A Well-founded Software Process Behavior Ontology to Support Business Goals Monitoring in High Maturity Software Organizations

Monalessa Perini Barcellos[1, 2], Ricardo de Almeida Falbo[1], Ana Regina Rocha[2]

[1]Department of Computer Science,
Federal University of Espírito Santo
Vitoria – Brazil

[2]COPPE
Federal University of Rio de Janeiro
Rio de Janeiro – Brazil

monalessa@inf.ufes.br, falbo@inf.ufes.br, darocha@cos.ufrj.br

Abstract - **Organizations define strategies and establish business goals aiming to be competitive. The process performance analysis supports goals monitoring, allowing to detect and to treat threats to goals achievement. In this context, measurement is essential. The collected data for measures are used to analyze the process performance and to guide informed decisions that lead to the achievement of business and technical goals. For software organizations, the process performance analysis is a high maturity practice. In this context, although there are several standards that address the importance of software measurement and its use in process performance analysis, the vocabulary used by these standards concerning software measurement is diverse. In order to establish a conceptualization regarding this domain, we developed a Software Measurement Ontology (SMO), grounded in the Unified Foundational Ontology. In this paper, we present a fragment of SMO with focus on software process behavior analysis.**

Keywords - **Software Process Behavior Ontology, Software Measurement, Software Measurement Ontology, Domain Ontologies, Foundational Ontology.**

I. INTRODUCTION

Competitiveness is a determining factor to the success of organizations. Each organization has an explicit or implicit competitive strategy [1]. Typically, the strategy of an organization is materialized in its strategic planning, which defines the organization's business goals, established with focus on organizational competitiveness and considering a certain period of time. Goals provide orientation and reflect the desired conditions for improving the organization global performance, guiding both decision-making and daily activities [2]. Thus, actions related to the goals defined in the strategic planning are planned and carried out aiming achieving the goals.

In addition to defining goals, it is important to monitor them continuously, verifying if they are being achieved [3]. The monitoring of the goals achievement should be based on organizational performance analysis, which is accomplished from the processes performance analysis [4]. Process performance analysis consists of analyzing data collected throughout the process executions to know process behavior and provide guidelines for taking corrective and improvement actions, aiming to achieve the established goals [5].

Since the process performance analysis uses data from the processes executions, measurement is one of its main pillars. To obtain and use quantitative information about the processes, a measurement program should be carried out. The measurement program defines measures aligned to organizational goals. These measures, in turn, are collected and analyzed to provide the information needed for decision-making.

In the context of software organizations, the organizational performance monitoring becomes more rigorous as the organizational maturity increases. Organizations that are in high maturity levels (characterized by levels 4 and 5 of the Capability Maturity Model Integration – CMMI [6]) perform measurement in order to control statistically their processes. Statistical Process Control (SPC) is performed to analyze the processes behavior, in order that they be managed, predicted, controlled and improved to meet the organizational business and technical goals [5].

There are several process quality standards and maturity models, such as ISO/IEC 12207 [7], CMMI [6] and MR MPS.BR [8], which support software organizations in achieving maturity in software development, including software measurement aspects. There are also several standards and methodologies devoted specifically to assist organizations in defining their software measurement process, such as ISO/IEC 15939 [9] and PSM [10]. Unfortunately, the vocabulary used by those standards and models, and as a consequence by the software organizations, is diverse. This leads to misunderstanding and problems related to the jointly use of different standards.

To deal with these problems, it is important to establish a common conceptualization regarding the software measurement domain. Thus, we need 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 [11].

A 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. Ideally, a reference ontology should be represented by

an ontologically well-founded language [11]. Such a language must explicitly commit to fundamental ontological distinctions in their metamodels, given by a foundational ontology [12].

In order to deal with the problem of vocabulary diversity in related standards, we decided to develop a Software Measurement Ontology (SMO). This reference ontology is represented using the ontologically well-founded UML modeling profile proposed in [12, 13]. This profile comprises a number of stereotyped classes and relations implementing a metamodel that reflects the structure and axiomatization of the Unified Foundational Ontology (UFO) [12, 13].

The SMO is partially presented in this paper. The focus here is on measurement at high maturity levels, more specifically on software process behavior analysis. This domain reference ontology was built based on the Unified Foundational Ontology (UFO) [12, 13]. Besides, the SMO was developed based on the vocabulary used in several standards and specific requirements of software measurement at high maturity levels. These requirements were identified in a study based on systematic review of the literature.

This paper is organized as follows. Section II discusses software measurement and process behavior analysis. Section III presents the concepts of the Unified Foundational Ontology considered relevant for this paper. Section IV presents an overview of the Software Measurement Ontology and details its fragment that treats software process behavior. Section V discusses related works, and Section VI presents our conclusions and future work.

II. Software Measurement and Process Behavior Analysis

Software Measurement is a primary tool 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 [9].

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) to consider 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 elements. 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, also following the guidelines established by the corresponding operational definitions. Finally, the measurement process and its products should be evaluated in order to identify potential improvements.

Depending on the organization's maturity level, software measurement is performed in different ways. In 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 [6]. In high 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 [5, 6]. 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.

A process under control is a stable process and, as such, has repeatable behavior. So, it is possible to predict its performance in future executions and, thus, to prepare achievable plans and to improve the process continuously. On the other hand, a process that varies beyond the expected limits is an unstable process and the causes of these variations (said special causes) must be investigated and addressed by improvement actions 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 in order to make it capable should be carried out [5].

Figure 1 summarizes the process behavior analysis using SPC principles. First, it is necessary to understand the business goals recorded in the strategic planning. Next, the processes related to business goals are identified and the measures used to provide quantitative information about the processes performance are defined. Data are collected for the measures, stored and used for analyzing the processes behavior using statistical techniques. If a process is unstable, the special causes should be treated. If it is incapable, it should be changed. Finally, if it is capable, it can be improved continuously [5].

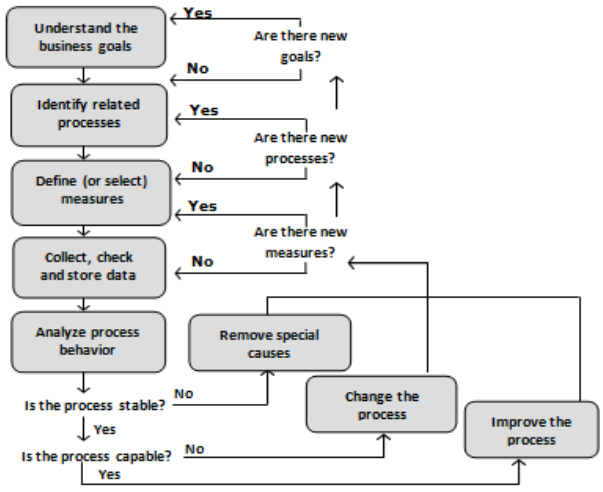

Figure 1. Process behavior analysis (adapted from [5]).

Although SPC is not new to the industry in general, its use in software organizations is recent. In this context, SPC is viewed as an evolution of the software measurement.

However, software measurement is also considered a relatively young discipline. The terminologies used by different measurement approaches and standards are diverse, and the problem of terminology harmonization still needs to be solved in this domain [14]. Besides, there are measurement aspects that are not addressed, mainly aspects of measurement at high maturity levels, which include process behavior analysis. Thus, we need to establish a common conceptualization of the software measurement domain, including aspects related to measurement at high maturity levels. To achieve this common conceptualization, ontologies can be used.

A domain ontology can be used to establish a common conceptualization about certain domain. Thus, a software measurement ontology, providing a coherent set of concepts, relations and axioms constraining their interpretation, is of great value [15]. Furthermore, as discussed in the introduction of this paper, we are interested in a domain reference ontology, grounded in a foundational ontology. So, for developing our Software Measurement Ontology, we decided to use the Unified Foundational Ontology (UFO) [12, 13], briefly presented in the next section.

III. THE UNIFIED FOUNDATIONAL ONTOLOGY

UFO is a foundational ontology that has been developed based on a number of theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. It 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 [12]. 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 [13].

A complete description of UFO falls outside the scope of this paper. However, in the sequel we give a brief explanation of its concepts that are important for this paper. These concepts belong to UFO-A and UFO-C parts. The description is based on [12, 13, 15]. Figure 2 shows a fragment of UFO-A. The concepts that are directly used here are shown detached in grey.

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). The model depicted in Figure 1 focus on universals. Universals can be *first order universals*, i.e., universals whose instances are particulars, or *high order universals*, which are universals whose instances are also universals. *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*[1] 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 of one single endurant (e.g., color). *Relators* depend on a plurality of endurants (e.g., an employment) and, for this reason, provide the material connection between these endurants. 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 relation part-of).

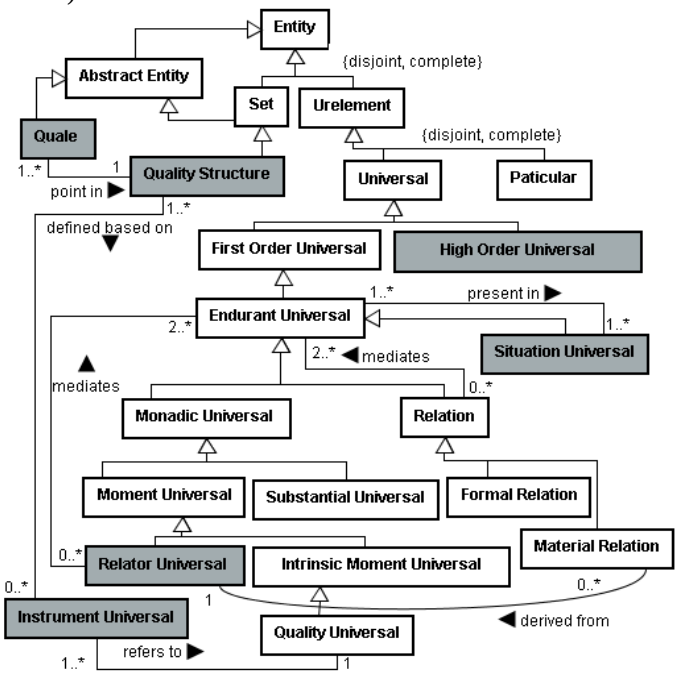

Figure 2. An UFO-A fragment focusing on universals.

A *quality universal* is an intrinsic moment universal that is associated with a *quality structure*. A quality structure can be understood as a measurement structure (or a space of values) in which individual qualities can take their values. A *quale* is a point (a value) in a quality structure. For instance, the quality universal Weight is associated to a space of values that is a liner structure isomorphic to the positive half-line of the real numbers. For the same quality universal, there can be potentially many quality structures associated with it, but a quality structure is always associated with a unique quality universal. An *instrument* (e.g., weight in grams) is used to associate a quality universal to values (qualia) in a quality structure. For a given quality universal, there can be different quality structures associated with different instruments.

[1] The word moment in UFO-A is derived from the german term *Momente* and it bears no relation to the notion of time instant. It is related to the ways things are.

Finally, *Situations* are special types of endurants. They are taken here to be synonymous to what is named *state of affairs* in the literature, i.e., a portion of reality that can be comprehended as a whole (e. g., "John being with fever and influenza").

Figure 3 shows a fragment of UFO including *substantial universals* as well as some concepts of UFO-C.

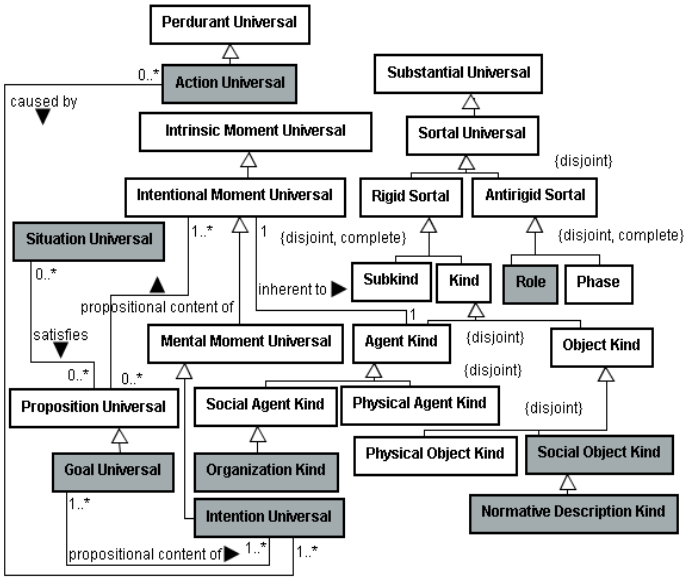

Figure 3. An UFO fragment including concepts from UFO-A and UFO-C.

While persisting in time, substantials 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* (for general substantials) and *subkind*. There are, however, types that a substantial instantiates in some circumstances but not in other circumstances. These are named *phases* and *roles*. A phase is a type instantiated in a given time period but not necessarily in all periods. A role is a type instantiated in a given context, such as the context of a given event participation or a given relation. For instance, Person can be considered a kind, while Child and Adult would be phases of person, and Student would be a role that a person plays when enrolled in some educational institution.

Taking into account kinds, an important distinction in UFO is between agents and objects. According to UFO-C, an *object kind* is a non-agentive substantial universal. Its instances (objects) do not act. They can only participate in actions. Object kinds can be categorized into *physical object* (e.g., Book) and *social object* (e.g., Language). A *normative description kind* is a social object kind whose instances define one or more rules/norms recognized by at least one social agent (e.g., a method describing a set of directives on how to perform some activity within an organization). An *agent kind* is a substantial universal that is capable to perform actions with some intention. Agent kinds can also be further categorized into *physical agent* (e.g., Person) and *social agent* (e.g., Team). *Organization kind* is a specialization of social agent kind

Intentional Moment Universal is a special kind of intrinsic moment universal that are inherent to agents and have a propositional content called *Proposition*. Intentional moments in which the intentionality is "intending something" are called *Intention*. An intention characterizes a situation desired by the agent (e.g., an organization *O* can have the intention "to be successful"). Intentions cause the agent to perform *Actions*. The propositional content of an intention is a *Goal* (e.g., the propositional content of the intention "to be successful" could be "to be among the ten best software organizations of its country"). Intentional moments are related to situations. This relation is defined as follows: a situation in the real world can satisfy the propositional content of an intentional moment, i.e., satisfy, in the logical sense, the proposition that represents the propositional content. For example, the situation "the organization *O* is the fifth best software organization of its country" satisfies the propositional content "to be among the ten best software organizations of its country".

IV. A SOFTWARE MEASUREMENT ONTOLOGY

For developing the Software Measurement Ontology (SMO), we used the SABiO (Systematic Approach for Building Ontologies) method [16]. This method has been used for the last ten years in the development of a number of domain ontologies in areas ranging from Harbor Management to Software Process to Electrocardiogram domain. SABiO prescribes an iterative process comprising the following activities: (i) *purpose identification and requirement specification* that concerns to clearly identify the ontology purpose and its intended uses, i.e., the competence of the ontology by means of competency questions; (ii) *ontology capture*, when relevant concepts, relations, properties and constraints should be identified and organized; (iii) *ontology formalization*, which comprises the definition of formal axioms in First-Order Logic; (iv) *integration of existing ontologies*, which involves the search for existing ontologies with reuse and integration in mind; (v) *ontology evaluation*, for identifying inconsistency as well as verifying the truthfulness with the ontology purpose; (vi) *ontology documentation*.

It is important to highlight that competency questions play a prominent role in this method by defining the scope and purpose of the domain conceptualization being developed, and serving as a testbed for ontology evaluation, since the competency questions are the questions the ontology is supposed to answer [16].

Another important feature of this method is that it suggests the use of a conceptual model written in a UML profile and a dictionary of terms to aid communication with domain experts. Concerning the UML profile, in this paper, we do not use the UML profile originally proposed by SABiO. Since our focus is on developing a domain reference ontology grounded in a foundational ontology, we have used an extension of the ontologically well-founded UML modeling profile proposed in [12, 13]. This profile comprises a number of stereotyped classes and relations implanting a metamodel that reflects the structure and axiomatization of UFO [12, 13].

Since the software measurement domain is strongly related to domains of software processes and organizations, we looked up to ontologies in these domains. We decided to use the software process ontology described in [13] that is already grounded in UFO. Concerning the domain of software organizations, we decided to reuse the software organization ontology proposed by Villela et al. [2]. This ontology, however, was not developed grounded in a foundational ontology, and thus we had to, first, reengineer it [17].

Since the scope of the SMO is very complex, we applied a decomposition mechanism allowing building the ontology in parts. Thus, SMO was divided into six sub-ontologies [15]. The *Measurable Entities & Measures* sub-ontology is the core of the SMO and it is presented in [15]. It treats 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 deals with the alignment of measurement to organizational goals. The *Operational Definition of Measures* sub-ontology addresses the detailed definition of operational aspects of measures, including data collection and analysis. The *Software Measurement* sub-ontology refers to the measurement per se, i.e., collecting and storing data for measures. The *Measurement Results* sub-ontology handles the analysis of the collected data for getting information to support decision making. Finally, the *Software Process Behavior* sub-ontology refers to applying the measurement results in the analysis of the behavior of the organizational software processes.

In this paper we discuss part of the Software Process Behavior sub-ontology, presenting some of its competency questions, conceptual models and axioms. Also, its evaluation is briefly discussed. Since process behavior analysis is strongly related to goals monitoring, we also present a fragment of the Measurement Goals sub-ontology. The definitions of the concepts of the SMO shown in this paper were mainly based on ISO/IEC 15939 [9], PSM [10] and IEEE Std 1061 [18].

A. The Measurement Goals Sub-ontology

Measurement should be aligned to organizational goals in order to produce useful data for analyzing process performance and consequently to monitor goals. Thus, the Measurement Goals sub-ontology should be able to answer, among others, the following competency questions:

CQ1. Based on which business goals is a software goal defined?
CQ2. Based on which goals is a measurement goal defined?
CQ3. Which are the information needs identified from a goal?
CQ4. Which measures can be used as indicators for monitoring the achievement of a goal?
CQ5. Which measures attend an information need?

Figure 4 shows the conceptual model that addresses the competency questions listed above. The concepts reused from the Software Organization Ontology [17] are identified in this model preceded by SOO. The concepts from the Measurable Entities & Measures sub-ontology [15] are identified preceded by MEM. The distinctions made in UFO are shown as stereotypes in the concepts of the SMO, indicating that they are subtypes of concepts of UFO, as defined in [12]. When a concept does not have a stereotype, it means that this concept is of the same type of its super-type.

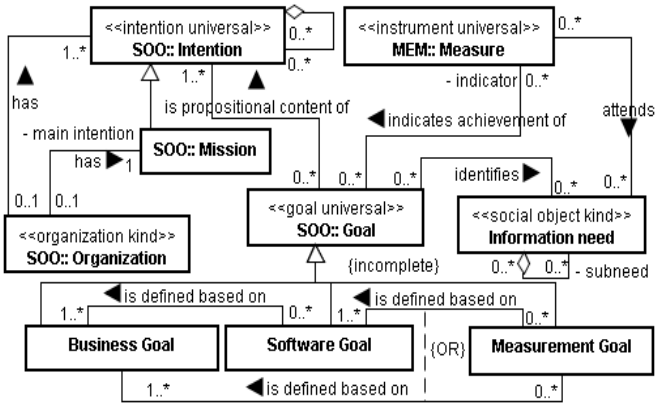

Figure 4. Fragment of the Measurement Goals sub-ontology.

An *Intention* is the purpose for which actions are planned and performed [17]. A *Goal* is the propositional content of an intention [17]. The main intention of an organization is its *Mission.* An *Organization* is a social agent which employs human resources for performing actions to achieve its goals [17].

In the context of software measurement, a goal can be a business goal, a software goal or a measurement goal. A *Business Goal* expresses the intention for which strategic actions are planned and performed (e.g., "increase 10% the number of clients"). A *Software Goal* expresses the intention for which actions related to software area are planned and performed (e.g., "achieve the CMMI level 4"). A *Measurement Goal* expresses the intention for which actions related to software measurement are planned and performed (e. g., "monitor the critical processes behavior"). Software and measurement goals are defined based on business goals. Measurement goals can be also defined directly from software goals.

Information Needs are identified from goals and they are attended by *Measures*. For instance, the measurement goal "improve the adherence to projects plans" could identify the information need "know the requirements stability after their approval by the client", which could be attended by the measure "requirements changing rate".

Measures can be used in order to indicate the achievement of goals. In this case, the measure fulfills the role of an *indicator.* Considering the example cited above, if the measure "requirements changing rate" is used for monitoring the achievement of the goal "improve the adherence to projects plans", then, in this context, it is an indicator.

During the development of the SMO, several constraints were identified and, since the conceptual models are not capable to capture several of them, we defined axioms to make them explicit. We used the axiom classification suggested by SABiO [16] that considers two classes of axioms: *derivation axioms*, which allow new knowledge to be derived from the

previously existing knowledge, and *consolidation axioms*, that define constraints for establishing a relation consistently. In the sequel, we present two axioms of the Measurement Goals sub-ontology.

The axiom MG-A1 is a consolidation axiom and it says that if a measure *m* is an indicator of the achievement of the goal *g*, then there should exist an information need *in*, identified from the goal *g*, that is attended by *m*.

MG-A1: $(\forall m \in Measure, g \in Goal)\ (indicator(m, g) \rightarrow (\exists in \in Information\ Need)\ (identifies(g, in) \wedge (attends(m, in))$

The axiom MG-A2 is a derivation axiom and it says that if a measurement goal *mg* is defined based on the software goal *sg* and *sg* is defined based on the business goal *bg*, then *mg* is also defined based on *bg*.

MG-A2: $(\forall mg \in Measurement\ Goal, sg \in Software\ Goal, bg \in Business\ Goal)\ (isDefinedBasedOn\ (mg, sg) \wedge isDefinedBasedOn\ (sg, bg) \rightarrow isDefinedBasedOn\ (mg, bg))$

B. The Software Process Behavior Sub-ontology

As said before, data are collected for measures and they are analyzed aiming to provide information that support decision-making. In high maturity levels, this information is applied in software process behavior analysis. Thus, the Software Process Behavior sub-ontology should be able to answer, among others, the following competency questions:

CQ1. For a given measure, which is the performance baseline of a standard software process?
CQ2. Which are the lower and upper limits of a process performance baseline?
CQ3. Which measurement analysis has identified a process performance baseline?
CQ4. From which measured values is a process performance baseline determined?
CQ5. In which context is a process performance baseline established?
CQ6. Which are the process performance baselines used to define a process performance model?
CQ7. Which is the specified performance for a standard software process considering a given measure?
CQ8. Which are the lower and upper limits of a specified process performance?
CQ9. Which is the capacity of a standard software process concerning a given measure?
CQ10. From which process performance baseline is a process capability obtained?
CQ11. In relation to which specified process performance is a process capability calculated?
CQ12. Which is the procedure used to determine a process capability?
CQ13. Concerning process behavior, which are the types of standard software process?
CQ14. Which specified process performance does a capable standard software process attend?

Figure 5 shows a fragment of the conceptual model of the Software Process Behavior sub-ontology that addresses the competency questions CQ1 to CQ6. The concepts arising from the Software Process Ontology [12], the Operational Definition of Measures sub-ontology, the Measurement sub-ontology and the Measurement Analysis sub-ontology are respectively identified preceded by SPO, ODM, M and MA.

For talking about process behavior, first, we have to introduce some concepts regarding measurement analysis. A *Measurement Analysis* is an action that analyses *Measurement Results*, i.e. *measured values* collected during a measurement. A *Measurement Analysis* adopts a *Measurement Analysis Procedure* that can suggest the use of analytical methods for representing and analyzing the measured values. *Analytical Method* is sub-kind of *Method*, a concept from Software Process Ontology [13], which describes systematic procedures for performing an activity (a normative description in UFO). Histograms and bar charts are examples of analytical methods. Analytical methods that use principles of statistical control to represent and analyze values are said *Statistical Control Methods*. The XmR and mXmR charts [5] are examples of statistical control methods.

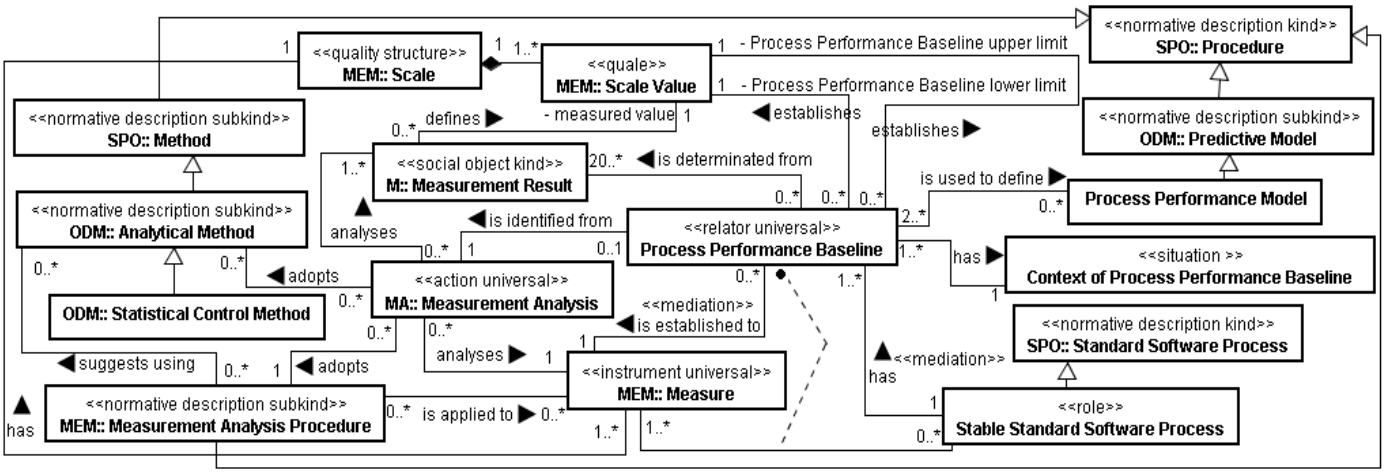

Figure 5. Fragment I of the Software Process Behavior sub-ontology.

Concerning process behavior, in a measurement analysis that adopts a statistical control method, it is possible to identify a *Process Performance Baseline* to a *Stable Standard Software Process* regarding to a *Measure*. Therefore, *Process Performance Baseline* is existentially dependent of both *Stable Standard Software Process* and *Measure*, and it corresponds to a relator universal in UFO. The relations that take place between a relator and the endurants it mediates are called *mediations* in UFO. Thus, the relations *has* (between *Standard Software Process* and *Process Performance Baseline*) and *is established to* (between *Process Performance Baseline* and *Measure*) are mediation relationships.

A standard software process is a description of a type of software process, defined in the context of an organization (e.g., the description of the Requirements Management process of the organization Org). It is a concept from Software Process Ontology [13] and it is a normative description in UFO, since it defines one or more rules/norms recognized by at least the organization that adopts it.

A process performance baseline is identified from twenty or more measurement results. It is the range of results achieved by a *Stable Standard Software Process*, obtained from measured values of a particular measure. This range is used as a reference for process performance analysis and it is defined by two limits: *process performance baseline upper limit* and *process performance baseline lower limit.* The limits values are part of the *Scale* of the measure considered for establishing the baseline. When a standard software process has a process performance baseline, we have a stable standard software process. For instance, consider the analysis of measured values of the measure "requirements change rate", related to the Requirements Management standard software process of the organization Org. Using XmR control chart, this measurement analysis could identify a process performance baseline composed by upper and lower limits 0,1 and 0,25, respectively. Thus, in this context, the Requirements Management process is considered a stable standard process.

A process performance baseline is established in a particular context (*Context of Process Performance Baseline*) that is a situation in UFO. In the previous example, we have the following situation for the first process performance baseline established to the Requirements Management standard process: "The data used to establish the baseline were collected in six small projects with the same team, under usual conditions. In the analysis, two points collected on exceptional situations were excluded".

Process performance baselines are used to define *Process Performance Models*. *Process Performance Model* is a specific type of *Predictive Model*. Predictive model, in turn, is a concept from the Operational Definition of Measure sub-ontology and it is a normative description in UFO. It describes a procedure for predicting the value of a measure by quantifying its relations with others measures (e.g., the Putnam Model ($E = S^3/ Ck^3T^4$) [19] that predicts the measure development effort from the measures of size, time and used technologies). Process performance models use process performance baselines to establish and quantify the relations between measures.

As said before, several axioms were defined in order to make explicit constraints that are not captured in the conceptual models. Regarding to the model shown in Figure 5, among others, the following consolidation axioms hold.

The axiom SPB-A1 says that if a process performance baseline *ppb* is identified from a measurement analysis *ma*, then *ma* should adopt a statistical control method *scm*.

SPB-A1: $(\forall ppb \in$ *Process Performance Baseline*, $ma \in$ *Measurement Analysis*$)$ $(isIdentifiedFrom(ppb, ma) \rightarrow (\exists scm \in$ *Statistical Control Method*$)$ $adopts(ma, scm))$

The axiom SPB-A2 says that if a process performance baseline *ppb* is established to the measure *m* and *ppb* is identified from a measurement analysis *ma*, then *ma* should analyze the measure *m*.

SPB-A2: $(\forall ppb \in$ *Process Performance Baseline*, $m \in$ *Measure*, $ma \in$ *Measurement Analysis*$)$ $(isEstablishedTo(ppb,m) \wedge isIdentifiedFrom(ppb, ma) \rightarrow (analyses(ma, m))$

The axiom SPB-A3 says that if a scale value *sv* is a lower limit or an upper limit of a process performance baseline *ppb* established to the measure *m* that has as scale *s*, then *sv* should be a value of the scale *s*.

SPB-A3: $(\forall sv \in$ *Scale Value*, $ppb \in$ *Process Performance Baseline*, $m \in$ *Measure*, $s \in$ *Scale*$)$ $((processPerformanceBaselineLowerLimit(sv, ppb) \vee processPerformanceBaselineUpperLimit(sv, ppb)) \wedge isEstablishedTo(ppb, m) \wedge has(m, s) \rightarrow isPart(sv, s))$

Figure 6 shows the conceptual model that addresses the competency questions CQ7 to CQ14. In this model, it is important to note that *Process Capability* is a relator universal in UFO and that it mediates a quaternary material relation between *Stable Standard Software Process*, *Measure*, *Process Performance Baseline* and *Specified Process Performance*. For best visualization, the model shows only the corresponding mediation relations (*is determined by*, *is established to*, *is calculated in relation to* and *is established in relation to*).

A *Specified Process Performance* is the range that describes the expected results of a standard software process, considering a particular measure. It is a relator universal in UFO, since it is existentially dependent of both *Standard Software Process* and *Measure*. A specified process performance is defined by two limits: *specified process performance upper limit* and *specified process performance lower limit.* As well as baseline limits, the specified process performance limits are part of the *Scale* of the measure used for defining the specified process performance. Returning to our previous example, consider the Requirements Management standard process of the organization Org. It could have a specified process performance defined in relation to the measure "requirements change rate", given by the upper and lower limits 0 and 0,25, respectively.

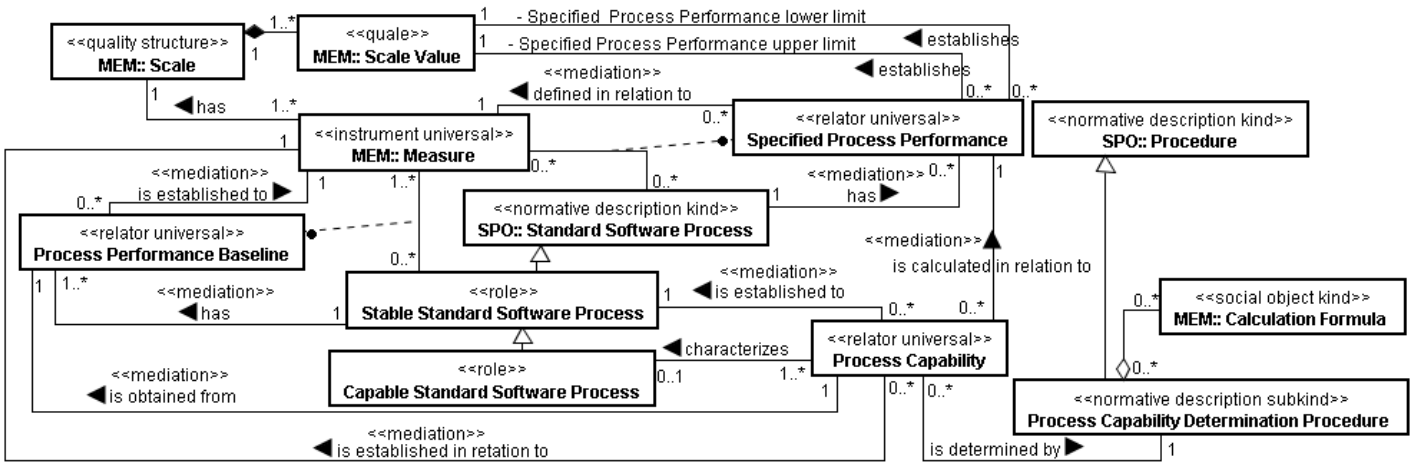

Figure 6. Fragment II of the Software Process Behavior sub-ontology.

Process Capability characterizes the ability of a stable standard software process to achieve the process performance specified for it, considering a particular measure. Process Capability is obtained from a process performance baseline and a specified process performance, and thus, it should be established to the same measure considered by them.

A process capability is determined by applying a *Process Capability Determination Procedure*. This kind procedure defines a logical sequence of operations used to determine the capacity of a standard software process and to identify if it is a capable process. The following is an example of a process capability determination procedure: "Calculate the process capability index using the calculation formula Cp = (ULb – LLb)/(ULs – LLs), where Cp = process capability index, ULb = process performance baseline upper limit, LLb = process performance baseline lower limit, ULs = specified process performance upper limit and LLs = specified process performance lower limit. If Cp is $\leq$ 1, verify if the process performance baseline limits are within the specified process performance limits. In affirmative case, the process is capable. Otherwise, the process is not capable".

When the process capability revels that the process is capable of achieving the expected performance, we have a *Capable Standard Software Process.* Regarding the examples cited before, consider applying the process capability determination procedure to the Requirement Management standard process of the organization Org. As a result, we obtained a capability index 0,6. Besides, the process performance baseline limits are within the specified process performance limits. So, this Requirement Management standard process is a capable standard process with respect to the measure "requirements change rate".

With respect to the conceptual model shown in Figure 6, the following axioms hold. Axioms SPB-A4 and SPB-A5 are consolidation axioms; axiom SPB-A6 is a derivation axiom.

The axiom SPB-A4 says that if a process capability *pc* is established in relation to a measure *m* and it is obtained from a process performance baseline *ppb*, then *ppb* should be established to the measure *m.*

SPB-A4: $(\forall pc \in$ *Process Capability*, $ppb \in$ *Process Performance Baseline*, $m \in$ *Measure*$)$
$(isEstablishedInRelationTo(pc, m) \wedge isObtainedFrom(pc,ppb) \rightarrow isEstablishedTo(ppb, m))$

The axiom SPB-A5 says that if a process capability *pc* is established in relation to a measure *m* and it is calculated in relation to a specified process performance *spp*, then *spp* should be defined in relation to the measure *m.*

SPB-A5: $(\forall pc \in$ *Process Capability*, $spp \in$ *Specified Process Performance*, $m \in$ *Measure*$)$
$(isEstablishedInRelationTo(pc, m) \wedge isCalculatedInRelationTo(pc,spp) \rightarrow isDefinedInRelationTo(spp, m))$

The axiom SPB-A6 says that if a process capability *pc* characterizes as capable the standard software process *ssp* and *pc* is calculated in relation to a specified process performance *spp*, then *ssp* attends *spp*

SPB-A6: $(\forall pc \in$ *Process Capability*, $ssp \in$ *Standard Software Process*, $spp \in$ *Specified Process Performance*$)$
$(characterizes(pc,ssp) \wedge isCalculatedInRelationTo(pc,spp) \rightarrow attends(ssp, spp))$

C. Evaluating the Software Process Behavior Sub-ontology

For evaluating the SMO ontology as a whole, we adopted two strategies. First, we checked if the ontology was able to answer the competency questions posed to it (verification). Aiming a minimum ontological commitment, we also verified if the ontology has only the concepts, axioms and relations needed to answer the competency questions. Second, we validated it with domain experts by using them as basis for defining a strategy to support organizations to obtain and maintain measurement repositories suitable for statistical process control (SPC), as well as to perform measurements appropriately in this context. This strategy is composed of three components [20]: the SMO itself, an Instrument for Evaluating the Suitability of a Measurement Repository to

SPC, and a Body of Recommendations for Software Measurement. The instrument and the body of recommendations have already been evaluated by experts and used in real cases. The preliminary results point out to its usefulness and also to an agreement of the vocabulary used.

Regarding the ontology verification, we checked it manually, since SMO is a reference ontology and it is not implemented in any computational language. Thus, during ontology verification, we related the concepts, relations and axioms of the SMO to the competency questions answered by them, as well as we used individuals (extracted from measure repositories of organizations) to evaluate if the ontology was actually able to represent concrete situations of the real world. Table 1 shows an example of the evaluation of the Process Software Behavior sub-ontology, considering the competency questions CQ2 and CQ10. Table 2 shows one of the instantiations we performed.

TABLE 1. ONTOLOGY EVALUATION.

QC	*Concept A*	*Relation*	*Concept B*	*Axioms*
CQ2	Process Performance Baseline	establishes	Scale Value (process performance baseline lower limit)	SPB-A3
	Process Performance Baseline	establishes	Scale Value (process performance baseline upper limit)	
	Process Performance Baseline	is established to	Measure	
	Measure	has	Scale	
CQ10	Process Capability	is obtained from	Process Performance Baseline	SPB-A4
	Process Capability	is established in relation to	Measure	
	Process Performance Baseline	is established to	Measure	

Concerning CQ2 – *Which are the lower and upper limits of a process performance baseline?* – the relations "*establishes*" are the main responsible for answering this question. However, these relations are constrained by the scale of the measure for which the process performance baseline is established (SPB-A3).

Regarding CQ10 – *From which process performance baseline is a process capability obtained?* – the relation "*is obtained from*" is the main responsible for answering this question. However, this relation should respect axiom SPB-A4 that establishes that the process performance baseline and the process capability should be established in relation to the same measure.

V. RELATED WORKS

Concerning the domain of software measurement, there are some initiatives committed with ontology-based modeling and formalization of this domain. Two of them are the ones described in [14] and [21]. These works are focused on the basic aspects of measurement and are very in line with our Measurable Entities & Measures sub-ontology. A comparison between these proposals and this sub-ontology can be found in [15]. However, these works did not focus on measurement aspects related to high maturity levels and did not address the software process behavior analysis. Furthermore, as a rule, such initiatives are not committed to the use of a foundational ontology as their basis, and, consequently, they rely on models of low expressivity.

TABLE 2. ONTOLOGY INSTANTIATION.

Concept	*Instance*
Standard Software Process	Project Management Process of the organization Org
Stable Standard Software Process	Project Management Process of the organization Org
Measure	Schedule adherence
Process Performance Baseline	PPB-02
Process Performance Baseline Lower Limit	0,85
Process Performance Baseline Upper Limit	0,95
Context of Process Performance Baseline	Second baseline established to the Project Management process. It was established after changes in the management process (were introduced standup meetings, in order to improve the monitoring activities). The data was collected in 4 small projects performed in parallel. The projects teams were homogeneous.
Specified Process Performance	SPP-01
Specified Process Performance Lower Limit	0,80
Specified Process Performance Upper Limit	1,00
Process Capability Determination Procedure	Calculate the process capability index using the calculation formula Cp = (ULb – LLb)/(USs – LLs), where Cp = process capability index, ULb = process performance baseline upper limit, LLb = process performance baseline lower limit, USs = specified process performance upper limit and LLs = specified process performance lower limit. If Cp is ≤ 1, verify if the process performance baseline limits are within the specified process performance limits. In affirmative case, the process is capable. Otherwise, the process is not capable.
Process Capability	PC-01
Capable Standard Software Process	Project Management Process of the organization Org

VI. CONCLUSIONS

In a competitive marketplace, organizations constantly question themselves about the achievement of their business goals. For objectively and accurately answering this question,

they need to know the performance of their processes related to these goals. A measurement program aligned to the business goals supplies the information needed for knowing the processes behavior and, thus, monitoring the business goals achievement. Regarding software organizations, the process behavior analysis is a practice of matured organizations that apply statistical techniques on measurement data.

Nowadays, there are several standards and models that address software measurement. However, the vocabulary used is diverse and some software measurement aspects are not treated, especially aspects related to software measurement at high maturity levels. Aiming to provide a common vocabulary to the software measurement domain, in several maturity levels, we developed a Software Measurement Ontology (SMO). Since we were interested in a reference domain ontology [11], we developed SMO grounded in the Unified Foundational Ontology [12, 13]. This paper presented a SMO sub-ontology: the Software Process Behavior sub-ontology. A part of the Measurement Goals sub-ontology was also presented.

Although several researchers argue in favor of using a foundational ontology as basis for developing domain ontologies [12, 22, 23], few works have explored this use. This is the case of the software measurement domain, in which the proposed ontologies are, in general, lightweight ontologies. We chose UFO because it has been used to evaluate, re-design and integrate (meta) models of conceptual modeling languages, as well as to evaluate, re-design and give real-world semantics to domain ontologies [13, 15, 24].

Currently, the SMO is being used as a conceptual specification for developing and integrating tools and measurement repositories of the High Maturity Environment at LENS (Software Engineering Laboratory) in COPPE/UFRJ. This environment aims to support software organizations to carry out process improvement practices, especially in high maturity levels. Besides, SMO is being used in the development of an approach for defining and monitoring strategically aligned software improvement goals [25], which uses information provided by software process behavior analysis. Finally, we start the use of SMO in the definition of a conceptual architecture to software measurement.

Acknowledgement. This research is funded by the Brazilian Research Funding Agencies FAPES (Process Number 45444080/09) and CNPq (Process Number 481906/2009-6).

REFERENCES

[1] M. E. Porter, Competitive Strategy: Techniques for Analyzing Industries and Competitors, Free Press, 2004.

[2] K. Villela, A. R. Rocha, G. H. Travassos, G.H. et al., “The use of an enterprise ontology to support knowledge management in software development environments”, Journal of the Brazilian Computer Society, Porto Alegre, Brazil, 11(2): 45-59, 2005.

[3] P. Jalote, Software Project Management in Practice, Addison Wesley, 2002.

[4] P. Huang, H. Lei, L. Lim, “Real time business performance monitoring and analysis using metric network”, In Proceedings of IEEE International Conference on e-Business Engineering, pp. 442-449, Shanghai, China, 2006.

[5] W.A. Florac, A.D. Carleton, Measuring the Software Process: Statistical Process Control for Software Process Improvement, Addison-Wesley, Boston USA, 1999.

[6] CMMI Product Team, CMMI for Development Version 1.2, Software Engineering Institute, Pittsburgh USA, 2006.

[7] ISO/IEC 12207 - Systems and Software Engineering - Software Life Cycle Process, 2008.

[8] M. Montoni, A. R. Rocha and K. C. Weber, "MPS.BR: a successful program for software process improvement in Brazil", Software Process Improvement and Practice, 14, 289-300, 2009.

[9] ISO/IEC 15939, Systems and Software Engineering – Measurement Process, 2007.

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

[11] G. Guizzardi, On Ontology, ontologies, Conceptualizations, Modeling Languages and (Meta)Models, In O. Vasilecas, J. Edler, A. Caplinskas (Org.). Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV. IOS Press, Amsterdam, 2007.

[12] G. Guizzardi, Ontological Foundations for Structural Conceptual Models Universal Press, The Netherlands, 2005.

[13] G. Guizzardi, R.A Falbo and R.S.S. Guizzardi, "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, 244-251, 2008.

[14] M.F. Bertoa, A. Vallecillo, F. García, "An ontology for software measurement", In: C. Calero, F. Ruiz, M. Piatini (Eds.), Ontologies for Software Engineeri ing and Software Technology, Springer-Verlag Berlin Heidelberg, 2006.

[15] M. P. Barcellos, R. A. Falbo, R. Dalmoro, "A well-founded software measurement ontology", In Proceedings of the 6th International Conference on Formal Ontology in Information Systems (FOIS 2010), Toronto - Canadá, 2010.

[16] R.A. Falbo, C.S. Menezes and A.R.C. Rocha, "A systematic approach for building ontologies", In Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, LNCS, vol. 1484, 1998.

[17] M.P. Barcellos, R.A. Falbo, "Using a foundational ontology for reengineering a software enterprise ontology", In Proceedings of the Joint International Workshop on Metamodels, Ontologies, Semantic Technologies, and Information Systems for the Semantic Web, 2009.

[18] IEEE, 1998, *Std 1061 – IEEE Standard for a Software Quality Metrics Methodology.*

[19] Putnam, L., 1978, "A general empirical solution to the macro software sizing and estimation problem", IEEE Transactions on Software Engeneering, pp. 345-361, 1978.

[20] M. P. Barcellos, A.R.C. Rocha, R.A. Falbo, "An ontology-based approach for software measurement and suitability measures basis evaluation to apply statistical software process control in high maturity organizations", In Proceedings of the ER2009 PhD Colloquium, 2009.

[21] M.A.Martin, L. Olsina, "Towards an ontology for software metrics and indicators as the foundation for a cataloging web system", First Latin American Web Congress, 2003.

[22] N. Guarino,"Formal ontology and information systems", In Proceedings of International Conference in Formal Ontology and Information Systems, pp 3-15, 1998.

[23] J.M. Fielding, J. Simon, W. Ceusters, B. Smith, "Ontological theory for ontology engineering", In Proceedings of the Ninth International Conference on the Principles of Knowledge Representation and Reasoning, Whistler, Canada, 2004.

[24] 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.

[25] Barreto, A.O.S., Rocha, A.R., “Defining and monitoring strategically aligned software improvement goals”, In Proceedings of the 11th International Conference on Product Focused Software Process Improvement, Ireland, 2010, in press.