (SEKE 2015).*

Ghezzi, G. and Gall, H. C. (2011). SOFAS: A Lightweight Architecture for Software Analysis as a Service. *9th Working IEEE/IFIP Conf on Software Architecture*, p. 93–102.

Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models, ISBN 90-75176-81-3, Universal Press, The Netherlands, 2005.

ISO/IEC (2007). *IEEE Standard Adoption of ISO/IEC 15939:2007—Systems and Software Engineering—Measurement Process*.

ISO/IEC (2008). I*EEE Standard Adoption of ISO/IEC 12207:2008 — Systems and Software Engineering — Software Life Cycle Processes*.

Izza, S. (2009). Integration of industrial information systems: from syntactic to semantic integration approaches. *Enterprise Information Systems*, v. 3, n. 1, p. 1–57.

LEDS (2016). Software Development Extension Laboratory. http://leds.sr.ifes.edu.br/

Maretto, C. X. and Barcellos, M. P. (2013). A Levels-based Approach for Defining Software Measurement Architectures. *Clei Electronic Journal*, v. 14, n. 3, p. 27.

McGarry, J., Card, D., Jones, C., et al. (2002). *Practical Software Measurement: Objective information for decision makers*. Boston, USA: Addison Wesley.

Nardi, J. C., Falbo, R. A. and Almeida, J. P. A. (2013). A Panorama of the Semantic EAI Initiatives and the Adoption of Ontologies by these Initiatives. In: *IWEI 2013, LNBIP 144*. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer Berlin Heidelberg. v. 144p. 198–211.

Pokraev, S. (2009). Model-Driven Semantic Integration of Service-Oriented Applications. University of Twente.

SEI (2010). CMMI® for Development, Version 1.3.

Solingen, R. and Berghout, E. (1999). The Goal/Question/Metric Method: a practical guide for quality improvement of software development. *A Practical Guide for Quality Improvement of Software Development. New York, McCraw-Hill Publishers*, p. 216.