DOI:10.1145/2818359

UPON Lite focuses on users, typically domain experts without ontology expertise, minimizing the role of ontology engineers.

BY ANTONIO DE NICOLA AND MICHELE MISSIKOFF

A Lightweight Methodology for Rapid Ontology Engineering

WE ARE LIVING in a reality that, thanks to economic globalization and the Internet, is increasingly interconnected and complex. There is thus a growing need for semantic technology solutions that can help us better understand it, particularly from a conceptual point of view. Ontologies represent an essential component to developing the Web of Data (such as Linked Open Data[1]) and Semantic Web applications. An ontology is a conceptual model of (a fragment of) an observed reality; it is, in essence, a repository of interlinked concepts pertaining to a given application domain. Traditionally, construction of an ontology (and its constant evolution, necessary to keep it aligned with reality) is lengthy and costly.

A high-quality ontology requires a rigorous, systematic engineering approach. Existing methodologies are quite complex, conceived primarily for skilled on-

» key insights

- **Ontology engineering helps domain experts better understand their own business reality by systematically exploring and describing it.**
- **The lightweight UPON Lite methodology for developing enterprise ontologies gives end users and business experts a central role, substantially reducing the need for ontology engineers.**
- **Ontology engineering is difficult for non-ontology specialists like business experts, but UPON Lite helps them develop ontologies without ontology engineers until the final step of formalization.**

tology engineers trained to develop large, industrial-strength ontologies. However, before embarking on a full-scale ontology project, it is useful to pursue pilot projects with experimental implementations, testing the applicability of semantic technologies in a confined enterprise area. From this perspective, available ontology engineering methodologies are often unsuitable, overly complex, and demanding in terms of time, cost, and skilled human resources.

There is thus a growing need for simpler, easy-to-use methods for ontology building and maintenance, conceived and designed for end users (such as domain experts, stakeholders, and even casual users in the relevant business domain), reducing the role of (and dependence on) ontology engineers. The objective is to shift responsibility for ontology building toward a community of end users through a social, highly participative approach supported by an easy-to-use method and tools.

We propose a simple, agile ontology engineering method intended to place end users at the center of the process. The proposed method, derived from the full-fledged Unified Process for ONtology building (UPON) methodology,[7] guarantees a rigorous, systematic approach but also reflects an intuitive nature. This method, or UPON Lite (to reflects its origin in its name), is conceived for a wide base of users (typically domain experts) without specific ontology expertise. UPON Lite is organized as an ordered set of steps, each releasing a self-contained artifact readily available to end users. Moreover, it is progressive and differential, with each new step using the outcome of the preceding step, providing well-defined enrichment to it.

The UPON Lite methodology contributes significantly to the "disintermediation" of ontology engineers. Before UPON Lite, it was necessary to assign the work of developing an ontology to a joint team of ontology engineers and domain experts; the latter bring knowledge of the domain, and the former are in charge of the ontology building process, following a rigorous method and notation. With UPON Lite, an ontology can be constructed largely by domain experts (along with end users) without ontology engineers. Only in the last step—once the domain content is elicited, organized, and validated—do ontology engineers intervene to deliver final ontology formalization before releasing it to users. We are confident there will soon be effective tools supporting domain experts (also in the last step), transforming semi-formal ontological knowledge into a formal ontological encoding.

UPON Lite is based on three main pillars of ontology development: a user-centered approach emphasizing the role of domain experts; a social approach leveraging the collective intelligence of domain experts, working to progressively achieve the steps in the method; and an ontology-building process articulated in six well-defined steps, each producing readily usable output.

The rest of this article explores the six steps of UPON Lite, the essence of the social approach, and a description of each step, showing the end user's role in progressively enriching the ontology base. It then covers related work and a comparative evaluation of the method. The final section draws a number of conclusions.

Figure 1. Sequence of steps in UPON Lite.

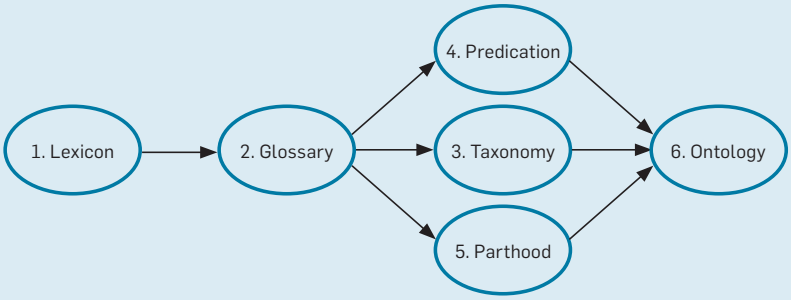


Table 1. Excerpt from a purchasing lexicon.

Address	Postal address	Request for quote
Customer	Price	RFQ
Delivery address	Purchasing conditions	Supplier
Invoice	Purchase order	Unit price
PO	Request for quotation	

Six Steps to Light Ontology Building

UPON Lite is organized as a sequence of steps, where the outcome of each one is enriched and refined in the succeeding step; the steps produce the following outcomes (see also Figure 1):

Step 1. Domain terminology. The domain lexicon listing the domain terms that characterize the observed domain;

Step 2. Domain glossary. The terms of the lexicon associated with a textual description, indicating also possible synonyms;

Step 3. Taxonomy.[11] Domain terms organized in a generalization/specialization (ISA) hierarchy;

Step 4. Predication.[17] Terms representing properties from the glossary identified and connected to the entities they characterize;

Step 5. Parthood (meronymy).[9] Complex entity names connected to their components, with all names needing to be present in the glossary; and

Step 6. Ontology. This last step produces the formally encoded ontology by using, say, the Web Ontology Language, or OWL, containing the conceptual knowledge collected in the five previous steps.

While this step numbering suggests a sort of sequencing, Figure 1 outlines the dependence among the different steps. In particular, there is no inherent dependency among intermediate steps 3, 4, and 5, as they can be performed in parallel. Moreover, the ontology-building process lacks a simple, linear progression, and the n^{th} step also provides feedback to improve the previous steps; to improve legibility, the figure omits backward arrows. Finally, depending on context and business objectives, users can skip one or two

intermediate steps. For instance, if interested in relational database design, they can concentrate on step 4, skipping step 3 and step 5, representing the rest of the knowledge as relational attributes; if interested in developing a product lifecycle management solution, they can also focus on step 5.

Before detailing the steps, we first explore the social approach of UPON Lite.

A Social Approach to Rapid Ontology Engineering

The traditional responsibility of an ontology-building project is given to a team of ontology engineers working with domain experts. However, it involves serious limitations as to the diffusion of ontologies and, more generally, semantic technologies, for several reasons. First is the shortage of ontology engineers with specialized technical competencies not generally available in the job market; second, ontology engineers, no matter how experienced, are seldom able to take in all relevant aspects of the application domain and, when an ontology is first released, there is always a need for domain-driven corrections, integrations, and extensions. Related to this need, and as the ontology is a sort of "conceptual image" of reality, even a "perfect" ontology must be maintained over time and, following the direction of domain experts, periodically realigned with the ever-changing world.

The idea of a "closed" team, no matter how articulate and skilled its members, can hardly respond to the indicated needs of the ontology to be developed. Conversely, the extensive involvement of users and stakeholders[15] is indeed the optimal solution. Users thus need to proceed along three lines:

Adopt simple tools. Simple tools for conceptual-modeling activities shield stakeholders, including domain experts and end users, from the intricacy and technical details of semantic technologies;

Open boundaries. The boundaries of an ontology team can be opened by adopting a social, participative approach to the collection, modeling, and validation of domain knowledge; and

Rethink the process. The ontology engineering process must be rethought to simplify the method, making it readily adoptable by non-ontology expert users (such as domain experts) and enforcing the methodological rigor necessary to guarantee the quality of the resulting ontology.

The objective is to shift responsibility for ontology building toward a community of end users through a social, highly participative approach supported by an easy-to-use method and tools.

There is also an organizational issue. Along with an enlarged, fluid organization for the ontology-engineering team comes a pivotal role for an ontology master with the expertise of an ontology engineer and the responsibility of monitoring and coordinating advancement of ontology-engineering activities.[2]

In UPON Lite, the whole ontology-building-and-management process is carried out through a socially oriented approach (in a transparent and participatory way) on a social-media platform. Here, we propose examples based on the Google Docs suite. In particular we experimented with shared Google Sheets for ontology engineering, plus Google Forms and Google+ for other functions (such as debating and voting on contentious issues).

The steps in UPON Lite are presented here through an example reflecting a (simplified) purchasing process, thus dealing with concepts like request for quotation, purchase order, and invoice. Purchasing is an activity that takes place in all business sectors; to make the example as general as possible, we thus concentrate on the business part, skipping the product-specific issues (such as the domain-specific goods or services to be purchased).

Step 1. Terminological Level

The first step in UPON Lite involves creating a domain-specific terminology, or list of terms characterizing the domain at hand. The outcome of this step is a domain lexicon, or information structure used to answer questions like "What are the characterizing words, nouns, verbs, and adjectives typically used while doing business in this domain? This is a preliminary step to start identifying domain knowledge and drawing the boundaries of the observed domain. The terminology in Table 1 is part of a purchasing lexicon. Note, at this level the terminology is a simple, flat, undifferentiated (with respect to, say, nouns and verbs) list of terms, including synonyms as separate entries. The basic rule for inclusion of terms in the list is the (statistical) evidence that a domain professional in the procurement sector would recognize each term as relevant.

In building a lexicon, users need not start from scratch. The Web makes it

Table 2. Glossary, including synonyms, kinds, and descriptions.

Term	Synonyms	Kind	Description [source]
Delivery address	Shipping address	Complex property	Location to which goods are to be sent.[1]
Invoice	Bill	Object	Itemized list of goods shipped, usually specifying the price and terms of sale.[2]
Postal address	Address	Complex property	Information that locates and identifies a specific address, as defined by the postal service.[3]
Purchasing conditions	Purchase terms and conditions	Object	Conditions related to the transaction and the trade.[4]
Purchase order	PO	Object	Commercial document issued by a buyer to a seller, indicating types, quantities, and agreed prices for products or services the seller will provide to the buyer.[5]
Customer	Client	Actor	One who purchases a commodity or service.[2]
Invoicing	Issuing invoice	Process	Making or issuing an invoice for goods or services.[6]
Purchasing	Buying	Process	Acquisition of something for payment.[6]

Sources: 1. http://glosbe.com; 2. http://www.merriam-webster.com; 3. http://www.ebxml.org; 4. http://docs.oasis-open.org/ubl/os-UBL-2.0/UBL-2.0.html; 5. http://www.wikipedia.org; 6. http://www.thefreedictionary.com

possible to find knowledge resources (such as textual documents, directories, dictionaries, taxonomies, standards, and ontologies) dealing with the business sector. If standards exist, it is worth taking advantage of them; for instance, business documents involve standards like the Universal Business Language[22] International Data Dictionary that may be useful in the ontology to be developed. Another method is extraction of the terminology from reference textual documents (such as manuals, textbooks, and whitepapers). A number of natural language processing tools are available for extracting terminology from text, including AlchemyAPI[23] and Open Calais,[24] which are readily available through a Web interface, and Term Extractor,[14] which is able to analyze a domain corpus and identify the terminology.

Challenges. The main challenge is deciding if a term is relevant and thus to identify the boundaries of the target application domain to include in the lexicon only the appropriate terminology. Identifying this boundary is not easy, since in nature there are no domain boundaries, and all is connected. Users must then consider two key dimensions: Along the "horizontal," what is the scope of the sought ontology?; imagine you are defining an ontology in a medical domain, how much of the technology domain (such as associated with electro-medical devices) should be included? There are two main strategies to defining the related scope: enriching the domain ontology with "foreign" terms or, if a "foreign" ontology is available, creating appropriate links to it. Along the "vertical" is the problem of the right level of detail, or granularity, that should be considered—not too much to avoid ineffective overloading, not too little to avoid critical omissions.

Another important challenge concerns the initial resources to be used as references. We mentioned there are plenty of terminological resources for all possible application domains. The problem is how users should select the ones they consider most representative. Domain corpora are important, but, despite the growing reliability and effectiveness of term extractors, post-processing of the extracted terms requires intervention of domain experts to produce a first list of relevant terms.

Social validation is another key challenge. While users have no problem publishing a particular lexicon online, the challenge they have concerns the method they choose for open consultation. There are a large number of off-the-shelf solutions for a deliberative online consultation. Here, users can adopt the simple method: "Like/Do Not Like" voting, since we find social participation decreases when people are asked to vote with more alternatives. Furthermore, our research suggests people, especially with a large lexicon, do not get through the whole list, tending instead to quickly browse the list and stop at terms they find objectionable.[21] An effective approach considers the listed terms as accepted if not explicitly rejected. Therefore, domain experts voting on relevant terms have only one option: "Do Not Like." Terms with a high number of rejections (above a given threshold) are removed from the lexicon. A richer method of social participation could include the option of proposing new terms and more sophisticated voting methods; a vast literature is available, starting with Parveen et al.[12]

Step 2. Glossary Level

Having produced a first lexicon, users could, in this step, enrich it by associating a textual description with each entry. This is critical; in fact, there are terms with a well-defined and widely accepted meaning, even defined by regulations and laws (such as those that apply to invoices), but there are also widely used terms that may have a different meaning in different business situations. For instance, what is a "delayed payment"? and How many days must elapse to classify a payment as "delayed"? Furthermore, a good engineering practice is not to "invent" descriptions but import them from authoritative sources.

Besides descriptions, users can start to add extra bits of semantics in this step. To this end, we adopt a method that uses the conceptual categories of the Object, Process, Actor modeling Language, or OPAL,[5] an ontology-structuring method that groups the concepts into three main categories—object, process, and actor—plus three auxiliary categories—complex, atomic, and reference properties. The actor category gathers active entities of a business domain, able to activate, perform, or monitor a process. The object category gathers passive entities on which a process operates. The process category gathers activities and operations aimed at helping achieve a business goal. We refer to such categories as "kinds" as a first semantic tagging of the terms representing the domain concepts.

Finally, having the description of

terms, it is easy to identify the synonyms. In identifying synonyms it is necessary to pinpoint the "preferred term" and label the others as synonyms (see Table 2).

Challenges. Users often find contradictory descriptions or descriptions pertaining to different points of view, or different roles in the enterprise, as when, say, an accounting department describes "inventory" differently from a stock management department. In case of multiple descriptions, users can create a synthesis or, according to the objective and scope of the ontology, privilege one over the other. This decision is typically left to the ontology master or to the "wisdom of the crowd"; the glossary is therefore first published with terms having more than one description, leaving it to the social-validation phase to converge toward a unique term description.

Another challenge is related to synonyms that require deciding what is the "preferred term." Voting, in this case, is a good way to achieve the result.

Step 3. Taxonomy Level

The first two knowledge levels reflect a terminological nature and exhibit a simple organization, a list of entries organized in alphabetical order. But the concepts denoted by the listed terms hide a rich conceptual organization users intend to represent through three different hierarchies. The first is a taxonomy based on the specialization relation, or the ISA relationship connecting a more specific concept to a more general one (such as invoice ISA business document). A taxonomy represents the backbone of an ontology, and its construction can be a challenge. It requires a good level of domain expertise and a consistent knowledge-modeling effort, since users must not only identify ISA relations between existing terms but also introduce more abstract terms or generic concepts seldom used in everyday life but that are extremely useful in organizing knowledge. During this step users thus provide feedback to the two previous knowledge levels—lexicon and glossary—since taxonomy building is also an opportunity to validate the two previous levels and extend them with new terms.

The example outlined in Table 3 is based on the use of a spreadsheet, where the specialization levels are organized, from left to right, in different columns.

Challenges. Defining a good taxonomy is difficult. Also difficult is organizing a flat list of concept names, or glossary terms, into a taxonomy. Care must be taken in considering different perspectives and opinions. The basic mechanism consists of the clustering of concepts, or terms, linking them to a more general concept (the bottom-up approach). Identifying a general concept is often not easy, and concepts can be clustered in different ways; in our simplified approach we avoid multiple generalization for a concept. Moreover, users must find a good balance between the breadth of the taxonomy, or average number of children of intermediate nodes, and its depth, or levels of specialization and the granularity of taxonomy leaves.

The ontology master plays an important role here, supported by numerous available resources (such as WordNet[25] and EuroVoc[26]).

The UPON Lite approach involves three disjoint sub-hierarchies, one for each OPAL kind. Therefore, when users specialize a concept, as in, say, an object, its more specific concepts cannot likewise become an actor or a process.

For these challenges, a social approach is highly advisable, along the lines of a folksonomy.[13,20]

Step 4. Predication Level

This step is similar to a database design activity, as it concentrates on the properties that, in the domain at hand, characterize the relevant entities. Users generally identify atomic properties (AP) and complex properties (CP). The former can be seen as printable data fields (such as `unit price`), and the latter exhibit an internal structure and have components (such as `address` composed of, say, `street`, `city`, `postal code`, and `state`). Finally, if a property refers to other entities (such as a customer referred to in an `invoice`) it is called a reference property (RP). In a relational database, an RP is represented by a foreign key. The resulting predicate hierarchy is organized with the entity at the top, then a property hierarchy below it, where nodes are tagged with CP, AP, and RP.

Continuing to use a spreadsheet, a user would build a table (see Table 4) where the first column reports the entities and the second the property name. In case of CP, the following columns on the right report the property components; in case of RP the referred entities are reported. Further information (such as cardinality constraints) can be added; for example, one invoice is sent to one and only one customer, who in turn may receive several invoices.

Challenges. Several decisions must be made in this step, starting with the granularity in representing properties. For instance, `address` can be a complex property, as covered earlier in this article, or can be an AP, where the

Table 3. Taxonomy excerpt with three specialization levels.

Top concept	First-level specialization	Second-level specialization
Business document	Invoice	
	Payment	Delayed payment
	Purchase order	
	Request for quotation	
Customer	Golden customer	
	Silver customer	

Table 4. Excerpt from a predication hierarchy.

Entity	Property	Sub-Property/Reference	Typing	Constraints
Invoice				
	Unit price [AP]		Currency value	
	Address [CP]	Street and number, city, state, Zip Code		
	Consignee [RP]	Customer		(1..1)
Customer				
	Name		String	
	Pending invoices	Invoice		(0..N)
	...			

whole address is encoded as one string. Likewise, an RP can be substituted by an AP if users adopt a "relational database" approach, viewing the property as the foreign key of the referenced entity (such as `customer` can be represented by `customer_code`). Other important points are represented by the typing of the AP (such as `string`, `integer`, and `Boolean`) and the cardinality constraints, or how many values a property can assume. Since UPON Lite is mainly for domain experts, typing may be too technical, such decisions can be delayed to a successive refinement cycle (mainly delegated to ontology engineers).

Step 5. Parthood Level

This step concentrates on the "architectural" structure of business entities, or parts of composite entities, whether objects, processes, or actors, by eliciting their decomposition hierarchy (or part-whole hierarchy). To this end a user would analyze the structure and components an entity exhibits, creating the hierarchy based on the `partOf` (inverse `hasPart`) relationship.

This hierarchy is particularly important in engineering and manufacturing and, more generally, when dealing with complex entities. For instance, a bill of material is a hierarchical decomposition of a product into its parts, subparts, and so on, until reaching elementary components, not further decomposable, representing the leaves of the decomposition tree, or, more precisely, a directed graph, generally acyclic.

Parthood can also be applied to immaterial entities (such as a regulation subdivided into sections and articles or a process subdivided into sub-processes and activities). In sub-processes and activities, users can enrich the model with other relations (such as "precedence" and "sync"), a subject beyond the scope of this article.

Challenges. In certain cases, users may have difficulty deciding if a hierarchical relation is `ISA` or `PartOf`. If we consider `price` and then `unit price`, it is not evident if `unit price` is a special case of the former or part of it. Such a relationship is highly dependent on business and organizational choices. At an abstract level, we may say `ISA` is more suited, but if we have, say, an invoice where price means final price and is obtained through multiplying a quantity (number of pieces) by unit price, then the latter is indeed a component of the former. Another problem for users is to distinguish a part from a property. For instance, an invoice has a footer that may report the tax and the final price. From a structural point of view (such as when printing an invoice) the footer is considered a component of the invoice, but from the information point of view it can be considered a CP holding structured information. In general, usage context determines the relationship, and, eventually, social validation provides the final interpretation, with a central role for the ontology master.

Step 6. Ontology Level

Using the outcomes of the previous five steps, ontology engineers can proceed to build the final artifact—the ontology. It gathers all the knowledge collected in those steps, in particular the concepts in Step 2 and the semantic relations elicited in three steps: a taxonomy (Step 3), further enriched by the predication (Step 4), and parthood (Step 5), as required. In addition, it is also possible to formally represent constraints (such as typing and cardinality constraints) and the needed domain-specific relations (such as `provides(Supplier, Product)`). Many of these relations can be obtained through "upgrading" the RP introduced in Step 4. To continue with the tabular approach, domain relations are represented in a new table with three columns, with an implicit orientation left to right (see Table 5).

The final step in implementing a "full-fledged" ontology is encoding it through a formal language like the Resource Description Framework (RDF) and OWL.[10] This is a technical task that requires the direct intervention of ontology engineers. However, a number of technical innovations aim to facilitate it to further increase end-user involvement. For instance, the semantic suite Anzo Enterprise (from Cambridge Semantics)[27] provides a plug-in for Excel that enables Excel spreadsheets to be mapped to an ontology, transforming the related data into RDF. Also along these lines is support (though less extensive) provided for the popular Protégé ontology system by the MappingMaster OWL plugin.[28]

A crucial aspect of the developed ontology is its quality, which is evaluated according to some predefined criteria. UPON Lite adheres to such an approach based on four perspectives: syntactic, semantic, pragmatic, and social.[3] In UPON Lite the syntactic and social quality of the produced ontology is addressed through the stepwise approach and social collaboration in knowledge encoding based on a shared online spreadsheet. Semantic quality concerns the absence of contradictions, a property that can be checked through specially designed software (such as the reasoners available with Protégé, including RacerPro,[29] Pellet,[30] and FaCT++[31]), once the ontology is imported through the MappingMaster plugin. The pragmatic quality of the ontology is guaranteed through the close involvement of end users in the whole ontology-building process.

Challenges. The first challenge concerns the choice of how expressive ontology engineers want the ontology to be. A minimal option is to include the taxonomy and the reference properties in the form of domain-specific relations. They can also add a number of constraints (such as typing and cardinality constraints). The second challenge concerns encoding the ontology in formal terms. Here, the role of the ontology engineer is central due to the high level of expertise required by this step.

Table 5. Domain-specific relations (excerpt).

Entity	Relation	Entity
Supplier	provides	Product
Invoice	paidBy	Client
Product	providedBy	Supplier
...	...	...

Related Work and Evaluation

Among the relevant methodologies of ontology engineering in the literature (such as Methontology,[8] On-To-Knowledge,[18] and DILIGENT[19]), none was expressly conceived for rapid ontology prototyping; each has a dif-

ferent purpose and scope, aiming at development of industrial-strength ontologies. For comparative studies of relevant ontology-engineering methods see De Nicola et al.[7] and Chimienti et al.[4] In 2013, GOSPL[6] was proposed as a collaborative ontology-engineering methodology aimed at building hybrid ontologies, or carrying informal and formal concepts. Finally, the NeON methodology[16] was conceived for developing ontology networks, introducing two different ontology-network-life-cycle models—Waterfall and Iterative-Incremental. They are more sophisticated and complex than UPON Lite and designed for ontology engineers; only DILIGENT and GOSPL explicitly address collaboration between domain experts and ontology engineers.

UPON Lite has been developed over the past 15 years through constant experimentation in research and industrial projects, as well as in a number of university courses. Beyond experimental evaluation, we also carried out a comparative assessment against existing ontology-engineering methodologies, using an evaluation method conceived for rapid ontology engineering based on 10 key features:

Social and collaborative aspects of ontology development. Considering the extent social and collaborative processes are included in the methodology;

Domain expert orientation. Referring to the extent the methodology allows domain experts to build and maintain an ontology without support of ontology engineers;

Cost efficiency. Concerning the focus of the methodology on cost reduction;

Supporting tools. Referring to the extent the methodology suggests tools to ease ontology development;

Adaptability. Referring to whether the methodology is flexible enough to be adopted in different industrial applications;

Reusability. Referring to the extent the methodology considers the possibility of reusing existing resources;

Stepwise and cyclic approach. Representing how much the methodology is based on an incremental cyclic process, avoiding a rigid "waterfall" linear model;

Natural language. Referring to the extent the methodology uses natural languages resources, processing techniques, and tools;

Documentation. Concerning production of supporting documentation and the extent an intermediate artifact can be considered a valuable documentation; and

Organizational structure. Referring to the extent project management methods are included in the methodology.

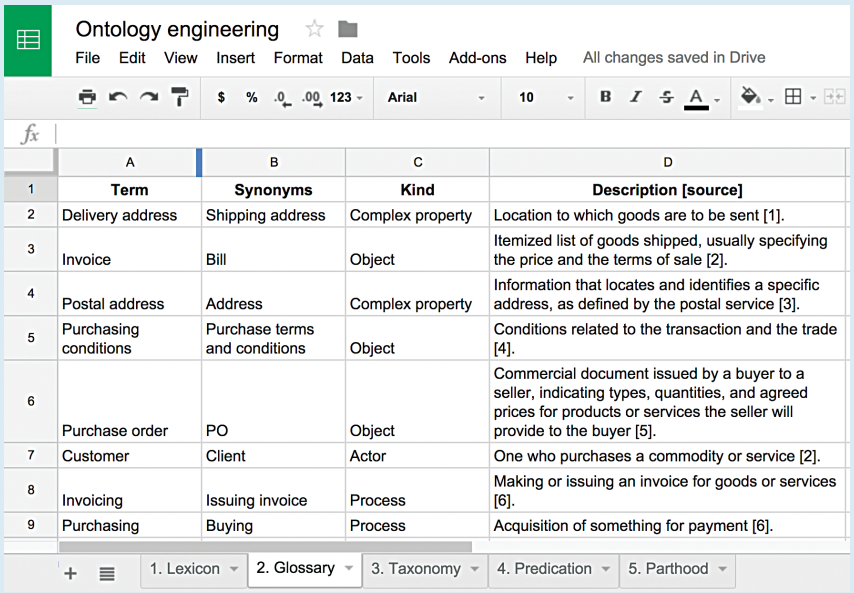
We defined them along the lines of other comparison methods in the literature, with an orientation toward rapid ontology engineering. Table 6 refers to the NeON methodology as NeONWf for the waterfall engineering model and NeONIn for the iterative-incremental model. The comparative evaluation for rapid ontology engineering shows UPON Lite and DILIGENT outperform the other methodologies, with a slight edge to UPON Lite.

Table 6. Ontology-engineering methodologies compared.

	(F.1)	(F.2)	(F.3)	(F.4)	(F.5)	(F.6)	(F.7)	(F8)	(F.9)	(F.10)	Ranking*
UPON Lite	H	H	H	H	H	H	H	H	M	M	9.3
Diligent	H	H	M	H	H	H	M	H	M	H	9.0
UPON	L	M	L	M	H	H	H	H	M	L	7.0
NeON$_{Wf}$	M	M	H	M	M	H	L	M	L	L	6.3
GOSPL	H	M	L	M	L	H	M	H	L	L	6.3
NeON$_{In}$	M	L	L	M	M	H	H	M	L	M	6.3
OntoKnowledge	L	L	L	M	H	M	H	L	L	H	6.0
Methontology	L	L	M	M	M	M	M	H	M	L	6.0

*Computed as normalized average, with H = 3, M = 2, and L = 1

Figure 2. Excerpt from a Google Docs UPON Lite spreadsheet.



	A	B	C	D
1	**Term**	**Synonyms**	**Kind**	**Description [source]**
2	Delivery address	Shipping address	Complex property	Location to which goods are to be sent [1].
3	Invoice	Bill	Object	Itemized list of goods shipped, usually specifying the price and the terms of sale [2].
4	Postal address	Address	Complex property	Information that locates and identifies a specific address, as defined by the postal service [3].
5	Purchasing conditions	Purchase terms and conditions	Object	Conditions related to the transaction and the trade [4].
6	Purchase order	PO	Object	Commercial document issued by a buyer to a seller, indicating types, quantities, and agreed prices for products or services the seller will provide to the buyer [5].
7	Customer	Client	Actor	One who purchases a commodity or service [2].
8	Invoicing	Issuing invoice	Process	Making or issuing an invoice for goods or services [6].
9	Purchasing	Buying	Process	Acquisition of something for payment [6].

Conclusion

We have proposed UPON Lite, an ontology-engineering methodology based on a lean, incremental process conceived to enhance the role of end users and domain experts without specific ontology-engineer expertise. Aiming to support rapid prototyping of trial ontologies, UPON Lite is a derivation of the full-fledged UPON Methodology. UPON Lite is characterized by three main aspects of ontology engineering: a user-centered approach conceived to be easily adopted by non-ontology experts, thus minimizing the role of ontology engineers; a socially oriented approach, where multiple stakeholders play a central role; and an intuitive ontology-

engineering process organized in six steps, supported by a "familiar" tool like a spreadsheet. Here, the spreadsheet tables are represented in cursory form; see Figure 2 for an actual excerpt from a Google Docs[32] spreadsheet we used in the running example.

UPON Lite has been used in industrial scenarios and university courses since 2001, producing more than 20 ontologies involving from a few domain experts up to 100, where non-ontology experts formed the great majority of ontology teams. The mean time of the ontology-building process varied from a week (in university courses) to a few months in industrial projects (not including maintenance). Since 2008, the methodology has been adopted by two European Union projects: Collaboration and Interoperability for Networked Enterprises in an Enlarged European Union (COIN) and Business Innovation in Virtual Enterprise Environments (BIVEE). COIN developed a trial ontology for the Andalusia Aeronautic cluster for a furniture ecosystem for the Technology Institute on Furniture, Wood, Packaging and related industries in Spain, and for the robotics sector, with Loccioni, in Italy. A national Italian project, E-Learning Semantico per l'Educazione continua nella medicina (ELSE), developed a trial ontology for lifelong education of medical doctors in the domain of osteoporosis. The feedback, collected through interviews and working meetings, covered various aspects of the methodology, from usability to efficiency and adoptability to flexibility; the results are encouraging. The field experiences in all these projects reflect the feasibility of stakeholders and end users producing good (trial) ontologies in a short time. Furthermore, the direct involvement of domain experts reduced the need for interaction with ontology engineers, as required by traditional methodologies, even for small ontology changes.

Involving communities of practice helps reduce the time and cost of rapid ontology prototyping. The UPON Lite stepwise approach has proved beneficial for the learning curve of domain experts new to the methodology, allowing them to quickly learn the process and its intuitive outcomes, including lexicon, glossary, taxonomy, predication, and parthood.

The UPON Lite approach advocates use of a social media platform and an online, shared spreadsheet as key tools; it also suggests other tools for supporting the methodology's first steps, including text mining, gloss extractors, and MappingMaster, can be used to automatically import spreadsheet content into an ontology editor (such as Protégé), thus producing an OWL ontology.

We performed a comparative evaluation of UPON Lite against the most popular ontology-engineering methodologies, as well as in the context of rapid development of a "lightweight" ontology. The results of the comparative assessment show UPON Lite offers the best solution.

In the future we intend to work on systematic monitoring of the adoption and use of UPON Lite in different domains, focusing on the problems that will emerge during the ontology-engineering process and how they should be addressed, considering the complexity of the ontology and the number of stakeholders involved.

Acknowledgment

This work is partially supported by European project BIVEE grant number FP7-ICT-285746.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. DBpedia: A nucleus for a Web of open data. In *Proceedings of the Sixth International Semantic Web Conference and Second Asian Semantic Web Conference* (Busan, Korea, Nov. 11–15). Springer-Verlag, Berlin, Heidelberg, Germany, 2007, 722–735.
2. Barbagallo, A., De Nicola, A., and Missikoff, M. eGovernment ontologies: Social participation in building and evolution. In *Proceedings of the 43rd Hawaii International Conference on System Sciences* (Honolulu, HI, Jan. 5–8). IEEE Computer Society, 2010, 1–10.
3. Burton-Jones, A., Storey V.C., Sugumaran, V., and Ahluwalia P. A semiotic metrics suite for assessing the quality of ontologies. *Data & Knowledge Engineering 55*, 1 (Oct. 2005), 84–102.
4. Chimienti, M., Dassisti, M., De Nicola, A., and Missikoff M. Evaluation of ontology building methodologies: A method based on balanced scorecards. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, J.L.G. Dietz, Ed. (Madeira, Portugal, Oct. 6–8). INSTICC Press, 2009, 141–146.
5. D'Antonio, F., Missikoff, M., and Taglino, F. Formalizing the OPAL eBusiness ontology design patterns with OWL. In *Proceedings of the Third International Conference on Interoperability for Enterprise Software and Applications* (Madeira, Portugal, Mar. 27–30). Springer, London, U.K., 2007, 345–356.
6. Debruyne, C., Tran, T.-K., and Meersman, R. Grounding ontologies with social processes and natural language. *Journal on Data Semantics 2*, 2-3 (June 2013), 89–118.
7. De Nicola, A., Missikoff, M., and Navigli, R. A software engineering approach to ontology building. *Information Systems 34*, 2 (Apr. 2009), 258–275.
8. Fernández-López, M., Gómez-Pérez, A., and Juristo, N. Methontology: From ontological art towards ontological engineering. In *Proceedings of the AAAI Spring Symposium Series* (Stanford, CA, Mar. 24–26). AAAI Press, Menlo Park, CA, 1997, 33-40.
9. Keet, C.M. and Artale, A. Representing and reasoning over taxonomy of part-whole relationships. *Applied Ontology 3*, 1-2 (Jan. 2008), 91–110.
10. McGuinness, D.L. and van Harmelen, F. *OWL 2 Web Ontology Language Overview. W3C Recommendation.* W3C, Cambridge, MA, 2003; http://www.w3.org/TR/owl2-overview/
11. Noy, N.F. and McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford University, Stanford, CA, 2001; http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html
12. Parveen, A., Habib, A., and Sarwar, S. Scope and limitation of electronic voting system. *International Journal of Computer Science and Mobile Computing 2*, 5 (May 2013), 123–128.
13. Pink, D.H. Folksonomy. *New York Times* (Dec. 11, 2005); http://www.nytimes.com/2005/12/11/magazine/11ideas1-21.html?_r=0
14. Sclano, F. and Velardi, P. TermExtractor: A Web application to learn the common terminology of interest groups and research communities. In *Proceedings of the Seventh Terminology and Artificial Intelligence Conference* (Sophia Antipolis, France, Oct. 8–9). Presses Universitaires de Grenoble, 2007, 85–94.
15. Spyns, P., Tang, Y., and Meersman, R. An ontology engineering methodology for DOGMA. *Applied Ontology 3*, 1-2 (Jan. 2008), 13–39.
16. Suárez-Figueroa, M.C., Gómez-Pérez, A., and Fernández-López, M. The NeOn methodology for ontology engineering. In *Ontology Engineering in a Networked World*. Springer, Berlin, Heidelberg, Germany, 2012, 9–34.
17. Sugumaran, V. and Storey, V.C. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. *ACM Transactions on Database Systems 31*, 3 (Sept. 2006), 1064–1094.
18. Sure, Y., Staab, S., and Studer, R. On-To-Knowledge Methodology (OTKM). In *Handbook on Ontologies*, S. Staab and R. Studer, Eds. Springer, Berlin, Heidelberg, Germany, 2004, 117–132.
19. Tempich, C., Simperl, E., Luczak, M., Studer, R., and Pinto, H.S. Argumentation-based ontology engineering. *IEEE Intelligent Systems 22*, 6 (Nov.-Dec. 2007), 52–59.
20. Van Damme, C., Hepp, M., and Siorpaes, K. FolksOntology: An integrated approach for turning folksonomies into ontologies. In *Proceedings of the Workshop Bridging the Gap Between Semantic Web and Web 2.0 of the Fourth European Semantic Web Conference* (Innsbruck, Austria, June 7), 2007, 57–70.
21. Velardi, P., Cucchiarelli, A., and Petit M. A taxonomy learning method and its application to characterize a scientific Web community. *IEEE Transactions on Knowledge and Data Engineering 19*, 2 (Feb. 2007), 180–191.

Web References

22. Universal Business Language; https://www.oasis-open.org/committees/ubl/
23. AlchemyAPI; http://www.alchemyapi.com/
24. Open Calais; http://www.opencalais.com/
25. WordNet; http://wordnet.princeton.edu/
26. EuroVoc; http://eurovoc.europa.eu/
27. Anzo; http://www.cambridgesemantics.com/
28. Protégé; http://protege.cim3.net/cgi-bin/wiki.pl?MappingMaster
29. RacerPro; http://franz.com/agraph/racer/
30. Pellet; http://clarkparsia.com/pellet/
31. FaCT++; http://owl.man.ac.uk/factplusplus/
32. Google Docs; https://docs.google.com/

Antonio De Nicola (antonio.denicola@enea.it) is a researcher in the Laboratory for the Analysis and Protection of Critical Infrastructures of the Smart Energy Division at the Italian National Agency for New Technologies, Energy, and Sustainable Economic Development, Rome, Italy, and a Ph.D. candidate in computer science, control, and geoinformation at the University of Tor Vergata, Rome, Italy.

Michele Missikoff (michele.missikoff@cnr.it) is an adjunct professor at the University of International Studies of Rome and an associate researcher at the Italian National Research Council, Semantic Technology Lab at the Institute for Cognitive Sciences and Technologies, Rome, Italy.