UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN
FACULTAD DE MINAS

DYNA
http://dyna.medellin.unal.edu.co/

# OntoAgile: an ontology for agile software development processes

Wilson Alfredo Ortega-Ordoñez [a], César Jesús Pardo-Calvache [a] & Francisco José Pino-Correa [b]

[a] *GTI Research Group, Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca, Popayán, Colombia. wortega@unicauca.edu.co, cpardo@unicauca.edu.co*
[b] *IDIS Research Group, Facultad de Ingeniería Electrónica y Telecomunicaciones, Universidad del Cauca, Popayán, Colombia. fjpino@unicauca.edu.co*

Received: December 6th, 2018. Received in revised form: January 20th, 2019. Accepted: March 12th, 2019.

**Abstract**
Currently, there is a broad portfolio of agile approaches to software development, however, in many cases their implementation is done informally and without a proper institutionalization of the agile values and principles. Although there are some proposals related to the adoption and assessment of agile approaches, efforts have been made without having a common terminology which has led to confusion and terminological conflict affecting the implementation of these approaches in organizations. This article proposes the ontology called OntoAgile, which aims to suggest a common and consistent terminology that allows sharing the knowledge generated around the implementation of the agile approaches in the software processes in a generic and formal way. Similarly, OntoAgile facilitates the assessment of the agility of the software processes from the identification of the relationships between the elements of the software processes and the agile principles and values. OntoAgile was assessed satisfactorily by three cases of application.

*Keywords*: ontology; agile; software processes; agility assessment; agile software development.

# OntoAgile: una ontología para procesos de desarrollo ágil de software

**Resumen**
Actualmente, existe un amplio portafolio de enfoques ágiles para el desarrollo de software, sin embargo, en muchos casos su adopción se realiza de manera informal y sin una correcta institucionalización de los valores y principios ágiles. Aunque existen algunas propuestas relacionadas con la adopción y evaluación de enfoques ágiles, los esfuerzos se han realizado sin contar con una terminología común lo que ha conllevado confusión y conflicto terminológico afectando la implantación de estos enfoques en las organizaciones. Este artículo propone la ontología denominada OntoAgile, cuyo objetivo es sugerir una terminología común y coherente que permita compartir el conocimiento generado en torno a la implementación de los enfoques ágiles en los procesos de software de manera genérica y formal. Asimismo, OntoAgile facilita la evaluación de la agilidad de los procesos software a partir de la identificación de las relaciones entre los elementos de los procesos software y los principios y valores ágiles. OntoAgile fue evaluada de manera satisfactoria mediante tres casos de aplicación.

*Palabras clave*: ontología; ágil; procesos software; evaluación de agilidad; desarrollo ágil de software.

## 1. Introduction

The frameworks and agile solutions (from now on agile approaches or AAs) for software development, are based mainly on the values and principles defined in the agile manifesto [1], the AAs have been designed to offer organizations software benefits that allow them to overcome problems related to the constant change of requirements and products that do not satisfy the costumer [2]. Currently, software organizations have a plethora of AAs that can be used to support the management and development of their software projects, among which are: Scrum [3], Extreme Programming (XP) [4], Crystal Clear [5], Lean Software Development (LSD) [6], Adaptive Software Development (ASD) [7], Dynamic Systems Development Method (DSDM) [8], Feature-Driven Development (FDD) [9], Agile Unified Process (AgileUP) [10], Kanban [11], among others. Also, a set of agile solutions known as agile scaled

**How to cite:** Ortega-Ordoñez, W.A., Pardo-Calvache, C.J. and Pino-Correa, F.J., OntoAgile: an ontology for agile software development processes. DYNA, 86(209), pp. 79-90, April - June, 2019.

approaches are currently highlighted, which aim to support the distributed development of software, among them are: SAFe [12], LeSS [13], Nexus [14], among others. The success of the AAs lies in providing companies mainly: (i) improvement in productivity, (ii) alignment between customers and development teams, (iii) anticipated results (time to market), (iv) flexibility and adaptation to the changes, (v) light and iterative processes, among others [15,16].

The wide range of AAs along with the traditional approaches (hereinafter TAs) allow software organizations to have different frames of reference with which they can carry out the preparation of processes, which involves the creation, modification or adaptation of the description of processes for a particular purpose within an organization [17]. This broad spectrum of frameworks allows organizations to choose between one or several referents to support specific needs of the organization, the use of various frameworks can be carried out through their harmonization [18], resulting in the creation of hybrid solutions. Some experiences suggest that the effectiveness of the implementation of an AA or TA may depend to a large extent on the careful adjustment made in the defined process [19,20], However, it is important to highlight that designing processes, tuning them and making sure that they are also agile is particularly difficult, because the definition of agile processes lacks support for many of the techniques defined for the analysis, modeling and redesign of processes; methods that have been used mainly for more structured processes based on traditional approaches [21].

On the other hand, based on the analysis of the state of the art performed, it has been possible to observe that in relation to the study and adoption of the TAs, ontologies have mainly been developed to represent the integration of the elements of specific domains, for example: ISO 9001 and the Capability Maturity Model Integration for Development (CMMI-DEV) [22], CMMI-DEV and ISO/IEC 15504 [23,24], It is even possible to find some efforts carried out for the development of specific domain solutions such as the Capability Maturity Model for Software (CMMI-DEV) [25], the Software Engineering Body of Knowledge (SWEBOK) [26], among others. However, these ontologies have been defined mainly to represent structures of quality approaches based on traditional processes, therefore, the concept of agility as a principle, value or approach, is not present. Similarly, for the EAs, ontologies and taxonomies have been defined for specific domains in XP [20], FDD [27], Scrum [28,29], and the generic ontology for agile methods presented in [30]. At the analysis level of the structure of the concepts related to the agility, some defined taxonomies have been found, which have allowed to identify the dependencies between agile software development projects and to facilitate the coordination and collaboration among projects [31] and also, solutions that have allowed analyzing the agile behavior of project managers [32].

However, it has been possible to observe that the proposed solutions are of a preliminary nature and are limited in the following aspects: i) a generic scope that supports multiple AAs and TAs is not included, ii) absence of concepts and fundamental agile relationships such as: values , principles, agile practices and activities of the process, iii) the relationship of the proposed concepts, the concept of quality, the agile principles and values are not observed, iv) there are still inconsistencies and conflicts related to terminology, and vii) the potential and benefits of the harmonization between the concepts and relationships present in AAs and TAs is not maximized.

Considering the above, this article aims to present a solution to this gap identified in the context of AAs through the definition of OntoAgile, which is an ontology that describes in a generic way the concepts and common relationships that must be present in the processes made with any EA, in this way, the ontology can be used independently of the agile approach that is being adopted. In this sense, OntoAgile organizes the knowledge related to the implementation of agile approaches in software processes of organizations in a generic and formal way, reducing the ambiguity of concepts and terms in that domain, and allowing the knowledge to be reused and shared by those interested in the area. Similarly, OntoAgile promotes and facilitates the evaluation of the compliance of the agility of the software processes through the identification of the relationships among the software process elements, the agile principles and the measurement of the software. On the other hand, OntoAgile has been extended and adapted to take into account the characteristics of the software processes of each organization, as well as to facilitate their adoption with software development TAs, this characteristic has been achieved through the integration of concepts of other ontological solutions as well as Ontology of Process-reference Models (PrMO) [33], which clarifies the main elements to express process-based approaches, and the sub-ontology Software Measurement Ontology (SMO) [34], which clarifies the key elements in the definition of a software measure and the terminology related to the action of measuring the software.

In addition to the current introduction, this article is organized as follows: Section 2 presents an analysis of the current status of the related works. Next, Section 3 introduces OntoAgile, an ontology that provides a coherent set of concepts and their relationships within a terminology that aims to serve as a reference for the adoption of multiple models, standards and approaches, in addition to the assessment of the agility of software processes. Section 4 describes three cases of application of the ontology and the assessment of its capacity to respond to the defined competency questions. Finally, conclusions and future work are presented in Section 5.

## 2. Background

### *2.1. An analysis of the current situation*

In our search in the literature, it has been possible to identify few efforts to develop formal ontologies to support the assessment of the agility of software processes. The work carried out in this sense has focused mainly on the development of ontologies to represent the key elements of particular domains, e.g. ontologies for traditional and agile approaches. Likewise, it has been possible to find some works which proposes some taxonomies. Below is the

literature found that follows these aspects:

- Ontologies for traditional approaches - OTAs. In the search of related literature, it is important to mention that a great effort has been focused on the definition of ontologies for software processes related to TAs, some of which can be highlighted are related to: CMMI-SW [25], ontology based on the body of knowledge proposed in SWEBOK [35], ontologies to support the implementation of CMMI and ISO/IEC 15504 as well as: Software Process Ontology (SPO), which allows expressing software processes at the conceptual level and provides an extension to generate ontologies for specific process models [36] [23] and the ontology for unified basic concepts for process capability models [24], amongst others. However, these ontologies have been defined mainly to represent structures of quality approaches based on processes, in this sense, it has been possible to show that some concepts, such as: principle, value, approach, amongst others, are not present.
- Ontologies for agile approaches - OAAs. Regarding the AAs it has been possible to identify some efforts presented in some works that develop ontologies aiming at supporting the analysis of the concepts and relationships around the processes related to this type of approaches, some of them are described below: (i) ontology XPO (eXtreme Programming Ontology) [20], which models three main concepts: organizational role, product and phase, XPO also allows indexing relevant documents, XP artifacts and Wiki pages, this, in order to extract and analyze agile processes, the activity of the programmers and the content of the repositories, (ii) Ontology based on the life cycle of feature-driven development (FDD or Feature Driven Development) [27], this ontology can be used in the development of application models and in the design and implementation of features in semantic web applications, (iii) Ontologies based on Scrum, such as: (a) the K-CRIO ontology [28], which is a conceptualization of Scrum that allows the description of the business processes that are devoted to the design of a product based on the key concepts and the relationships of the software development process, besides, it illustrates the use of the ontology taking as example the Scrum development process and (b) a proposed ontology that allows to generate evaluations of the Scrum process in a software development organization, the evaluations generated seek to help software engineers understand the process they are following to develop software [29]. Finally, (iv) an ontology to support the development of agile software (OSDAS), which took seven agile methods such as: Agile Microsoft Solutions Framework (AMSF), Agile UP, Crystal Clear, Dynamic Systems Development Method (DSDM), eXtreme Programming (XP) and Scrum, with which the authors tried to summarize the varied terminology in a generic ontology, with this ontology the authors sought to cover the diverse characteristics of the commonly used agile approaches [30].
- Taxonomies. It has also been possible to find some taxonomies, the first one, is a dependency taxonomy which allows us to identify the dependencies for agile software development projects, this work studies one of the most important features and pillars in agile projects, the collaboration, and how to get it through an effective coordination by means of the analysis of dependence between projects [31]. Similarly, it was possible to find a study which analyses the behavior of project managers when they adopt agile approaches in software development projects [32].

After analyzing the related works, some aspects associated to the limitations of these proposals are presented below:

- Most proposed solutions are of a preliminary nature and are limited in their application to support a specific approach, i.e: the proposals are focused to be used in specific domains such as TAs or AAs and not in an integrated manner or hybrid. It is important to note that currently the agile movement has taken on great importance and has begun to be integrated with other legacy approaches, the vast majority being traditional, as well as: ISO standards as well as, ISO/IEC 15504, ISO 9001, including the new ISO 29110 standard, or de facto models such as CMMI, among others.
- While some proposals represent the concepts and relationships of some AAs, it is possible to observe the absence of concepts and relationships of important terms that determine the true essence and quality of the AAs, some of these concepts are: values, principles, thinking and agile practices.
- It is still possible to observe inconsistencies and conflicts related to terminology, this may be due to the fact that the proposed solutions focus on proposing conceptual solutions about the terms of specific domains, which have very particular characteristics that make it impossible to support the descriptions of multiple approaches.
- It has not been possible to observe an ontological solution that encourages and/or facilitates the evaluation of the agility of the software processes conceived from AAs. Although solutions have been defined to address the evaluation of certain AAs as well as Scrum, these are not generic and also do not incorporate concepts such as values and agile principles, fundamental aspect to consider to carry out the evaluation of the agility of software processes of an organization regardless of the approach that is being used as a reference model.

From the results obtained in the analysis of the related works, it has been possible to identify that different proposals have been developed to support different specific domains from the conceptualization of the terms/concepts and relationships that describe them. However, in the related works different terms/concepts are used to identify their elements, this is normal, because each proposal has been defined by different authors or research groups.

Therefore, it is possible to observe that an effort is required to resolve some differences on terminological conflicts in the environment of the definition of an agile solution that allows to support the evaluation of agility and in turn, it serves as a bridge to conceptually support the concepts between AAs and TAs.

Likewise, it is possible to note that a homogeneous approach that facilitates the reconciliation/harmonization of the differences between multiple approaches is required,

which allows identifying and generating a common and consistent terminology to enhance or extend the integrated use of multiple approaches from AAs and TAs.

### *2.2. Conceptual and theoretical background*

#### 2.2.1. REFSENO: a representation formalism for software engineering ontologies

There are several methodologies that allow systematizing the implementation of ontologies, for example: knowledge management based on ontologies [37], Methontology [38], a translation approach to portable ontology specifications [39], ontology and first order logic [40], representation of ontologies for software engineering known as REFSENO [41], amongst others. After studying the different methodologies that could be used for the definition of ontologies, it was decided to perform the specification and conceptualization of OntoAgile using the formalism defined in REFSENO since: (i) is based on an adaptation of Methontology, which is widely used to define ontologies in different contexts, (ii) REFSENO has been specially designed as a specialization of Methontology for the development of ontologies in software engineering through constructors that allow define concepts, attributes and relationships among them, (iii) other methodologies only allow less intuitive and complex representations for people who are not familiar with the logic of predicates of the first order or similar, REFSENO provides several techniques that allow supporting the analysis of the consistency of the ontology and the instances in the level of implementation. Knowledge can be represented in REFSENO on two levels of abstraction: knowledge levels and symbol level. The symbol level is defined by means of implementation [41] while the knowledge levels are independent of the implementation [41]. In short, a knowledge level describes "what" to represent, while the symbol level describes "how" to represent it. According to [42], REFSENO proposes three knowledge levels: the epistemological level, the conceptual level and the linguistic level. Epistemological level describes the epistemistic primitives such as concepts, attributes, relations amongst others. Conceptual level describes the standard vocabulary and linguistic level describes concrete instances of the constructs defined on the above level. The suggested stages for the construction of an ontology from REFSENO in the Conceptual level are: i) Construction, ii) Evolution and iii) Validation. In addition to these stages, for the definition of OntoAgile was also followed the Description Logic - DL for being more expressive than the propositional logic [43].

#### 2.2.2. Ontology integration

In the context of ontologies, it is possible that the word integration has several meanings [44], the overuse of this word makes it difficult to organize the different defined ontologies that result from an integration. It may seem incredible and even funny, but even in the area of knowledge related to the design and creation of ontologies it is possible to notice that there are certain ambiguities. Different authors use the term integration in different ways, some use different terms to refer to the same or vice versa, they use the same term, but with different meanings [45]. The term integration can be defined through different meanings [45]: (i) matching: related to the process of finding relationships or correspondences between different ontology entities, (ii) alignment: related to the result obtained after identifying the set of correspondences between two or more ontologies, (iii) mapping: the oriented or directed version of an alignment , this maps the entities of an ontology into at least one entity of another ontology, (iv) merging: the creation of a new ontology from two ontologies of origin, possibly superimposed, and (v) integration: the inclusion of an ontology in another ontology together with the assertions that relate them.

## 3. OntoAgile: ontology for agile software development processes

Taking into account the analysis of the current state of art, the following objectives were established for the definition of OntoAgile: (i) location and identification of synonyms, homonyms, inconsistencies and terminological conflicts and (ii) integration of the concepts found in the analyzed literature. These objectives are achieved with OntoAgile, which is a common ontology that represents the domain of agile approaches providing concepts, terms with clear and concise definitions, and their respective relationships, and can also be used as a basis to support the requirements of assessment and evaluation of the agility of an organization, as well as harmonization or reconciliation with other approaches that need to be implemented in an integrated manner. According to the classification proposed by [46], in the software engineering area, ontologies can be grouped into two categories: i) domain ontologies, those that describe the knowledge of a particular domain and ii) ontologies as software artifacts, which serve as artifacts during the process of software development. According to the above, OntoAgile is a domain ontology that allows organizing knowledge related to the concepts and relationships around the agile software development. Next, a general approximation of the ontology is presented in terms of its purpose, the glossary of concepts and its graphic representation.

Table 1.
OntoAgile's competency questions (CQ).

| Id | Competency Questions |
|---|---|
| CQ1 | What is the relationship between values and agile principles? |
| CQ2 | What types of software development approaches are guided by agile values? |
| CQ3 | What are the main elements that make up a software process? |
| CQ4 | What is the relationship between the activities of the process and the agile principles? |
| CQ5 | What is the relationship between the resources used in the process and the agile principles? |
| CQ6 | What is the relationship between the products generated in the process and the agile principles? |

Source: The Authors.

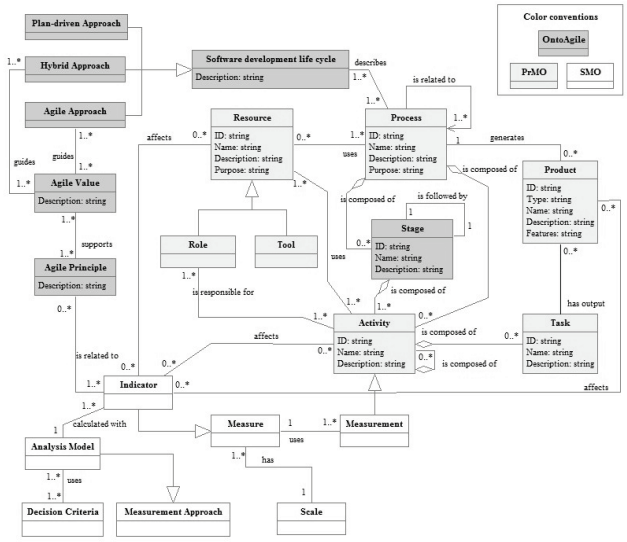


Figure 1. OntoAgile UML representation.
Source: The Authors.

### *3.1. Purpose*

Allow those interested in the agile software development to know the concepts and relationships that should be present in the agile development processes, regardless of the particular approach that is being adopted. OntoAgile can be used from the following perspectives:

- Academic, because those interested in learning or researching about this topic can use ontology to know the terms and relationships used in agile software development.
- Industrial, since software organizations can use ontology to instantiate their software processes and to know to what extent the concepts and basic relationships that occur in the agile contexts of software development are present. Likewise, it can be used as a basis to support the assessment and evaluation of the agility of an organization, as well as a conceptual tool to facilitate the harmonization of multiple models that must be implemented in an integrated manner for the definition of hybrid process solutions.

  OntoAgile allows to answer to the competency questions (CQ) presented in the Table 1.

### *3.2. Integrating OntoAgile with other ontologies*

Considering that OntoAgile supports the assessment of the agility of the software processes of organizations, it was necessary to include concepts related to software processes and software measurement. In this sense, an OntoAgile integration process was carried out with the ontology: Ontology of Process-reference Models (PrMO) [33], which establishes and clarifies the main elements to express process-based approaches, specifically in traditional approaches, such as: SWEBOK, CMMI, ISO 9001, ITIL, ISO 27001, ISO / IEC 15504, among others. Also, some concepts of the Software Measurement Ontology (SMO) sub-ontology presented in [34]. SMO establishes and clarifies the key elements in the definition of a software measure and the terminology related to the action of measuring the software. Fig. 1 presents the graphical representation of OntoAgile, its terms and relationships with the PrMO and SMO ontologies using the UML (Unified Modeling Language) representation (available at: http://bit.ly/owl-ontoagile).

### *3.3. Concepts of the OntoAgile*

The accurate definitions of the concepts included in the ontology are presented in Table 2, which is arranged alphabetically and is organized as follows: the first column shows the term that is described, the second column shows the super concept, then the third column shows the definition of the term in OntoAgile. The fourth column shows the source where the term has been adopted or adapted. Table 3 presents the relationships between the concepts presented in Table 2, the description of the attributes of terminal concepts has been omitted. The descriptions presented in the fourth column of Table 2 can be of the following types:

Taken from [source], if the term has been defined in another work and is key to the definition of OntoAgile. In addition, the term has not been changed or adapted. Defined from [source], if the term has been defined from a source that does not provide a particular definition, that is, the term has been defined without highlighting, changing or complementing an existing term, but the work presented in it has been key to establishing a definition. Cited in [source], if the term has been cited by a source and it is not the original source. The term has not been modified.

### *3.4. Discussion of some concepts and their relationships*

Some of the terms used in OntoAgile deserve a special discussion. As such, in this section we expand the definition and analysis of some terms such as: Software development life cycle, agile value and indicator.

A software development life cycle (SDLC) describes an approach to design, build and maintain software. There is a wide variety of SDLCs such as Waterfall, spiral, incremental, rational unified process (RUP), agile software development, among others. However, in general terms, all SDLCs consist of a sequence of steps that software developers must follow to obtain and deliver a software product [50]. SDLCs may be classified into plan-driven approaches, hybrid approaches and agile approaches. The agile values were defined in the agile manifest to summarize the values that should be present in the agile software development teams. The four core values of Agile software development described in the agile manifest are: i) individuals and interactions over processes and tools, ii) working software over comprehensive documentation, iii) customer collaboration over contract negotiation, and iv) responding to change over following a plan. The four values are supported by twelve agile principles which are described and can be consulted in [1].

An indicator is a measure derived from one or more measures that allows knowing the degree of compliance with an objective. In the case of agility assessment, indicators related to each of the agile principles can be defined and, in this way, know the degree of agility of the software development process based on the analysis of compliance with each of the agile principles. Indicators may be affected by process elements (products, roles and activities).

Table 2.
Glossary of concepts in the Ontoagile.

| Concept | Super-concept | Definition | Source |
|---|---|---|---|
| Activity | Concept | Comprises a set of tasks or actions used to produce and maintain devices as well as to achieve the objectives of the process. The activity includes the procedures, standards, policies, and objectives to create and modify a set of work products. | Taken from [33] |
| Agile approach | Software development life cycle | Software development approach based on iterative development, frequent inspection and adaptation and incremental deliveries, in which requirements and solutions evolve through collaboration in cross-functional teams and through continuous feedback from the participants. | Defined from: [47] |
| Agile principle | Concept | Guideline that is considered desirable in the agile development and that guides the actions of the development team. Each agile principle is derived from one or several agile values. | Defined from [1] |
| Agile value | Concept | Value that must be present in the members of an agile development team. The agile manifest defines four agile values. | Defined from [1] |
| Analysis model | Measurement approach | Algorithm or calculation combining one or more measures with associated decision criteria. | Taken from [34] |
| Decision criteria | Concept | Thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result. | Taken from [34] |
| Hybrid approach | Software development life cycle | A hybrid approach is any combination of agile and traditional (plan-driven or rich) approaches that an organizational unit adopts and customizes to its own context needs. | Taken from: [48] |
| Indicator | Measure | A measure that is derived from other measures using an analysis model as measurement approach. | Taken from [34] |
| Measure | Concept | The defined measurement approach and the measurement scale. | Taken from [34] |
| Measurement | Concept | A set of operations having the object of determining a value of a measurement result, for a given attribute of an entity, using a measurement approach. | Taken from [34] |
| Measurement approach | Concept | Sequence of operations aimed at determining the value of a measurement result. | Taken from [34] |
| Plan-Drive approach | Software development life cycle | Software development approach that has the following characteristics: The requirements are specified at the beginning of the project; a detailed plan is defined from the beginning to the end of the project; the requirements are specified in great detail and then a rigorous change request process is implemented; the architecture and design specification must be complete before the implementation begins; the programming work is only focused on the programming phase; the tests are carried out at the end of the project; the quality guarantee is handled in a formal way. | Cited in [49] |
| Process | Concept | Coherent set of policies, organizational structures, technologies; procedures, purposes, objectives, and work products that are needed to design, develop, deploy and maintain a software product. | Taken from [33] |
| Product | Concept | The set of artifacts to be developed, delivered and maintained in a project is called the product. The products can be of input or output type; mandatory or optional. Products are in most cases tangible artifacts consumed, produced, or modified by tasks. | Taken from [33] |
| Resource | Concept | A resource is an asset a business needs to have. In the field of software engineering, there are two main resources of importance: the developers and the tools. | Taken from [33] |
| Role | Concept | Describes a set or group of responsibilities, duties and skills required to perform a specific activity. | Taken from [33] |
| Scale | Concept | A set of values with defined properties. | Taken from [34] |
| Software development life cycle | Concept | A software development life cycle refers to the framework that is used to plan, manage, and control the process of developing an information system. | Cited in: [50] |
| Stage | Concept | Period within the life cycle of an entity that relates to the state of its description or realization. Stages relate to major progress and achievement milestones of the entity through its life cycle. | Taken from [47] |
| Task | Concept | Process element that defines the work done by roles. A task is associated with the input and the output products. | Taken from [33] |
| Tool | Resource | The tools automate the execution of certain activities. | Taken from [33] |
| Unit of measurement | Concept | Particular quantity, defined and adopted by convention, with which other quantities of the same kind are compared in order to express their magnitude relative to that quantity. | Taken from [34] |

Source: The Authors.

Table 3.
Relationships in the Ontoagile.

| Name | Concepts | Descriptions |
|---|---|---|
| describes | Software development life cycle - Process | A Software development life cycle describes one or more processes. A process is described by one or more software development life cycle. |
| guides | Agile value - Agile approach | An agile value guides one or more agile approaches. An agile approach is guided by one or more agile values. |
| guides | Agile value - Hybrid approach | An agile value guides one or more hybrid approaches. A hybrid approach is guided by one or more agile values. |
| supports | Agile principle - Agile value | An agile principle supports one or more agile values. An agile value is supported by one or more agile principles. |
| is composed of | Process - Stage | A process is composed of stages. A stage is part of a process. |
| is followed by | Stage - Stage | A stage may be followed by another stage. |
| is composed of | Stage - Activity | A stage is composed of one or more activities. An activity is part of a stage. |
| is composed of | Process - Activity | A process is composed of one or more activities. An activity is part of a process. |
| is related to | Process - Process | A process may be related to another process. |
| generates | Process - Product | A process generates or does not generate many products. A product is generated by a process. |
| uses | Process - Resource | A process uses or does not use many resources. A resource is used by one or more processes. |
| uses | Activity - Resource | An activity uses one or more resources. A resource is used by one or more activities. |
| has output | Task - Product | A task has as output one or many products. A product is an output of a task. |
| is composed of | Activity - Task | An activity is composed or not of many tasks. A task is part of one activity. |
| is composed of | Activity - Activity | An activity is composed or not of one or more activities. |
| is responsible for | Role - Activity | A role is responsible for one or more activities. An activity has at least one responsible. |
| affects | Product - Indicator | A product affects or not one or more indicators. An indicator may be affected by one or more products. |
| affects | Resource - Indicator | A resource affects or not one or more indicators. An indicator may be affected by one or more resources. |
| affects | Activity - Indicator | An activity affects or not one or more indicators. An indicator may be affected by one or more activities. |
| is related to | Indicator - Agile principle | An indicator is related or not to one or more principles. A principle is related or not to one or more indicators. |
| calculated with | Indicator - Analysis model | Every indicator is calculated with one analysis model. Every analysis model may define one or more indicators. |
| uses | Analysis model - Decision criteria | An analysis model uses one or more decision criteria. Every decision criteria is used in one or more analysis models. |
| uses | Measurement - Measure | Every measurement uses one measure. One measure may be used in several measurements. |
| has | Measure - Scale | Every measure has a scale. A scale may serve to define more than one measure. |

Source: The Authors.

## 4. Evaluation of the OntoAgile

According to [51], the ontology assessment must be focused mainly on the user evaluation of three elements related to the performance and correct definition: consistency, completeness, and conciseness, i.e. the consistency assess if it is possible to infer knowledge from the ontology, completeness can only be assessed by proving the nonexistence of incompleteness, by determining if the individual definitions are well established and if all that is supposed to be stated in the ontology is present or can be inferred. In addition, if the proposed ontology is not redundant and with useless definitions, then the ontology can be considered as concise.

For the assessment of the OntoAgile some interviews with agile practitioners and software engineering specialists were performed. The participants evaluated the present definitions in a consensual manner as complete, consistent and concise for the scope of assessment of the agility of software processes. As result of interviews, some suggestions emerged and they were considered. Likewise, it is important to highlight that practitioners and software engineering specialists stated that OntoAgile can be a great help resource that provides a clearer and more complete vision about the evaluation of the agility of the software processes. The ontology was reviewed and updated by means of an incremental and iterative process which included: the concepts attributes and their relations established.

The ontology summarized and presented in this document was implemented in the OWL ontology language [52] using Protégé [53]. With OntoAgile implemented in OWL (available at: http://bit.ly/owl-ontoagile), the Hermit reasoner [54] was used to evaluate the consistency of the ontology structure. Once the analysis was completed, it was possible to verify that OntoAgile does not present inconsistencies in its structure. Subsequently, OntoAgile has been used successfully in three implementation cases. The first case of implementation refers to the instantiation of the OntoAgile with the main elements of the agile Scrum approach. In the second case, the ontology was used to support the formulation of questions related to the evaluation of agility based on agile principles. The third case of implementation corresponds to the development of a Web application that makes queries to OntoAgile using the Protocol and RDF Query Language (SPARQL) to visualize the relationship between values and agile principles. In addition, the capacity of the OntoAgile to answer competency questions using SPARQL was evaluated. Finally, the contributions of the proposed ontology to the academic community and the industry were analyzed through their comparison with

existing ontologies for agile approaches.

Due to space limitations, the detailed results of the OntoAgile evaluation were published in a public repository (available at: http://bit.ly/OntoAgileEvaluation). The repository includes the results of the evaluation by experts through a focus group and the source code of the web application.

### *4.1. Supporting the instantiation of Scrum*

The first case of implementation involved the participation of a group of researchers interested in the Agile Scrum Framework and the formalization of its main elements such as roles, meetings, artifacts, among others. For the instantiation of the elements that make up Scrum, the following concepts defined in OntoAgile were used: Process, Stage, Activity, Task, Tool, Role, and Product. Fig. 2 shows an extract of the instance created from OntoAgile, where it can be observed that a software process guided by Scrum is made up of the stages: Initiate, Sprints and Realease. In turn, the stage of Sprints is composed of the stages: Plan Estimate, Implement and Review Retrospect. The stages are made up of activities, as in the case of the Create User Stories activity that is part of the Plan Estimate stage. In this activity, the Customer Interviews tool is used in order to know the customer's needs to carry out the Write user stories and Write Acceptance Criteria tasks. As a result of the Write User Stories task, you get the User Stories product, which is the Product Owner's role responsibility.

Although only an extract of the instantiation of Scrum is shown, it can be seen that OntoAgile provides the relationships and concepts to represent the main elements of the agile approach.

### *4.2. Supporting the instantiation of software process Agility Reference Model (AgilityRM)*

The second case of implementation is related to the evaluation of agility, which takes as a basis the agile principles for the definition of a set of agility indicators of software processes. For the instantiation of the elements that make up AgilityRM, several concepts defined in OntoAgile were used, for example: Activity, Agile principle, Agile value, Indicator and Product.

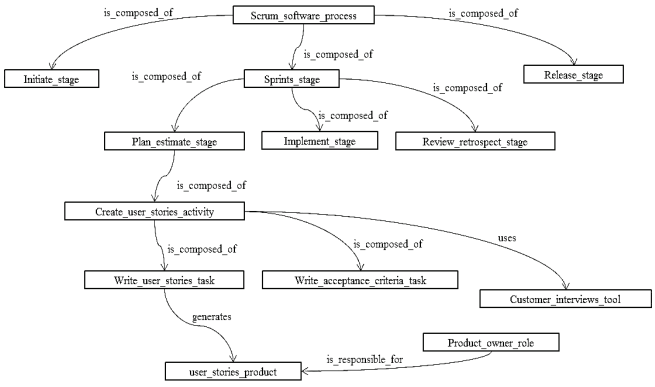


Figure 2. Instantiation excerpt of Scrum.
Source: The Authors.

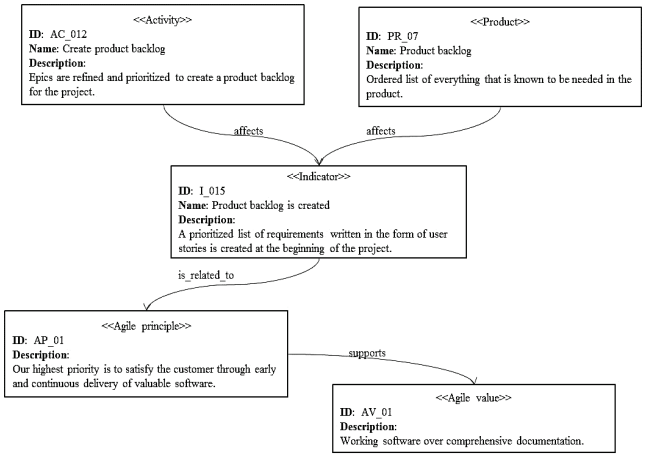


Figure 3. Excerpt from the OntoAgile instantiation to support the definition of the AgilityRM model.
Source: The Authors.

Fig. 3 shows an example of the formulation of an indicator generated from the analysis of the way in which an agile principle can be fulfilled and in turn provide support to one of the agile values. In this case, to verify compliance with the indicator a "Product backlog is created", it must be verified that in the software process an activity aiming to create a product backlog has been defined and that a product backlog has been generated as a product. AgilityRM is not yet finished, however, it has shown that the development of an assessment framework based on an ontology allows defining a coherent and solid structure.

It is important to highlight that thanks to the generality of OntoAgile, it is possible to extend its application to contexts that allow carrying out the assessment of the processes of the organizations in relation to their agility, this offers to the organizations that adopt agile approaches the possibility of defining software processes that include activities, roles and products that are aligned with agile principles and values, thus taking advantage of the benefits provided by this type of approaches.

### *4.3. Supporting the visualization of relationships between values and agile principles*

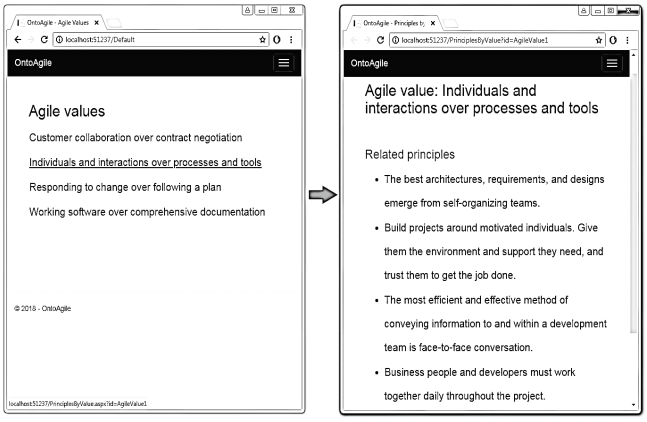


Figure 4. Displaying relationships in OntoAgile using a Web application.
Source: The Authors.

The third case of implementation is related to the development of a web application that allows visualizing the agile principles that serve as support for each of the four agile values based on SPARQL queries sent to OntoAgile.

In Fig. 4 it can be seen that the initial page of the application shows the four agile values in the form of links that allows to navigate to a page that shows the agile principles that serve as support for each value.

The development of this web application allowed to establish an architecture that serves as a starting point to add new queries to OntoAgile, such as visualizing the agility indicators associated with each principle in order to guide the assessment of the agility of software processes.

### *4.4. Answering the competency questions through SPARQL*

In order to evaluate the purpose fulfillment of the ontology, the translation of each one of the competence questions expressed in natural language to SPARQL queries was carried out. As a result of the execution of the queries, it was proven that OntoAgile is able to answer all the competency questions proposed in the initial stage of its development. In Table 4 can be observed the competency questions expressed in SPARQL and the result obtained. An example of the execution of some of the competency questions using Protégé is available at: http://bit.ly/OntoAgileEvaluation

### *4.5. Comparing ontologies for agile approaches*

Table 5 presents the comparison between OntoAgile and existing ontologies for agile approaches. To carry out the comparison, the following criteria were used: C1) the ontology is independent of a particular agile approach, C2) the concepts of the agile manifesto are included, C3) integration with existing ontologies, C4) competency questions are defined, C5) the ontology is integrated with software tools, C6) a formal language is used to represent the ontology, and C7) the ontology is available in a public repository.

According to the comparison, it can be observed that although there are ontologies that allow agile software processes to be instantiated, there is no evidence of a formal evaluation that ensures compliance with its academic or industrial purpose. In addition, existing ontologies are not available in public repositories that facilitate their use and evaluation.

Table 4.
Competency questions expressed in SPARQL.

| Competency question | SPARQL Query | Result |
|---|---|---|
| CQ1: What is the relationship between values and agile principles? | SELECT ?e1 ?rel WHERE { ?rel rdfs:domain ?e1 . ?rel rdfs:range ontoagile:Agile_principle . FILTER (?e1 IN (ontoagile:Agile_value )) } | Agile_value isSupportedByPrinciple |
| CQ2: What types of software development approaches are guided by agile values? | SELECT ?Approach WHERE { ?r rdfs:domain ontoagile:Agile_value . ?r rdfs:range ?Approach . ?Approach rdfs:subClassOf ontoagile:Software_development_life_cycle } | Hybrid_approach<br>Agile_approach |
| CQ3: What are the main elements that make up a software process? | SELECT ?ProcessElement WHERE { ?r rdfs:domain ontoagile:Process . ?r rdfs:range ?ProcessElement } | Resource - Activity<br>Product - Stage |
| CQ4: What is the relationship between the activities of the process and the agile principles? | SELECT ?e1 ?rel1 ?e2 ?rel2 WHERE { ?rel1 rdfs:domain ?e1 . ?rel1 rdfs:range ?e2 . ?rel2 rdfs:domain ?e2 . ?rel2 rdfs:range ontoagile:Activity . FILTER (?e1 IN (ontoagile:Agile_principle )) } | Agile_principle isRelatedToIndicator Indicator isAffectedByActivity |
| CQ5: What is the relationship between the resources used in the process and the agile principles | SELECT ?e1 ?rel1 ?e2 ?rel2 WHERE { ?rel1 rdfs:domain ?e1 . ?rel1 rdfs:range ?e2 . ?rel2 rdfs:domain ?e2 . ?rel2 rdfs:range ontoagile:Resource . FILTER (?e1 IN (ontoagile:Agile_principle )) } | Agile_principle isRelatedToIndicator Indicator isAffectedByResource |
| CQ6: What is the relationship between the products generated in the process and the agile principles? | SELECT ?e1 ?rel1 ?e2 ?rel2 WHERE { ?rel1 rdfs:domain ?e1 . ?rel1 rdfs:range ?e2 . ?rel2 rdfs:domain ?e2 . ?rel2 rdfs:range ontoagile:Product . FILTER (?e1 IN (ontoagile:Agile_principle )) } | Agile_principle isRelatedToIndicator Indicator isAffectedByProduct |

Source: The Authors.

Table 5.
Ontologies comparison.

| Ontology | C1 | C2 | C3 | C4 | C5 | C6 | C7 |
|---|---|---|---|---|---|---|---|
| XPO [20] | | | | | | ✓ | |
| FDD [27] | | | | | | ✓ | |
| K-CRIO [28] | ✓ | | ✓ | | | ✓ | |
| OSDAS [30] | ✓ | ✓ | | | | | |
| ONTOAGILE | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

Source: The Authors.

## 5. Conclusions and future work

Although the adoption of agile approaches has brought great benefits to small and medium-sized software companies, their application in development environments involves a total transformation of the approach and paradigm of the traditional development applied until now in the processes of organizations. However, the implementation in some cases is carried out informally, without a correct application and institutionalization of the values, principles and agile practices, in this regard, many organizations fail because of: (i) the excessive informality in the application of the practices, (ii) false expectations when thinking that the agile methodologies are the entire solution to all the problems and/or (iii) the constant use of bad practices that hinder the teams' daily work. Additionally, the adoption of agile approaches is particularly difficult since their implementation in many cases lacks formal support to back up in a more organized and systematic way the institutionalization of practices, even with other approaches.

Considering the above, some researchers and research groups have defined a series of proposals to support the work within this research stream. However, efforts have been made without considering a common and consistent terminology that allows to create and share the knowledge generated around this research domain. This has increased the confusion and terminological conflict that affects both researchers and organizations that make use of that knowledge. A consistent terminology focused on the adoption of an agile approach, and the evaluation of the agility of software processes can become an important tool to understand and support organizations in their process improvement projects, as well as to strengthen this research domain.

In this paper, a summary of a semi-formal ontology for the agile domain of software processes of organizations called OntoAgile has been presented. The vocabulary described in OntoAgile allows to organize the knowledge related to the agile software development in a generic and formal way, reducing problems of inconsistency, integrity and ambiguity of the concepts and terms present in this research domain. Also, OntoAgile has been represented by the OWL language (available at http://bit.ly/owl-ontoagile) in order to facilitate its use by the academic community and software organizations.

On the other hand, OntoAgile's main objective is to provide a basis for the discussion of the terms, concepts and relationships identified in this research domain, which has been applied in three implementation cases: (i) the instantiation of the elements present in the Agile Scrum framework for being one of the most EAs currently used by software organizations for the management of their projects, (ii) the instantiation of some of the concepts that allow supporting the evaluation of the agility of the software processes based on compliance with agile principles and the analysis of activities, roles and products of the process was carried out; and (iii) a web application to visualize the relationship between values and agile principles through SPARQL queries was implemented. In addition, it was possible to answer the competency questions that were established in the purpose of the ontology through the SPARQL query language.

The results obtained in the development of this document will be used to address some future work in the following order: i) It is expected to evaluate OntoAgile through the application of case studies in software organizations that use agile approaches in order to instantiate their software processes, ii) update and extend the ontology based on the recommendations obtained with its application in some organizations as real case studies, iii) use ontology as a conceptual basis for the definition of a framework that supports the evaluation of the agility of the software processes, iv) carry out the application of OntoAgile in the harmonization of multiple models and approaches both traditional and agile, which will ease the struggle around the integration of different reference models, and finally (v) we will focus on automation of the assessment stage, since the assessment process is currently a manual area. Therefore, as future work, the next step in this research project will focus on the development of algorithms that allow us to improve and expand the capacity of the assessment process through automation.

## Acknowledgement

Wilson Ortega, César Pardo and Francisco Pino appreciate the contribution of the Universidad del Cauca where they work in the Systems Department as Assistant, Associate and Full Professors, respectively.

## References

[1] Beck, K. et al., Manifesto for agile software development, 2001. [Online]. [date of reference: June 1rd of 2018]. Available at: http://agilemanifesto.org/.

[2] Ozcan-Top, O. and Demirörs, O., A reference model for software agility assessment: AgilityMod, Procs. 15th International Conference on Process Improvement and Capability Determination in Software, Systems Engineering and Service Management - SPICE, 2015, pp. 145-158. DOI: 10.1007/978-3-319-19860-6_12

[3] Takeuchi, H. and Nonaka, I., The new product development game, Harvard Business Review, 64(1), pp. 137-146, 1986. DOI: 10.1016/0737-6782(86)90053-6

[4] Beck, K.C., A. Extreme programming explained. Pearson Education, UK, 1999.

[5] Cockburn, A., Crystal clear: a human-powered methodology for small teams. Pearson Education, UK, 2004.

[6] Poppendieck, M. and Poppendieck, T., Lean software development: an agile toolkit (The agile software development series), Addison-Wesley Professional, USA, 2003.

[7] Highsmith, J.A., Adaptive software development: a collaborative approach to managing complex systems, 12. Dorset House Publishing Co Inc., New York, USA, 2000.

[8] Stapleton, J., DSDM, Dynamic Systems Development Method: the method in practice. Addison-Wesley Professional, USA, 1997.

[9] Palmer, S.R. and Felsing, M., A practical guide to feature driven development. Prentice Hall, USA, 2002.

[10] Ambler, S.W., Agile unified process. [Online]. [date of reference: June 1st of 2018]. Available at: http://www.ambysoft.com/unifiedprocess/agileUP.html.

[11] KanBan Fundamentals, Software. [Online]. [date of reference: June 1st of 2018]. Available at: http://www.scrumhub.com/kanban-fundamentals/.

[12] Leffingwell, D., Scaled agile framework - SAFe, 2018. [Online]. [date of reference: June 1st of 2018]. Available at: https://www.scaledagile.com/.

[13] Vodde, B. and Larman, C., LeSS Framework. [Online]. [date of reference: June 1st of 2018]. Available at: http://less.works/.
[14] Schwaber, K., The Nexus guide. [Online]. [date of reference: June 1st of 2018]. Available at: http://bit.ly/nexus-guide.
[15] VersionOne Inc., The 11th annual State of Agile, 2017. [Online]. [date of reference: June 1st of 2018]. Available at: https://explore.versionone.com/state-of-agile.
[16] Campanelli, A.S. and Parreiras, F.S., Agile methods tailoring - A systematic literature review, Journal of Systems and Software, 110, pp. 85-100, 2015. DOI: 10.1016/j.jss.2015.08.035
[17] SEI. Capability maturity model for software - CMMI for Development V1.3, Pittsburg, Pennsylvania, USA, 2010.
[18] Pardo, C., García, F., Pino, F., Piattini-Velthuis, M. and Baldassarre, M.T., A reference ontology for harmonizing process-reference models, Revista Facultad de Ingeniería, Universidad de Antioquia, (73), pp. 29-42, 2014.
[19] Beck, K., Extreme programming explained. Addison-Wesley Professional, 1999.
[20] Ceravolo, P., Damiani, E., Marchesi, M., Pinna, S. and Zavatarelli, F., A ontology-based process modelling for XP, Proceedings of the Tenth Asia-Pacific Software Engineering Conference, 2004, pp. 236-242. DOI: 10.1109/APSEC.2003.1254376
[21] Scacchi, W., Understanding software process redesign using modeling, analysis and simulation., Software Process: improvement and practice, 5(2-3), pp. 183-195, 2000. DOI: 10.1002/1099-1670(200006/09)5:2/3<183::AID-SPIP115>3.0.CO;2-D
[22] Ferchichi, A. et al., An ontology for quality standards integration in software collaborative projects, First International Workshop on Model Driven Interoperability for Sustainable Information Systems, MDISIS'08., 2008, pp. 17-30.
[23] Liao, L., Qu, Y. and Leung, H.K.N., A software process ontology and its application, 4th International Semantic Web Conference, 2008.
[24] Salviano, C.F. and Figueiredo, A.M.C.M., Unified basic concepts for process capability models, Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE'08), 2008, pp. 173-178.
[25] Gokhan-Halit, S. and Mieczyslaw, M.K., An OWL ontology for representing the CMMI-SW model, The 2nd International Workshop on Semantic Web Enabled Software Engineering, 2006.
[26] IEEE, C.S. Guide to the Software engineering body of knowledge. SWEBOK. Los Alamitos, California, USA, 2004.
[27] Siddiqui, F. and Afshar-Alam, M., Ontology based application model for feature driven development, Proceedings of the 5th Indian International Conference on Artificial Intelligence, IICAI 2011, 2011, pp. 1125-1137.
[28] Lin, Y., Hilaire, V., Gaud, N. and Koukam, A., Scrum conceptualization using K-CRIO ontology, Data-Driven Process Discovery and Analysis, 2012, pp. 189-211. DOI: 10.1007/978-3-642-34044-4_11
[29] Zualkernan, I.A., An Ontology-Driven approach for generating assessments for the scrum software process, new trends in software methodologies, tools and techniques, Proceedings of the Seventh SoMeT 2008, 2008.
[30] Parsons, D., Agile software development methodology, an ontological analysis, Proceedings of 9th International Conference on Applications and Principles of Information Science, 2010. DOI: 10.13140/2.1.3298.6883
[31] Strode, D.E., A dependency taxonomy for agile software development projects, Information Systems Frontiers, 18(1), pp. 23-46, 2016. DOI: 10.1007/s10796-015-9574-1
[32] Sutling, K., Mansor, Z., Widyarto, S., Letchmunan, S. and Arshad, N.H., Agile project manager behavior: the taxonomy, 8th Malaysian Software Engineering Conference MySEC 2014, 2014, pp. 234-239. DOI: 10.1109/MySec.2014.6986020
[33] Pardo-Calvache, C.J., García-Rubio, F.O., Piattini- Velthuis, M., Pino-Correa, F.J. and Baldassarre, M.T., A reference ontology for harmonizing process- reference models. Revista Facultad de Ingeniería Universidad de Antioquia, (73), pp. 29-42, 2014.
[34] García, F. et al., Towards a consistent terminology for software measurement, Information & Software Technology, 48(8), pp. 631-644, 2006. DOI: 10.1016/j.infsof.2005.07.001
[35] Mendes, O. and Abran, A., Software engineering ontology: a development methodology, Metrics News, 2004, pp. 68-76.
[36] Liao, L., Qu, Y. and Leung, H.K.N., A software process ontology and its application, Proceedings of the 4th International Semantic Web Conference (ISWC 2005), 2005.
[37] Fensel, D., Ontology-based knowledge management, Computer, 35(11), pp. 56-59, 2002. DOI: 10.1109/MC.2002.1046975
[38] Fernández, M., Gómez-Pérez, A. and Juristo, N., Methontology: from ontological art towards ontological engineering, Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, 1997, pp. 33-40.
[39] Gruber, T.R., A translation approach to portable ontology specifications, Knowl. Acquis., 5(2), pp. 199-220, 1993. DOI: 10.1006/knac.1993.1008
[40] Hikita, T. and Matsumoto, M.J., Business process modelling based on the ontology and first-order logic, Proc. 3rd Int. Conf. on Enterprise Information Systems (ICEIS'2001), 2001, pp. 717-723.
[41] Tautz, C. and Wangenheim, C.G., REFSENO: a representation formalism for software engineering ontologies, Fraunhofer IESE-Report No. 015.98/E V1.1, 1998.
[42] Reimer, U., Introduction to knowledge representation: net-like and schema-based representation formats (in German), Teubner, Stuttgart, Germany, 1991.
[43] Sikos, L.F., Description logics in multimedia reasoning, 1st ed. Springer International Publishing, 2017.
[44] Pinto, H.S., Gomez-Perez, A. and Martins, J.P., Some issues on ontology integration, In Proceedings of IJCAI99's Workshop on ontologies and problem solving methods: lessons learned and future trends, 1999, pp. 7-12.
[45] Euzenat, J. and Shvaiko, P., Ontology matching, 1st Ed. Springer-Verlag Berlin Heidelberg, 2007. DOI: 10.1007/978-3-540-49612-0
[46] Ruiz, F. and Hilera, J., Using ontologies in software engineering and technology, in: Calero, C., Ruiz, F. and Piattini, M., Eds., Ontologies for Software Engineering and Software Technology, Springer-Verlag, 2006, pp. 49-102. DOI: 10.1007/3-540-34518-3_2
[47] ISO/IEC/IEEE 12207:2017(E), Systems and software engineering - Software life cycle processes, First edition, 2017.
[48] Kuhrmann, M. et al., Hybrid software and system development in practice: waterfall, scrum, and beyond, Proceedings of the 2017 International Conference on Software and System Process - ICSSP 2017, 2017, pp. 30-39. DOI: 10.1145/3084100.3084104
[49] Petersen, K. and Wohlin, C., The effect of moving from a plan-driven to an incremental software development approach with agile practices: an industrial case study, Empirical Software Engineering, 15(6), pp. 654-693, 2010. DOI: 10.1007/s10664-010-9136-6
[50] Bassil, Y., A simulation model for the waterfall software development life cycle, International Journal of Engineering & Technology, 2(5), pp. 2049-3444, 2012. DOI: 10.15680/ijircce.2015.0305013
[51] Saraiva-de Almeida, R., Pardo-Calvache, C.J. and Mira-da Silva, M., An Ontology-based model for ITL process assessment using TIPA for ITIL, Comunications in Computer and Information Science - CCIS 918, P. In press.
[52] OWL 2 Web Ontology Language, 2012. [Online]. [date of reference: August 27th of 2018]. Available at: https://www.w3.org/TR/owl2-overview/.
[53] Protégé. [Online]. [date of reference: August 27th of 2018]. Available at: https://protege.stanford.edu/.
[54] HermiT OWL Reasoner. [Online]. [date of reference: August 26th of 2018]. Available at: http://www.hermit-reasoner.com/

**W.A. Ortega-Ordoñez,** received his BSc. degree in Systems Engineering (2004) and Sp. in Project Management (2007), all of them from the Universidad del Cauca, Colombia. Currently, he is an assistant professor and MSc. student in Computer Science at the Faculty of Electronic Engineering and Telecommunications at the Universidad del Cauca, Colombia. He is member of the GTI Research Group and his research interests include agile methods and approaches, software process and software engineering.
ORCID: 0000-0001-6558-7891

**C.J. Pardo-Calvache,** received his MSc (2009) and International PhD (2012) degrees in Computer Science from the University of Castilla-La Mancha (UCLM), Spain. Currently, he is an associate professor at the Faculty of Electronic Engineering and Telecommunications at the

Universidad del Cauca, Colombia. He is member of the GTI Research Group and his research interests include agile approaches, software process improvement, harmonization of multiple models and standards and quality characteristics of process-supported software products.
ORCID: 0000-0002-6907-2905

**F.J. Pino-Correa,** received his European PhD (2010) degree in Computer Science from the University of Castilla-La Mancha (UCLM), Spain. Currently, he is a full professor at the Electronic and Telecommunications Engineering Faculty at the University del Cauca, Colombia. He is a member of the IDIS Research Group and his research interests include software process improvement in small companies and qualitative research methods for Software Engineering.
ORCID: 0000-0003-0668-4485