Towards a Measurement Ontology Pattern Language

Monalessa P. BARCELLOS, Ricardo A. FALBO and Vinícius G. V. FRAUCHES

Ontology and Conceptual Modeling Research Group (NEMO), Computer Science Department,
Federal University of Espírito Santo, Vitória, Brazil
{monalessa, falbo, vgvfrauches}@inf.ufes.br

Abstract. Measurement is an important activity in several domains. Although there are specific concepts regarding measurement in each domain, some concepts are common to all of them. This paper presents a Measurement Ontology Pattern Language (M-OPL) that addresses the measurement core conceptualization. M-OPL can be used for building measurement ontologies to several domains. The ontology patterns that compose M-OPL are developed taking the Unified Foundational Ontology (UFO) as a basis. As an example, we present the use of M-OPL for building a software measurement domain ontology.

Keywords. Measurement, Ontology, Core Ontology, Ontology Pattern, Ontology Pattern Language, Software Measurement Ontology.

1 Introduction

Measurement is a very important discipline in many domains, since it provides useful information for getting conclusions and making decisions. Depending on the application domain, knowledge related to measurement presents some particularities. However, regardless the domain, some measurement aspects remain unchanged.

There are several standards that address measurement in specific domains (e.g., [1, 2]), but there are also some proposals addressing measurement general aspects. VIM [3], for instance, defines a vocabulary regarding measurement as an attempt to standardize terminology across different domains. When analyzing different standards, it is possible to identify some core concepts. On the other hand, the terminology used is diverse. Many times, the same concept is designated by different terms in different standards; others, the same term refers to different concepts.

Ontologies have been recognized as a useful instrument for reducing conceptual ambiguities and inconsistencies, and for making knowledge structures clearer. In this sense, ontologies can be used for promoting common understanding among knowledge workers [4]. The core conceptualization about measurement should be shared across many domains, thus it can be represented as a core ontology. A core ontology provides a precise definition of the structural knowledge in a specific field that spans across several application domains in that field [5]. Moreover, a core ontology should be extensible to incorporate particularities of specific domains, so that ontology engineers can reuse and extend the core ontology when building a domain ontology.

The use of Ontology Pattern Languages (OPL) to organize core ontologies favors reusing and extending the core knowledge [6]. An OPL [6] is a network of interconnected ontology modeling patterns that provides holistic support for solving ontology

development problems for a given field. An OPL offers a set of interrelated patterns, plus a process guiding on what problems can arise in that field, informing the order to address these problems, and suggesting one or more patterns for solving them.

In this paper, we present the first version of the Measurement OPL (M-OPL), which addresses the core conceptualization about measurement. This version comprises patterns covering six measurement aspects: Measurement Entities, which includes patterns related to the entities and their properties that can be measured; Measures, which deals with defining measures and classifying them according to their dependence on others measures; Measurement Units & Scales, which concerns the scales related to measures and the measurement units used to partition scales; Measurement Procedures, dealing with the procedures needed to collect data for measures; Measurement Planning, which addresses the goals that drive measurement as well as the measures used to verify goals achievement; and Measurement & Analysis, which concerns data collection and analysis. M-OPL patterns are defined based on the Unified Foundational Ontology (UFO) [7].

This paper is organized as follows. Section 2 presents the background of the paper, including a brief description of measurement, aspects related to core ontologies and ontology patterns languages, and UFO's fragments relevant to this work; Section 3 presents M-OPL; Section 4 illustrates the use of M-OPL for developing a software measurement ontology; Section 5 discusses related work, and Section 6 presents our final considerations.

2 Background

Measurement can be defined as a set of actions aiming to characterize an entity by attributing values to its properties. The main activities involved in measurement are: planning, execution and analysis. During planning, the entities to be measured are identified (e.g., objects, phenomena, and so on), as well as the properties to be measured (e.g., size, cost, etc.), the measures to be used to quantify those properties (e.g., size in meters), and how measurement of each measure should be carried out (e.g., the area of a square object must be measured by applying the formula $a=s^2$, where s is the side length). Decisions regarding which entities and properties are to be measured and which measures are to be used should be driven by measurement goals. Once measurement is planned, it can be performed. Measurement execution consists on collecting data for the measures by applying measurement procedures. Finally, measurement analysis comprises analyzing collected data, aiming to get and communicate conclusions regarding the measured entity [1, 3].

There are several important concepts that are general to measurement despite any specific domain in which it may occur. This core conceptualization can be captured by a core ontology. Core ontologies are conceived mainly aiming at reuse, and thus, a pattern-oriented design approach is appropriate for organizing them [5]. An ontology pattern describes a particular recurring modeling problem that arises in specific ontology development contexts and presents a well-proven solution for the problem [8]. By following a pattern-oriented design approach, core ontologies become modular and extensible [5]. In fact, core ontologies are amenable to be represented as Ontology Pattern Languages (OPLs) [6]. An OPL is more than a catalogue of patterns. It includes, besides the patterns themselves, a process suggesting an order to apply them.

Finally, when designing a core ontology, it is desirable to use a solid modeling basis given by a foundational ontology. Concepts and relations defined in a core ontology should be aligned to the basic categories of a foundational ontology [5]. By doing that, core ontologies incorporate a solid, semantically precise basis. In this paper, we used the Unified Foundational Ontology (UFO) [7]. UFO has been developed based on theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics and Cognitive Psychology. It is composed by three main parts: UFO-A, which is an ontology of endurants; UFO-B, which is an ontology of perdurants (events); and UFO-C, an ontology of social entities, built on the top of UFO-A and UFO-B. A complete description of UFO falls outside the scope of this paper. Thus, in the sequel we describe some concepts (based mainly on [7, 9, 10]) that are important for this paper. Figure 1 shows a fragment containing concepts from UFO-A, B and C. The concepts shown in grey are the ones directly used in this paper.

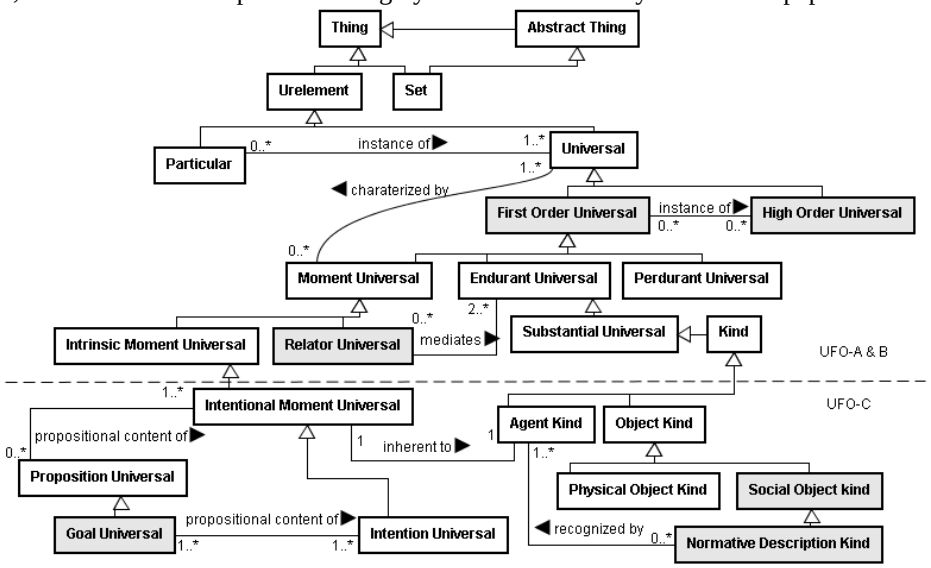

Fig. 1. UFO fragment.

The root concept of UFO is *Thing*, which is specialized into *Urelement* and *Set*. Sets are abstract entities (*Abstract Thing)* that simply exist, not being explicitly created or destroyed. Urelements are all elements that are not sets. Concerning Urelements, a fundamental distinction in UFO is between the categories of *Particular* and *Universal*. Universals are patterns of features that can be realized in a number of different entities (e.g., Person). Particulars, in turn, are entities that exist in reality, possessing a unique identity (e.g., the person Mary). The model depicted in Figure 1 focus on universals. Since the concepts defined from Particular are analogous to the ones specialized from Universal, they are not showed in Fig. 1.

Universals can be *First Order Universals*, i.e., universals whose instances are particulars (e.g., Person), or *High Order Universals*, which are universals whose instances are also universals (e.g., Mammal, whose instances could be Person, Dog, etc.). First Order Universals can be classified in *Endurant Universals*, *Perdurant Universals* and *Moment Universals.* Endurants persist in time maintaining their identity (e.g., a person). Perdurants (events) unfold in time with their multiple temporal parts

(e.g., a process). Moment Universals (properties) are universals that characterize others. For instance, the moment universal Color characterizes the universal Apple. Moments are existentially dependent of another entity, 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 Moment Universals* are dependent on one single entity (e.g., Color). *Relator Universals*, in turn, depend on a plurality of entities (e.g., Employment) and, for this reason, provide the material connection between them.

Substantial Universals comprehend the entities that do not need another entity in order to exist. While persisting in time, substantials can instantiate several types of substantial universals. *Kind* is the type of substantial that instantiates in every possible situation and defines what the substantial is (e.g., Person). An important distinction in UFO is between agents and objects. 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 Kind* (e.g., Book) and *Social Object Kind* (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 agent (e.g., a method describing how to perform some activity within an organization). *Agent Kind* is a substantial universal whose instances are capable of performing actions with some intention (e.g., Person and Organization).

Intentional Moment Universal is a special type of intrinsic moment universal whose instances are inherent to agents and has a propositional content (*Proposition Universal*). Intentional moment universals in which the intentionality is "intending something" are said *Intention Universal*. An intention characterizes a situation desired by the agent (e.g., an organization can have the intention 'to be successful'). The propositional content of an intention is said a *goal* (e.g., 'to be among the ten top companies in the world' could be the propositional content of the intention 'to be successful').

Figure 2 shows a fragment of UFO-A addressing concepts related to qualities. *Quality Universals* refer to the properties that characterize Universals (e.g., Weight and Height). They are Intrinsic Moment Universals associated to *Quality Structures*, which can be understood as the set of all possible regions that delimits the space of values possible to be associated to a particular *Quality Universal.* For instance, the quality universal Weight is associated to a space of values that is a linear structure isomorphic to the positive half-line of the real numbers. The regions that compose a Quality Structure are called *Quality Regions*, and are regions that approximate qualia. A *Quale* is a perception of a quality in a quality structure. *Function* is a specialization of Set that maps instances of a Quality Universal to points in a Quality Structure.

According to the quality structures to which they are associated, *Quality Universals* are classified into *Simple* and *Composed Quality Universals*. The first one is associated to one-dimensional quality structures (e.g., Weight) and the last one is associated to multi-dimensional quality structures (e.g., Body Mass Index). An important distinction regarding quality universals is related to their nature. *Measurable Quality Universals* are quality universals that can be objectively measured by cognitive agents or measurement devices, and it is possible to establish distances among their quality regions (e.g., Weight and Height). Differently, *Nominal Quality Universals*, such as Name and Zip Code, are usually based on social conventions and cannot

be objectively measured. In this paper we are interested in *Measurable Quality Universals*.

Measurement Quality Structures are structures that allow for objectively evaluating the distance between two values and verifying if the values are equal or not. They are classified, according to the number of dimensions, into *Measurement Quality Dimension* and *Measurement Quality Domain.* The first one represents the most elementary (one-dimensional) measurement quality structures, and the last one represents n-dimensional quality structures. *Measurement Quality Domains* can be *Cognitive Measurement Quality Domain* or *Scientific Measurement Quality Domain.* The practical difference between them is that regions from scientific domains can be qualitatively evaluated and ordered, while regions from cognitive domains cannot. Scientific domains are composed following some kind of algebra and have an *Expression* that determines their formation. For example, the scientific domain for the Body Mass Indicator (BDI) is formed using the dimensions Weight and Height (BMI = Weight/ (Height x Height)).

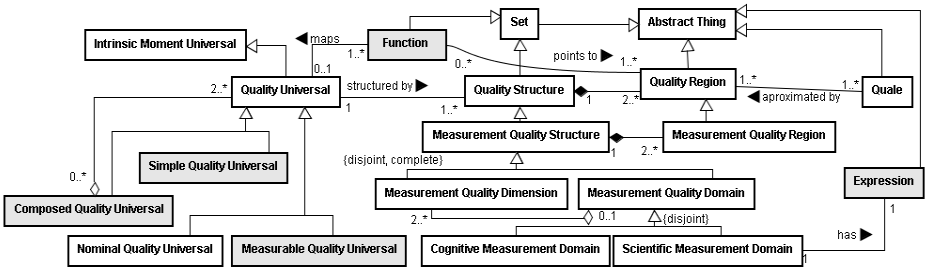

Fig. 2. UFO-A fragment related to Qualities.

Figure 3 shows an UFO-A fragment addressing concepts related to *Reference Structures.* As said before, *Quality Regions* are regions that approximate qualia. A quale is intrinsic to cognitive agents and therefore cannot be shared or communicated. In order to allow quale communication, it is necessary to use *Lexical Elements* (e.g., 1,86 can be the lexical element used to communicate the height of a person) associated to *Reference Regions* and *Reference Structures.* A *Reference Region* is an abstract entity based on a *Quality Region* that acts as a bridge between that region and the lexical elements used to communicate the approximated quale. A *Reference Structure*, in turn, is associated to a *Quality Structure* and is a set of *Reference Regions* grounded in *Quality Regions* of that *Quality Structure*. When the 'value' of a particular quality is being referred by lexical elements (e.g., 1,86), what is actually being referred is a quality region that most approximates the quale.

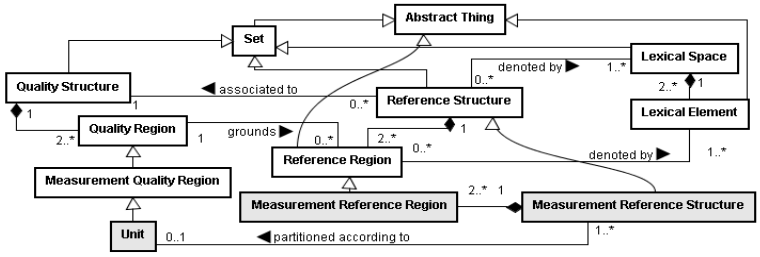

Fig. 3. UFO-A fragment related to Reference Structures.

Reference Structures are topologically isomorphic to the *Quality Structures* to which they are associated. Thus, they have the same number of dimensions and their *Reference Regions* are isomorphic to the *Quality Regions* of the *Quality Structure*. *Reference Structures* associated to *Measurement Quality Structures* are called *Measurement Reference Structures* and act like scales grounded by quality structures. They are composed by *Measurement Reference Regions*. *Measurement Reference Structures* can be partitioned in spaces with the same magnitude according to a *Unit*.

3 M-OPL: A Measurement Ontology Pattern Language

In this section, we present the first version of M-OPL, which comprises a set of ontology patterns plus a process describing how to combine them when building a domain ontology (in fact, an ontology about measurement in a specific domain). Patterns in M-OPL are organized in six groups, namely: Measurable Entities, Measures, Measurement Units & Scales, Measurement Procedures, Measurement Planning and Measurement & Analysis.

Measurable Entities group deals with modeling problems related to identifying the entities and elements to be measure (**MEnt** pattern) and types of measurable elements (**TMElem** pattern). Figure 4 shows the patterns in this group. In this figure we also show UFO's concepts (in white) that ground the patterns.

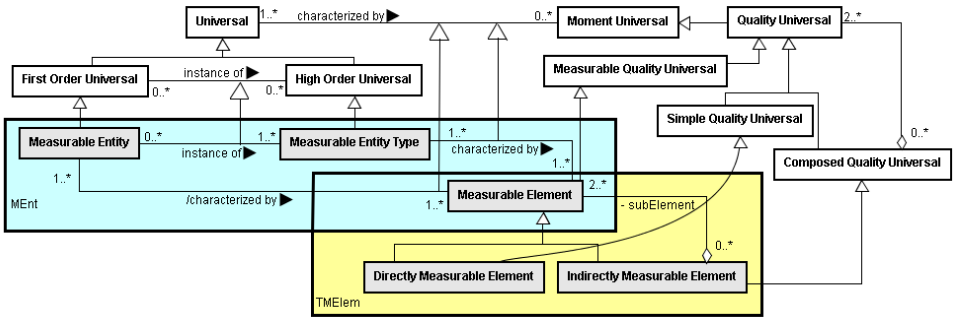

Fig. 4. Measurable Entities patterns.

Measurable Entity is a concept whose instances represent anything that can be measured, such as a person, a project, an organization. Measurable entities are classified according to **Measurable Entity Types**, which is a *High Order Universal* in UFO. For example, John (an instance of Measurable Entity) is an instance of Person (an instance of Measurable Entity Type). Measurable Entity Types are characterized by **Measurable Elements** (e.g., entities of the type Person can be characterized by the measurable elements weight, height and BMI). Measurable Elements can be **Directly Measurable Elements** (elements that do not depend on others to be measured, such as weight and height) or **Indirectly Measurable Elements** (which depend on **subelements** in order to be measured, e.g., BMI). Measurable Entity corresponds in UFO to *First Order Universals* that are characterized by *Measurable Quality Universals*. Notice that the relation 'characterized by' between Measurable Entity and Measurable Element is specialized from the homonym relation between Universal and Moment Universal, however the minimum cardinality is 1 stead of 0, making explicit that we

are talking about the entities that can be measured. Measurable Elements are *Measurable Quality Universals* (specialization of *Quality Universal*) in UFO. Directly and Indirectly Measurable Elements are *Simple* and *Composed Quality Universals*, respectively. In UFO, *Quality Universals* are associated to *Quality Structures* that can be one or n-dimensional. Since Directly and Indirectly Measurable Elements are also specializations of Measurable Element, they correspond to Measurable Quality Universals associated respectively to one and n-dimensional quality structures.

Figure 5 shows the patterns of the ***Measures*** group, which concerns modeling problems related to identify measures that quantify measurable elements (**Mea** pattern) and to classify measures into types (**TMea** pattern).

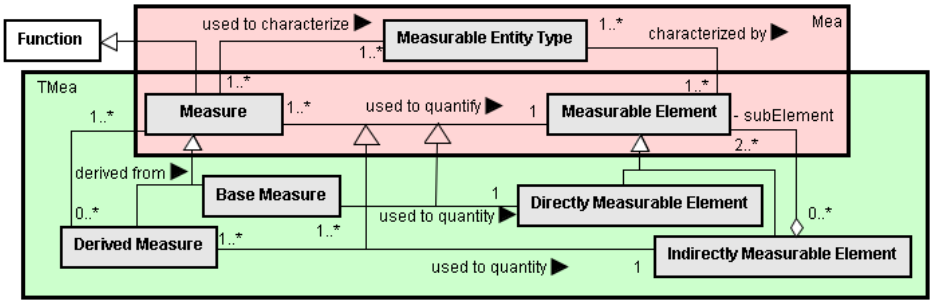

Fig. 5. Measures patterns.

Measures are used for quantifying Measurable Elements and characterize Measurable Entity Types. For instance, the measure number of lines of code (LOC) can be used to quantify the measurable element size of measurable entities of the type Software Program. Measure is a *Function* in the sense that it maps an instance of Measurable Element to a value (the measured value, which is explained forward). Measures are categorized into two types: **Base Measure** and **Derived Measure**, which are used to quantity Directly and Indirectly Measurable Elements, respectively. Number of LOC is an example of base measure and defect density (number of defects/number of LOC) is an example of derived measure.

Figure 6 shows the patterns from the ***Measurement Scales & Units*** group, which deals with problems related to scales for measures (**MScale** pattern), types of scales (**TMScale** pattern), units in which measures are expressed (**MUnit** pattern), and the relation between measurement units and scales (**MUnit&Scale** pattern).

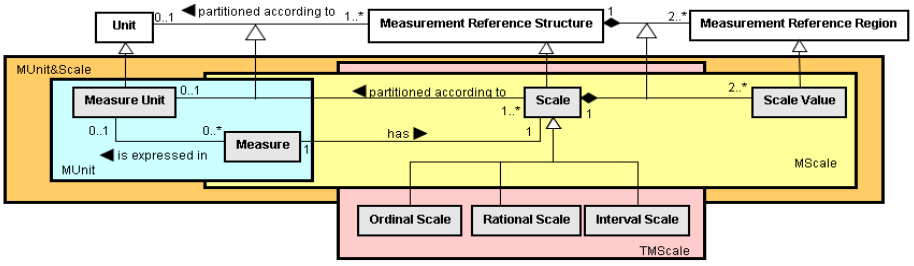

Fig. 6. Patterns from the Measurement Scales & Units group.

Measures have **Scales** composed by all possible values (**Scale Value**) to be associated by the measure to a measurable element. According to UFO, a *Quality Univer-*

sal is structured by *Quality Structures* that delimit the space of values possible to be associated to the *Quality Universal. Reference Structures* are associated to *Quality Structures* and are composed of *Reference Regions* that act as a bridge to the lexical element that communicate a *Quale*. Thus, taking the fact that we are talking about *Measurable Qualities Universals* into account, Scale is subtype of *Measurement Reference Structure* and Scale Value is subtype of *Measurement Reference Region*.

Measures can be expressed in **Measure Units** (e.g., meter, kilo). A measure unit in which a measure is expressed partitions its scale. For instance, if the measure height is expressed in meters, it means that its scale (a liner structure isomorphic to the positive half-line of the real numbers) is partitioned in meters.

Figure 7 shows one of the patterns (**MProc**) from the ***Measurement Procedures*** group, which addresses **Measurement Procedures**, i.e., procedures applicable to Measures and that describe the steps to be carried out aiming to collect data for these measures. A measurement procedure is a *Normative Description* in UFO.

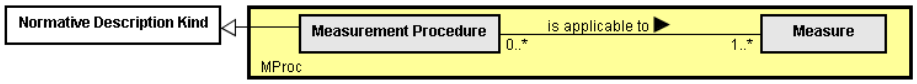

Fig. 7. MProc - Measurement Procedure pattern.

Figure 8 shows the patterns of the ***Measurement Planning*** group. The **TMGoal** pattern deals with measurement goal decomposition. The **INeed** pattern addresses information needs identified from measurement goals. The **MPI** pattern concerns planning the measurement by specifying, for each measurement goal or information need, which measure should be collected. The **MPI-MP** pattern regards selecting the measurement procedure to be used. Finally, the **Ind** pattern concerns which measure plays the role of an indicator to indicate the achievement of some measurement goals.

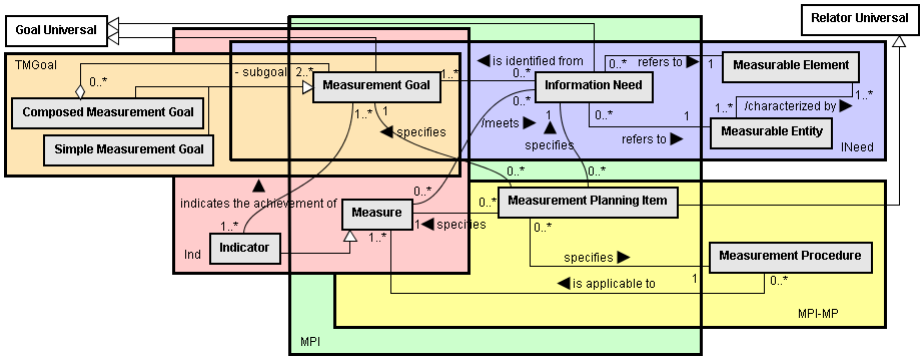

Fig. 8. Measurement Planning related patterns.

Measurement Goals are goals that can be used to drive the identification of the measures needed in a certain context. For instance, check the patient health can be a measurement goal of a doctor. Measurement goals are classified into **Simple Measurement Goal** and **Composed Measurement Goal**. The last is composed by other goals, said its **sub-goals**. For instance, check the patient health is a Composed Measurement Goal, and check if the patient has a good BMI can be one of its sub-goals. In UFO, an *Intention* is something that an agent intentions. For instance, a doctor can

have the intention of taking a good care of his patient. The propositional content of an *Intention* is a *Goal*. Thus, Measurement Goal is a *Goal Universal*.

From Measurement Goals it is possible to identify **Information Needs** that determines which information have to be met by measures in order to monitor the measurement goal. For instance, the information need know the patient BMI can be identified from the measurement goal check if the patient has a good BMI. Information Need refers to a Measurable Element and to a Measurable Entity. For instance, know the patient BMI refers to the measurable element BMI and to a patient (e.g., John), which is the measurable entity. Measures meet Information Needs taking the Measurable Element and the Measurable Entity into account (e.g., the measure BMI in kilos/meters^2 can be used to meet the information need know the patient BMI). Measures directly used to indicate the achievement of Measurement Goals are called **Indicators**. In the example, BMI plays the role of indicator, since it is used to indicate the achievement of the measurement goal check if the patient has a good BMI.

Finally, a **Measurement Planning Item** connects a Measurement Goal, an Information Need, a Measure, and a Measurement Procedure, meaning that the Measure meets the Information Need that was identified from the Measurement Goal. Since Measurement Planning Item connects several entities, it is a *Relator Universal.*

Figure 9 shows the Measurement pattern (**Mea**) from the ***Measurement & Analysis*** group, which deals with problems related to data collection and analysis. **Measurement** is performed based on a Measurement Planning Item. It measures a Measurable Element of a Measurable Entity by applying a Measure and adopting a Measurement Procedure. The result is a **Measured Value.** A measured value refers to a value of the measure scale. Thus, Measured Value can be seen as the role played by a Scale Value when it is associated to a particular measurable element in a particular measurement. **Measurement** is a *Relator Universal*, since it connects the entities involved in a measurement. For instance, the measurement of the John's weight by applying the measure weight in kilos and adopting a certain measurement procedure could determine the measured value 80 (in fact, according to UFO a quale cannot be communicated, thus the symbol 80 is the lexical element used to communicate the measured value).

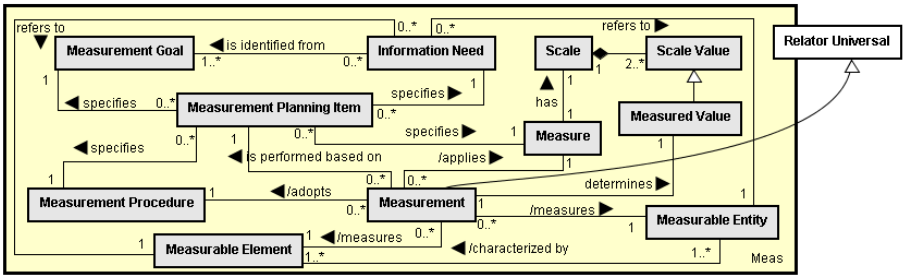

Fig. 9. Meas - Measurement pattern.

During the patterns development, we identified several constraints that cannot be captured by the models. In order to address them, we defined axioms. For instance, concerning the model depicted in Fig. 9, there is, among others, a constraint regarding Measured Value and Scale: if a measurement *mea* applies a measure *m* that has the scale *sc* and determines the measured value *mv*, then *mv* should be a value of the scale

sc: *(∀ mea ∈ Measurement, m ∈ Measure, sc ∈ Scale, vl ∈ Measured Value) (applies(mea,m) ∧ has(m,sc) ∧ determines(mea,mv) → isPartOf (mv,sc)).* Due to space limitations, we do not discuss other axioms in this paper.

As aforementioned, an OPL includes, besides the patterns, a process suggesting an order to apply them. Figure 10 presents the M-OPL process, which is represented by means of a UML activity diagram, as suggested in [6]. In the figure, patterns are represented by action nodes (the labeled rounded rectangles). Initial nodes (solid circles) are used to represent entry points in the OPL, i.e., patterns in the language that can be used without using other patterns first. Control flows (arrowed lines) represent the admissible sequences in which patterns can be used. Decision nodes (diamond-shaped symbols) represent alternative paths in the OPL. Fork nodes (line segments with multiple output flows) are used to represent independent and possibly parallel paths. Joint nodes (line segments with multiple input flows) are used to represent independent paths that must be used in order to proceed to the next flow. Patterns in grey and darker lines are the patterns and paths used in the example discussed in Section 4.

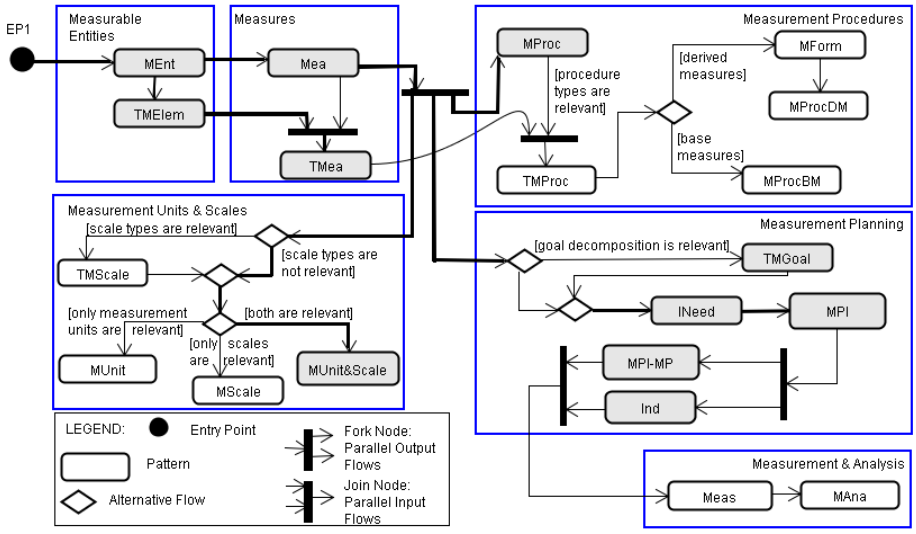

Fig. 10. Measurement Ontology Pattern Language - M-OPL.

M-OPL has one entry point (EP1 in Fig. 10). The first pattern to be used is **MEnt** (Measurable Entities). After using this pattern, two patterns are applicable: **TMElem** (Types of Measurable Elements) and **Mea** (Measures). If **TMElem** is used, then **TMea** (Types of Measures) has also to be used. Note that, for using **TMea**, both **TMElem** and **Mea** should be used first.

From **Mea**, there are three possible parallel paths to follow. The first one goes to the *Measurement Units & Scales* group. In this group, if types of scales (e.g., interval, ordinal and rational) are relevant, the ontology engineer must use **TMScale** (Types of Scale). If she needs to model measurement units and scales, the pattern **MUnit&Scale** (Measurement Unit and Scale) should be used; if only scales are relevant to the scope of the ontology, **MScale** (Measurement Scale) is to be used; if only measurement units are relevant, **MUnit** (Measurement Unit) should be used.

The second path from **Mea** goes to the *Measurement Procedures* group. **MProc** (Measurement Procedure) allows modeling the procedures that guide data collection for measures. If the ontology engineer needs to address different types of measurement procedures, depending on the type of measures, she must use **TMProc** (Types of Measurement Procedure). In this case, besides **Mea**, **TMea** should also be previously used. Measurement procedures for base measures and derived measures are addressed respectively by **MProcBM** and **MProcDM**. To deal with measurement procedures for derived measures, the engineer must first use the **MForm** pattern, which addresses measurement formulas used to calculate derived measures.

The third path from **Mea** goes to the Measurement *Planning* group. **TMGoal** (Types of Measurement Goal) should be used only if measurement goal decomposition is relevant. **INeed** (Information Need) must be used to address information needs. **MPI** (Measurement Planning Item) must be used to deal with planning measurement items. **MPI-MP** (Measurement Planning Item – Measurement Procedure) should be used to indicate which measurement procedure must be applied when collecting data for the measure specified in the measurement planning item. **Ind** (Indicator) allows the engineer to model the relationship between measurement goals and the measures used to indicate their achievement. Once addressed measurement planning, it is possible to address issues related to Measurement (**Mea**) & Analysis (**MAna**).

4 Using M-OPL for Building a Software Measurement Ontology

In this section, we discuss how M-OPL was used for developing a Software Measurement Ontology (SMO). SMO aims to capture the conceptualization regarding measurement in software engineering. This involves defining the types of software entities typically measured in this domain, measures used for this purpose, goals that drive software measurement, among others. In order to develop SMO, we followed the paths indicated by the darker lines in Fig. 10, and thus we used the patterns shown in grey in that figure. Figure 11 shows a fragment of the resulting ontology.

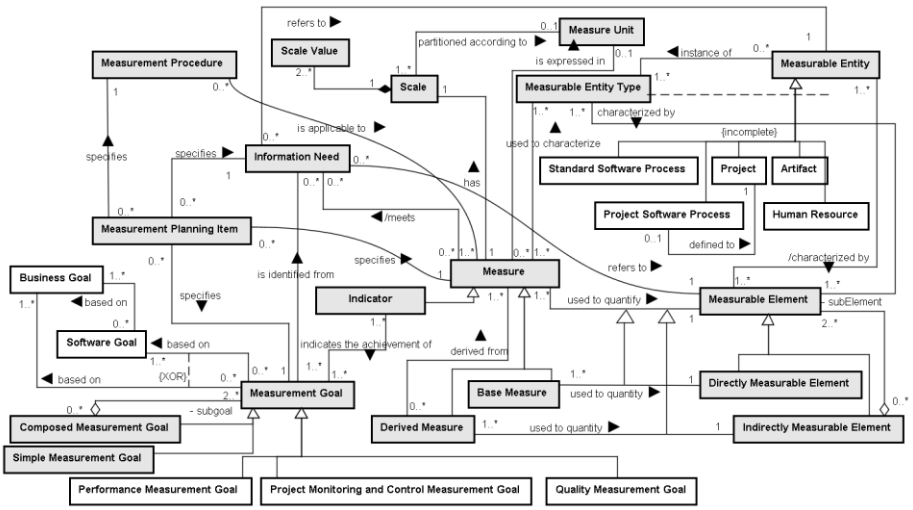

Fig. 11. Fragment of a Software Measurement Ontology developed from M-OPL.

First, we used the **MEnt** pattern, which has been extended to consider the types of measurable entities relevant to the software domain. Some of them are: **Standard Software Process**, which refers to a generic process defined by an organization, establishing basic requirements for processes to be performed in that organization [11]; **Project Software Process**, which refers to the processes defined for specific projects, considering their particularities [11]; **Project**, which refers to a set of commitments established by the involved parties in order to achieve established goals; **Artifact**, which refers to results that can be produced or used when activities are performed [11]; and **Human Resource**, which refers to the general role played by a person in an organization. It is important to notice that this generalization set is incomplete, meaning that there are other types beyond the ones presented in Fig. 11.

The patterns **TMElem**, **Mea**, **TMea**, **MUnit&Scale** and **MProc** were then used exactly as they are defined in M-OPL, i.e., none extension was necessary. In order to address software measurement specific needs, it was necessary to extend the pattern **MGoal**. Types of measurement goals relevant to the software domain were identified, namely: **Project Monitoring and Control Measurement Goal** (e. g., improve the adherence to projects plans), **Quality Measurement Goal**, (e. g., improve the tests coverage) and **Performance Measurement Goal** (e.g., improve the capability of critical processes). As the names suggest, these goals express intentions for which actions related to project management, software quality and behavior process analysis are planned and performed. Besides, two new concepts were included and related to Measurement Goal, namely: **Business Goal**, which expresses the intention for which strategic actions are planned and performed (e.g., increase the clients level of satisfaction), and **Software Goal**, which expresses the intention for which actions related to the software area are planned and performed. In the software measurement domain, Measurement Goals are established based on **Business Goals** or **Software Goals**, and Software Goals are established based on Business Goal. Reduce the number of defects in the delivery software products could be a Software Goal established based on the cited business goal. Verify tests efficiency, in turn, could be a Measurement Goal established based on that software goal. Finally, **INeed**, **MPI-withMP** and **Ind** patterns were applied without any further extension. Aspects related to Measurement & Analysis were not addressed by this version of SMO.

5 Related Work

There are several works addressing measurement concepts. The work described in this paper focus on representing the core conceptualization about measurement as an Ontology Pattern Language (OPL), in order to favor reuse. The use of OPLs is a recent initiative, thus there are only few works proposing OPLs. In [6] and [12], we proposed OPLs to the Software Process and Enterprise domains respectively. However, at the best of our knowledge, there is no similar initiative in the measurement field. Thus, there is not a work to compare with ours in respect to the organization of the core conceptualization about measurement in ontology patterns.

Regarding the conceptualization underlying M-OPL, as said before, we focus on capturing the core conceptualization of measurement. Thus, it is worthwhile to point out that we are not proposing a foundational ontology; contrariwise, we are grounded in the foundations provided by UFO. In this sense, we do not consider related works

the ones addressing measurement theories or measurement aspects in the foundational level, such as [10, 13]. We also do not consider related work domain ontologies approaching measurement, since this kind of works focus on specific domains. We consider related work only core ontologies of measurement, such as the one proposed by Kim et al. [14]. In [14], Kim et al. present the TOVE Measurement Ontology (TMO), a measurement core ontology for Semantic Web applications. TMO addresses concepts related to the following aspects: (i) measurement system, which deals with attributes that can be measured, samples, and quality requirements; (ii) measurement activities, which deals with data collection, inspection and test; and (iii) measurement points, addressing measured values and their conformance to the quality requirements. There is some equivalence between TMO and M-OPL concepts (e.g., measure, measure unit and measurable element), although different terms are used in some cases. However, TMO does not address some aspects covered by M-OPL, such as measurement goal, indicator, scale, measurement procedure, measurement formula, and measurement analysis, among others. Besides, TMO does not explore some relations between concepts. For instance, in TMO there is no relation between measure and measured attribute (equivalent to Measurable Element in M-OPL). Finally, TMO is defined by means of propositions founded upon a first-order language and, different from our proposal, there is no mechanism to facilitate reuse.

6 Final Considerations

Currently, reuse is recognized as an important practice for Ontology Engineering. Ontology patterns are considered a promising approach that favors reuse of encoded experiences and good practices in Ontology Engineering [15]. Moreover, core ontologies organized as OPLs have potential to amplify the benefits of ontology patterns [6]. Agreeing with these statements, we developed the first version of the Measurement OPL (M-OPL), which addresses core measurement aspects. As a proof of concept of the utility of M-OPL, we developed a software measurement ontology. As a result, we noticed that the use of a pattern language such as M-OPL tends to bring some benefits to the development of domain ontologies: (i) the resulting ontology tends to contain less inconsistency mistakes given that many of the potentially recurring source of inconsistencies in the domain tend to be solved by the basic patterns of the core ontology; (ii) the development process of the derived domain-specific measurement ontologies tends to be accelerated by the massive reuse of modeling fragments and decisions embedded in the patterns of the language; and (iii) M-OPL guides pattern selection, also facilitating combining them. Although we have perceived these benefits, real case experiments have to be conducted to truly confirm them.

It is worthy commenting that aiming to facilitate the patterns reuse, in the full M-OPL documentation, in addition to the conceptual models, other information regarding each pattern are available to the ontology engineer, such as: purpose, rationale, competence questions that the pattern aims to answer, model description, axiomatization, foundations, among others.

As future work, we intend to evolve M-OPL by adding new patterns related to aspects not addressed by the current version (for instance, aspects related to measurement data quality) and explore new applications of M-OPL aiming to get relevant feedback for improvements.

Acknowledges

This research is funded by the Brazilian Research Funding Agency CNPq (Process Number 485368/2013-7).

References

1. ISO/IEC:ISO/IEC15939(E) Software Engineering - Software Measurement Process. (2007)
2. ASTM: ASTM D6956 Standard Guide for Demonstrating and Assessing Whether a Chemical Analytical Measurement System Provides Analytical Results Consistent with Their Intended Use. USA (2011)
3. JCGM: VIM - International Vocabulary of Metrology – Basic and General Concepts and Associated Terms. Joint Committee for Guides in Metrology, France (2012)
4. Uschold, M., Jasper, R.: A Framework for Understanding and Classifying Ontology Applications. In Proceedings of IJCAI Workshop on Ontologies and Problem-Solving Methods, Stockholm, Sweden 1-11 (1999)
5. Scherp, A., Saathoff, C., Franz, T., Staab, S.: Designing core ontologies. Applied Ontology 6, 177-221 (2011)
6. Falbo, R.A., Barcellos, M.P., Nardi, J.C., Guizzardi, G.: Organizing Ontology Design Patterns as Ontology Pattern Languages. Proceedings of the 10th Extended Semantic Web Conference - ESWC Montpellier, France (2013)
7. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Universal Press, The Netherlands (2005)
8. Falbo, R.A., Guizzardi, G., Gangemi, A., Presutti, V.: Ontology Patterns: Clarifying Concepts and Terminology. Proceedings of the 4th Workshop on Ontology and Semantic Web Patterns, Sidney, Australia (2013)
9. Guizzardi, G., Falbo, R.A., Guizzardi, R.S.S.: Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. In: Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments, Recife - Brasil (2008)
10. Albuquerque, A., Guizzard, G.: An Ontological Foundation for Conceptual Modeling Datatypes Based on Semantic Reference Spaces. 7th International Conference on Research Challenges in Information Science (RCIS), Paris, France (2013)
11. Briguente, A.C.O., Falbo, R.A., Guizzardi, G.: Using a Foundational Ontology for Reengineering a Software Process Ontology. In: Proceedings of the XXVI Brazilian Symposium on Data Base (2011)
12. Falbo, R.A., Ruy, F., Guizzard, G., Barcellos, M.P., Almeida, J.P.: Towards an Enterprise Ontology Pattern Language. 29th ACM Symposium On Applied Computing (2014)
13. Probst, F.: Observations, Measurements and Semantic Reference Spaces. Journal of Applied Ontology 3, 63-89 (2008)
14. Kim, H.M., Sengupta, A., Fox, M.S., Dalkilic, M.: A Measurement Ontology Generalizable for Emerging Domain Applications on the Semantic Web. Journal of Database Management 18, 20-42 (2007)
15. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: eXtreme Design with Content Ontology Design Patterns. Proceedings of the First Workshop on Ontology Patterns Washington D.C., EUA (2009)