# A Simplified Agile Methodology for Ontology Development

Silvio Peroni[1]

DASPLab, DISI, University of Bologna, Bologna, Italy
silvio.peroni@unibo.it

**Abstract.** In this paper we introduce *SAMOD*, a.k.a. *Simplified Agile Methodology for Ontology Development*, a novel agile methodology for the development of ontologies by means of small steps of an iterative workflow that focuses on creating well-developed and documented models starting from exemplar domain descriptions.
**RASH:**
https://w3id.org/people/essepuntato/papers/samod-owled2016.html

**Keywords:** Agile Ontology Development Methodology, Conceptual Modelling, Knowledge Engineering, OWL Ontologies, Ontology Engineering, SAMOD, Test-Driven Development

## 1 Introduction

In the past twenty years, the Software Engineering domain has seen the proposal of new *agile* methodologies for software development, in contrast with *highly-disciplined* processes that have characterised such discipline from its beginning. Following this trend, recently agile development methodologies have been proposed in the field of Ontology Engineering as well. Such kind of methodologies would be preferred when the ontology to develop should be composed by a limited amount of ontological entities – while the use of highly-structured and strongly-founded methodologies remain valid and, maybe, mandatory to solve and model incredibly complex enterprise projects.

One of main characteristics that ontology development methodologies usually have is the use of *exemplar data* during the development process so as to:

- *avoid inconsistencies* – a common mistake when developing a model is to make a TBox that is consistent if considered alone and that becomes inconsistent when we define an ABox for it, even if all the classes and properties are completely satisfiable. Using real-world data, as exemplar of a particular scenario of the domain we are modelling, can definitely prevent this problem;
- *have self-explanatory and easy-understandable models* – trying to implement a particular real-world and significative scenario related to a model by using data allows one to better understand if each TBox entity has a meaningful name that fits for describing clearly the intent and the usage of the entity itself. This makes users understanding a model without spending a lot of effort for reading any comment or documentation. The use of data as part of the ontology development obliges ontology engineers

and developers to think about the possible ways users will understand and use the ontology they are developing, in particular the very first time they look at it;

- *provide examples of usage* – producing data within the development process means to have a bunch of exemplars that describe the usage of the model in real-world scenarios. This kind of documentation, implicitly, allows users to apply a learn-by-example approach [1] in understanding the model and during their *initial skill acquisition* phase.

In this paper we introduce *SAMOD* (*Simplified Agile Methodology for Ontology Development*), a novel agile methodology for the development of ontologies, partially inspired to the Test-Driven Development process in Software Engineering [2]. The methodology is organised in small steps within an iterative process that focuses on creating well-developed and documented models by using significative exemplar of data, so as to produce ontologies that are always ready-to-be-used and easily-understandable by humans (i.e., the possible customers) without spending a lot of effort. Described with details in the following sections, SAMOD is the result of our dedication to the development of ontologies in the past eight years. While the first draft of the methodology has been proposed in 2010 as starting point for the development of the Semantic Publishing and Referencing Ontologies[1] [12], it has been revised several times so as to come to the current version presented in this paper – which has been already used for developed several ontologies, such as the Vagueness Ontology[2], the F Entry Ontology[3], the OA Entry Ontology[4], and the Imperial Data Ontology for describing selling data related to the fashion design industry (which has not been released yet). While the complete introduction to SAMOD is provided in [13], in this paper we provide a summary of it and we discuss some outcomes of an user-based evaluation we have conducted in the past months.

The rest of the paper is organised as follows. In Section 2 we introduce the the entities involved in the methodology. In Section 3 we present all the step of SAMOD, providing details on each of them. In Section 4 we discuss the outcomes of some experiments where we asked to subjects with limited knowledge about Semantic Web technologies and Ontology Engineer to use SAMOD for developing an ontology. In Section 5 we present some of the most relevant related works in the area. Finally, in Section 6 we conclude the paper sketching out some future works.

## 2 Preliminaries

The kinds of people involved in SAMOD are domain experts and ontology engineers. A *domain expert*, or *DE*, is a professional with expertise in the domain to be described by the ontology, and she is is mainly responsible to define, often in natural language, a detailed description of domain that has to

[1] `http://www.sparontologies.net/`
[2] `http://www.essepuntato.it/2013/10/vagueness`
[3] `http://www.essepuntato.it/2014/03/fentry`
[4] `http://purl.org/emmedi/oaentry`

be modelled. An *ontology engineer*, or *OE*, is a person who, starting from an informal and precise description of a particular problem or domain provided by DEs, construct meaningful and useful ontologies by using a particular formal language, such as OWL 2[5].

A *motivating scenario* (MS) [18] is a small story problem that provides a short description and a set of informal and intuitive examples to the problem it talks about. In SAMOD, a motivation scenario is composed by a *name* that characterises it, a natural language *description* that presents a problem to address, and one or more *examples* according to the description.

An *informal competency question* (CQ) [18] is a natural language question that represents an informal requirement within a particular domain. In SAMOD, each informal competency question is composed by an unique *identifier*, a natural language *question*, the kind of *outcome* expected as answer, some *exemplar answers* considering the examples provided in the related motivating scenario[6], a list of identifiers referring to higher-level informal competency questions *requiring* this one, if any.

A *glossary of terms* (GoT) [7]is a list of term-definition pairs related to terms that are commonly used for talking about the domain in consideration. The term in each pair may be composed by one or more words or verbs, or even by a brief sentence, while the related definition is a natural language explanation of the meaning of such term.

SAMOD prescribes an iterative process which aims to build the final model through a series of small steps. At the end of each iteration a particular preliminary version of the final model is released. Within a particular iteration $i_n$, the *current model* is the version of the final model released at the end of the iteration $i_{n-1}$. Contrarily, a *modelet* is a stand-alone model describing a particular domain. By definition, a modelet does not include entities from other models and it is not included in other models.

A *test case* $T_n$, produced in the $n^{th}$ iteration of the process, is a sextuple including a motivating scenario, a list of scenario-related informal competency questions, a glossary of terms for the domain addressed by the motivating scenario, a *TBox* of the ontology implementing the description introduced in the motivating scenario, an exemplar *ABox* an exemplar dataset implementing all the examples described in the motivating scenario according to the TBox, and a set of SPARQL[7] queries (SQ) formalising the informal competency questions. A *bag of test cases* (*BoT*) is a set of test cases.

Given as input a MS, the TBox and the related GoT – a *model test* aims at checking the validity of the TBox against specific requirements, i.e. [**formal requirement**] understanding (even by using appropriate unit tests [20]) whether the TBox is consistent, and [**rhetorical requirement**] understand-

[5] `http://www.w3.org/TR/owl2-syntax/`

[6] Note that if there are no data in any example of the motivating scenario that answer to the question, it is possible that either the competency question is not relevant for the motivating scenario or the motivating scenario misses some important exemplar data. In those cases one should remove the competency question or modifying the motivating scenario accordingly.

[7] `http://www.w3.org/TR/sparql11-query/`

ing whether the TBox covers the related MS and whether the vocabulary used by the TBox is appropriate.

Given as input a MS, the TBox formalising it and an ABox built according to the TBox, and considering the examples described in the MS, a *data test* aims at checking the validity of the model and the dataset and against specific requirements, i.e. [**formal requirement**] understanding whether the TBox is still consistent when considering the Abox, and [**rhetorical requirement**] understanding whether the ABox describes all the examples accompanying the motivating scenario completely.

Given as input a TBox, a related ABox, a set of CQs, and a set of SQs, each mapping a particular CQ question, a *query test* aims at checking the validity of the TBox, the ABox and each SQ query against specific requirements, i.e. [**formal requirement**] understanding whether each SQ is well-formed and can correctly run on the Tbox+ABox, and [**rhetorical requirement**] understanding whether each CQ is mapped into an appropriate SQ and whether, running each of them upon the TBox+ABox, the result conforms to the expected outcome of each CQ and in the ABox.

## 3 Methodology

SAMOD is based on the following three iterative steps (briefly summarised in Fig. 1) – where each step ends with the release of a snapshot of the current state of the process called *milestone*:

1. OEs collect all the information about a specific domain, with the help of DEs, in order to build a modelet formalising the domain in consideration, following certain ontology development principles, and then create a new test case that includes the modelet. If everything works fine (i.e., all model, data and query test are passed) release a milestone and proceed;
2. OEs merge the modelet of the new test case with the current model produced by the end of the last process iteration, and consequently update all the test cases in BoT specifying the new current model as *TBox*. If everything works fine (i.e., all model, data and query test are passed according to their formal requirements only) release a milestone and proceed;
3. OEs refactor the current model, in particular focussing on the last part added in the previous step, taking into account good practices for ontology development processes. If everything works fine (i.e., all model, data and query test are passed) release a milestone. In case there is another motivating scenario to be addressed, iterate the process, otherwise stop.

The next sections elaborate on those steps introducing a real example[8] considering a generic iteration $\mathrm{i_n}$.

### 3.1 Define a new test case

OEs and DEs work together to write down a motivating scenario $\mathrm{MS_n}$, being as close as possible to the language DEs commonly use for talking about the domain. An example of motivating scenario is illustrated in Table 1.

[8] The whole documentation about the example is available at `http://www.essepuntato.it/2013/10/vagueness/samod`.

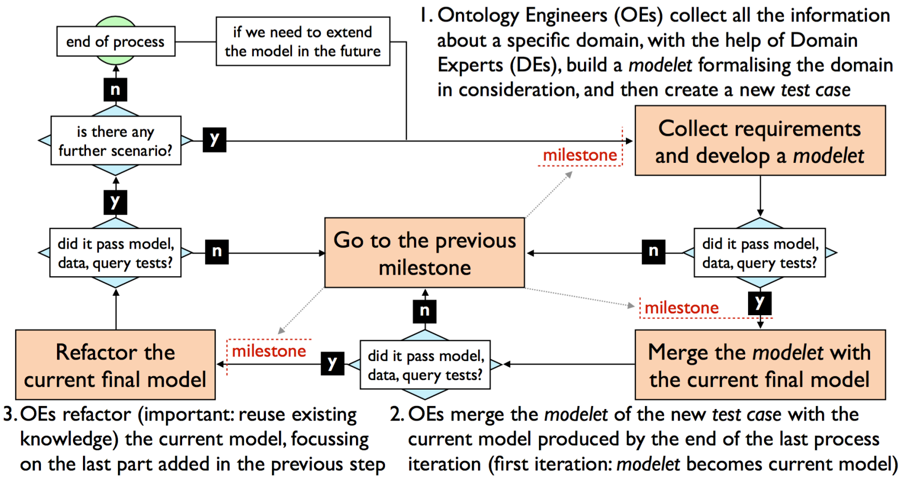


**Fig. 1.** A brief summary of SAMOD, starting with the "Collect requirements and develop a modelet" step.

**Table 1.** An excerpt of a motivating scenario.

| | |
|---|---|
| **Name** | Vagueness of the TBox entities of an ontology |
| **Description** | Vagueness is a common human knowledge and language phenomenon, typically manifested by terms and concepts like High, Expert, Bad, Near etc. In an OWL ontology vagueness may appear in the definitions of classes, properties, datatypes and individuals. For these entities a more explicit description of the nature and characteristics of their vagueness/non-vagueness is required. [...] |
| **Example 1** | Silvio Peroni thinks that the class TallPerson is vague since there is no way to define a crisp height threshold that may separate tall from non-tall people. Panos Alexopoulos, on the other hand, considers someone as tall when his/her height is at least 190cm. Thus, for Panos, the class TallPerson is not vague. |
| **Example 2** | In an company ontology, the class StrategicClient [...] |

Given a motivating scenario, OEs and DEs should produce a set of informal competency questions $\mathrm{CQ_n}$, each of them identified appropriately. An example of an informal competency question, formulated starting from the motivating scenario in Table 1, is illustrated in Table 2.

Now, having both a motivating scenario and a list of informal competency questions, KEs and DEs write down a glossary of terms $\mathrm{GoT_n}$. An example of glossary of terms is illustrated in Table 3.

**Table 2.** An example of competency question.

| | |
|---|---|
| **Identifier** | 3 |
| **Question** | What are all the entities that are characterised by a specific vagueness type? |
| **Outcome** | The list of all the pairs of entity and vagueness type. |
| **Example** | StrategicClient, quantitative<br>StrategicClient, qualitative |
| **Depends on** | 1 |

**Table 3.** An excerpt of a glossary of terms.

| Term | Definition |
|---|---|
| description of vagueness | The descriptive characterisation of vagueness to associate to an ontological entity by means of an annotation. It specifies a vagueness type and provides at least one justification for considering the target ontological entity vague. |
| vagueness type | A particular kind of vagueness that characterises the entity. |
| [...] | [...] |

The remaining part of this step is led by OEs only[9], who are responsible of developing a modelet according to the motivating scenario, the informal competency questions and the glossary of terms[10].

In doing that work, they must strictly follow the following principles:

- **Keep it small.** Keeping the number of developed ontology entities small – e.g. Miller's magic number "$7 \pm 2$" [11] entities per type (classes, object properties, data properties);
- **Use patterns.** OEs should take into consideration existing knowledge, in particularexisting and well-documented patterns – the Semantic Web Best Practices and Deployment Working Group page[11] and the Ontology Design Patterns portal[12]are both valuable examples – and other widely-adopted Semantic Web vocabularies;
- **Middle-out development.** OEs should define firstly the most relevant concepts and latterly adding the most high-level and most concrete ones – such middle-out approach [19] allows one to avoid unnecessary effort during the development because detail arises only as necessary, by adding sub- and super-classes to the basic concepts;
- **Keep it simple.** The modelet must be designed according to the information obtained previously (MS, CQs, GoT) in an as quick as possible way, spending the minimum effort and without adding any unnecessary

[9] The OEs involved in our methodology can vary in number. However SAMOD has been thought for being used also by one OE only.

[10] Note that it is possible that multiple entities (i.e. classes, properties, individuals) are actually hidden behind one single definition in the glossary of terms.

[11] `http://www.w3.org/2001/sw/BestPractices/OEP/`

[12] `http://www.ontologydesignpatterns.org/`

semantic structure – avoiding to think about inferences at this stage, and rather describing the motivating scenario fully.

- **Self-explanatory entities.** The aim of each ontological entity must be understandable by humans simply looking at its local name (i.e., the last part of the entity IRI). No labels and comments have to be added at this stage and all the entity IRIs must not be opaque – class local names has to be capitalised and in camel-case notation if composed by more than one word (e.g. *DescriptionOfVagueness*), property local names must start with a non-capitalised [13]verb[14] and in camel-case notation if composed by more than one word; (e.g. *wasAttributedTo*), and individual local names must be non-capitalised and dash-separated if composed by more than one word (*quantitative-vagueness*).

The goal of OEs is to develop a $\mathrm{modelet_n}$, eventually starting from a graphical representation written in a proper visual language – such as Graffoo [6] – so as to convert it automatically in OWL by means of appropriate tools, e.g. DiTTO [8].

Starting from the OWL version $\mathrm{modelet_n}$, OEs proceed in four phases:

1. run a model test on $\mathrm{modelet_n}$. If it succeeds, then
2. create an exemplar dataset $\mathrm{ABox_n}$ that formalises all the examples introduced in the MS according to $\mathrm{modelet_n}$. Then, it runs a data test and, if succeeds, then
3. write $\mathrm{SQ_n}$ as many informal CQ related to the MS. Then, it runs a query test and, if it succeeds, then
4. create a new test case $\mathrm{T_n} = (\mathrm{MS_n}, \mathrm{CQ_n}, \mathrm{GoT_n}, \mathrm{modelet_n}, \mathrm{ABox_n}, \mathrm{SQ_n})$ and add it in BoT.

When running the model test, the data test and the query test, it is possible to use any appropriate available software to support the task, such as reasoners (Pellet[15], HermiT[16]) and query engines (Jena[17], Sesame[18]).

Any failure of any test that is considered a serious issue by the OEs results in getting back to the more recent milestone. It is worth mentioning that an exception should be also arisen if OEs think that the motivating scenario $\mathrm{MS_n}$ is to big to be covered by one only iteration of the process. In this case, it may be necessary to re-schedule the whole iteration, for example split adequately the motivating scenario in two new ones.

### 3.2 Merge the current model with the modelet

At this stage, OEs merge $\mathrm{modelet_n}$, included in the new test case $\mathrm{T_n}$, with the current model, i.e., the version of the final model released at the end of the previous iteration (i.e., $\mathrm{i_{n-1}}$). OEs have to proceed in three consecutive steps:

[13] http://www.jenitennison.com/blog/node/128
[14] http://www.jenitennison.com/blog/node/128
[15] http://clarkparsia.com/pellet
[16] http://hermit-reasoner.com/
[17] http://jena.sourceforge.net/
[18] http://www.openrdf.org/

1. to define a new $\mathrm{TBox_n}$ merging[19] of the current model with $\mathrm{modelet_n}$, by adding all the axioms from the current model and $\mathrm{modelet_n}$ to $\mathrm{TBox_n}$ and then collapse semantically-identical entities, e.g. those that have similar names and that represent the same entity (for instance *Person* and *HumanBeing*);
2. to update all the test cases in BoT, swapping the *TBox* of each test case with $\mathrm{TBox_n}$ and refactoring each *ABox* and *SQ* according to the new entity names if needed, so as to refer to the more recent model;
3. to run the model test, the data test and the query test on all the test cases in BoT, according to their formal requirements only;
4. to set $\mathrm{TBox_n}$ as the new current model.

Any serious failure of any test, that means something went bad in updating the test cases in BoT, results in getting back to a previous milestone. In this case, OEs have to consider the more recent milestones, if they think there was a mistake in a procedure of this step, or, the milestones before, if the failure is demonstrably given by any of the components of the new test case $\mathrm{T_n}$.

### 3.3 Refactor the current model

In the last step, OEs work to refactor the current model, shared among all the test cases in BoT, and, accordingly, each *ABox* and *SQ* of each test case, if needed. In doing that task, OEs must strictly follow the following principles:

– **Reuse existing knowledge.** Reusing concepts and relations defined in other models is encouraged and often labelled as a common good practice [19];
– **Document it.** Adding annotations – i.e., labels (i.e., *rdfs:label*), comments (i.e., *rdfs:comment*), and provenance information (i.e., *rdfs:isDefinedBy*) – to ontological entities, so as to provide natural language descriptions of them and to allow tools (e.g., LODE [14]) to produce an HTML human-readable documentation from the ontology source;
– **Take advantages from technologies.** Enriching the current model by using all the capabilities offered by OWL 2 in order to infer automatically as much information as possible starting from a (possible) small set of real data, by avoiding over-classifications (e.g. assertions that may be automatically inferred by a reasoner).

Finally, once the refactor is finished, OEs have to run the model test, the data test and the query test on all the test cases in BoT. This is an crucial task to perform, since it guarantees that the refactoring has not damaged any existing conceptualisation implemented in the current model.

### 3.4 Output of an iteration

Each iteration of SAMOD aims to produce a new test case that will be added to the bag of test cases (BoT). Each test case describes a particular aspect of

[19] If $\mathrm{i_n}$ is actually $\mathrm{i_1}$, then the $\mathrm{modelet_n}$ becomes the current model since no previous model is actually available.

the same model, i.e., the *current model* under consideration after one iteration of the methodology.

In addition of being integral part of the methodology process, each test case represents a complete documentation of a particular aspect of the domain described by the model, due to the natural language descriptions (the motivating scenario and the informal competency questions) it includes, as well as the formal implementation of exemplar data (the ABox) and possible ways of querying the data compliant with the model (the set of formal queries). All these additional information should help end users in understanding, with less effort, what the model is about and how they can use it to describe the particular domain it addresses.

## 4 Experiments

We performed an experiment so as to understand to which degree SAMOD can be used by people with limited experience in Semantic Web technologies and Ontology Engineering. In particular, we organised user testing session so as to gather some evidences on the usability of SAMOD when modelling OWL ontologies.

We asked nine Computer Science and Law people – one professor, two post-docs, and six Ph.D. students – to use SAMOD for modelling a particular motivating scenario provided as exercise. SAMOD, as well as the main basics on Ontology Engineering, OWL, and Semantic Web technologies were introduced to the subjects during four lectures of four hours each. At the end of the last lecture, we asked them to answer three questionnaires: a background questionnaire containing questions on previous experience in Ontology Engineering and OWL; another questionnaire containing ten likert questions according to the System Usability Scale (SUS), which also allowed us to measure the sub-scales of pure Usability and pure Learnability, as proposed recently by Lewis and Sauro [10]; and a final questionnaire asking for the experience of using SAMOD for completing the task.

The mean SUS score for SAMOD was 67.25 (in a 0 to 100 range), approaching the target score of 68 to demonstrate a good level of usability (according to [16]). The mean values for the SUS sub-scales Usability and Learnability were 65.62 and 73.75 respectively. In addition, one score was calculated for each subject by considering the values of the answers given in the background questionnaire. We compared this scores (x-axis in Fig. 2) with the SUS values and the other sub-scales (y-axis) using the Pearson's $r$. As highlighted by the red dashed lines (referring to the related Least Squares Regression Lines), there is a positive correlation between the experience score and the SUS values – i.e., the more a subject knew about ontology engineering in general, the more SAMOD is perceived as usable and easy to learn – only in the latter case we had a statistical significant value with $p < 0.05$.

Axial coding of the personal comments expressed in the final questionnaires [17] revealed a small number of widely perceived issues. Overall the methodology proposed have been evaluated positively (described with adjectives such as "useful", "natural", "effective", "consistent" etc.) by 7 subjects, but it has also received criticisms by 5 subjects, mainly referring to the need of more

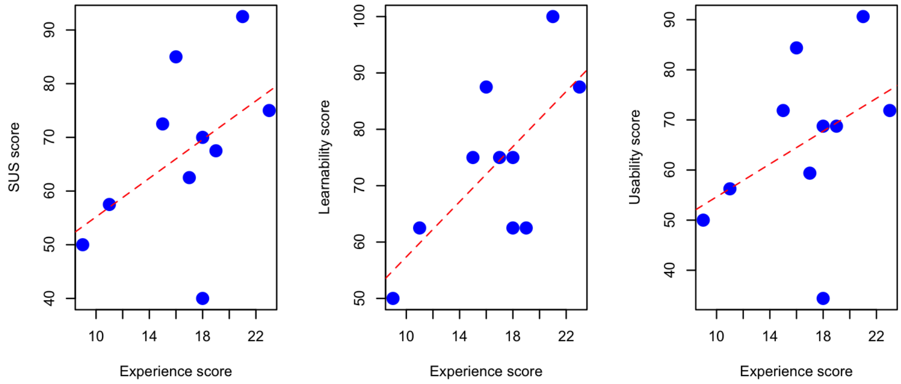


**Fig. 2.** Three comparisons between the SUS score (and its sub-scales) and the experience score by the subjects.

expertise in Semantic Web technologies and Ontology Engineering for using it appropriately. The use of the tests for assessing the ontology developed after a certain step has been appreciated as well (3 positive comments vs. 1 negative one), as well as the use of the scenarios and examples in the very first step of SAMOD (3 positive comments) and the implementation of competency questions in form of SPARQL queries (2 positive comments). All the outcomes of the questionnaires are available online in the SAMOD GitHub repository[20].

## 5 Related works

Several quick-and-iterative ontology development processes have been introduced recently, which could be preferred when the ontology to develop should be composed by a limited amount of ontological entities – while the use of highly-structured and strongly-founded methodologies (e.g. [7] [18] [19]) remain valid and, maybe, mandatory to solve and model incredibly complex enterprise projects. In this section we introduce some of the most interesting agile approaches to ontology development.

One of the first agile methodology introduced in the domain is *eXtreme Design* (*XD*) [15], which has been inspired by the eXtreme Programming methodology in Software Engineering. The authors described XD as "an approach, a family of methods and associated tools, based on the application, exploitation, and definition of ontology design patterns (ODPs) for solving ontology development issues". Summarising, XD is an agile methodology that uses pair design (i.e. groups of two ontology engineers working together during the development) and an iterative process which starts with the collection of *stories* and *competency questions* as requirements to address, and then propose the re-use of existing ontology design patterns for addressing such informal requirements.

Another recent approach has been proposed by Keet and Lawrynowicz in [9]. They propose to transfer concepts related to the Test-Driven Development

[20] `http://github.com/essepuntato/samod`

in Software Engineering [2] into the Ontology Engineering world. The main idea behind this methodology is that tests have to be run in advance before to proceed with the modelling of a particular (aspect of a) domain and, thus, they have to be proposed and developed in advance. Of course, the first execution of the tests should fail, since no ontology has been already developed for addressing them properly, while the ontology developed in future iterations of the process should result in passing the test eventually.

De Nicola and Missikoff [5] recently introduced their *Unified Process for ONtology building* methodology (a.k.a. *UPON Lite*), which is an agile ontology engineering method that place end users without specific ontology expertise (domain experts, stakeholders, etc.) at the centre of the process. The methodologies prescribes to follow an ordered set of six steps. Each step outputs a self-contained artefact immediately available to end users, and that is used as input of the subsequent step. This makes the whole process progressive and differential, and involves ontology engineerings only the the very last step of the process, i.e. when the ontology has to be formalised in some standard language.

## 6 Conclusions

In this paper we have introduced *SAMOD*, a *Simple Agile Methodology for Ontology Development*. In particular, we have introduced its process by describing in detail each of the steps which compose it, and we have also discussed the results of an experiment we have run involving nine people with no or limited expertise in Semantic Web technologies and Ontology Engineering.

In the future, we plan to involve a larger set of users so as to gather additional data about its usefulness, its usability, and its effectiveness. In addition, we plan to develop supporting tools for accompanying and facilitating users in each step of the methodology.

**Acknowledgements.** We would like to thank Jun Zhao for her precious comments and concerns about the initial drafts of SAMOD, David Shotton for our fruitful discussions when developing the SPAR Ontologies, Francesca Toni as one of the first users of such methodology, and Panos Alexopoulos as a co-author of the Vagueness Ontology that we used herein to introduce all the examples of the SAMOD development process.

## References

1. Atkinson, R. K., Derry, S. J., Renkl, A., Wortham, D. (2000). Learning from Examples: Instructional Principles from the Worked Examples Research. Review of Educational Research, 70 (2): 181–214. http://dx.doi.org/10.3102/00346543070002181
2. Beck, K. (2003). Test-driven development by example. Addison-Wesley. ISBN: 978-0321146533
3. Brockmans, S., Volz, R., Eberhart, A., Löffler, P. (2004). Visual Modeling of OWL DL Ontologies Using UML. In Proceedings of ISWC 2004: 7–11. http://dx.doi.org/10.1007/978-3-540-30475-3_15

4. Chen, P. P. (1974). The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems, 1 (1): 9–36. http://dx.doi.org/10.1145/320434.320440
5. De Nicola, A., Missikoff, M. (2016). A Lightweight Methodology for Rapid Ontology Engineering. Communications of the ACM, 59 (3): 79–86. http://dx.doi.org/10.1145/2818359
6. Falco, R., Gangemi, A., Peroni, S., Vitali, F. (2014). Modelling OWL ontologies with Graffoo. In The Semantic Web: ESWC 2014 Satellite Events: 320–325. http://dx.doi.org/10.1007/978-3-319-11955-7_42
7. Fernandez, M., Gomez-Perez, A., Juristo, N. (1997). METHONTOLOGY: from Ontological Art towards Ontological Engineering. In Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering: 33–40. http://aaaipress.org/Papers/Symposia/Spring/1997/SS-97-06/SS97-06-005.pdf
8. Gangemi, A., Peroni, S. (2013). DiTTO: Diagrams Transformation inTo OWL. In Proceedings of the ISWC 2013 Posters & Demonstrations Track. http://ceur-ws.org/Vol-1035/iswc2013_demo_2.pdf
9. Keet M., Lawrynowicz. A. (2016). Test-Driven Development of Ontologies. In Proceedings of ESWC 2016: 642-657. DOI: http://dx.doi.org/10.1007/978-3-319-34129-3_39
10. Lewis, J. R., Sauro, J. (2009). The Factor Structure of the System Usability Scale. In Proceedings of HCD 2009. http://dx.doi.org/10.1007/978-3-642-02806-9_12
11. Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63 (2): 81–97. http://dx.doi.org/10.1037/h0043158
12. Peroni, S. (2014). The Semantic Publishing and Referencing Ontologies. In Semantic Web Technologies and Legal Scholarly Publishing: 121-193. http://dx.doi.org/10.1007/978-3-319-04777-5_5
13. Peroni, S. (2016). SAMOD: an agile methodology for the development of ontologies. figshare. http://dx.doi.org/10.6084/m9.figshare.3189769
14. Peroni, S., Shotton, D., Vitali, F. (2012). The Live OWL Documentation Environment: a tool for the automatic generation of ontology documentation. In Proceedings of EKAW 2012: 398–412. http://dx.doi.org/10.1007/978-3-642-33876-2_35
15. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E. (2009). eXtreme Design with Content Ontology Design Patterns. In Proceedings of WOP 2009. http://ceur-ws.org/Vol-516/pap21.pdf
16. Sauro, J. (2011). A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices. ISBN: 978-1461062707
17. Strauss, A. Corbin, J. (1998). Basics of Qualitative Research Techniques and Procedures for Developing Grounded Theory (2nd edition). Sage Publications: London. ISBN: 978-0803959408
18. Uschold, M., Gruninger, M. (1996). Ontologies: Principles, methods and applications. IEEE Intelligent Systems, 11 (2): 93-155. http://dx.doi.org/10.1109/MIS.2002.999223
19. Uschold, M., King, M. (1995). Towards a Methodology for Building Ontologies. In Workshop on Basic Ontological Issues in Knowledge Sharing. http://www.aiai.ed.ac.uk/publications/documents/1995/95-ont-ijcai95-ont-method.pdf
20. Vrandecic, D., Gangemi, A. (2006). Unit Tests for Ontologies. In OTM 2006 Workshops: 1012–1020. http://dx.doi.org/10.1007/11915072_2