A Method for Defining Human-Machine Micro-Task Workflows for Gathering Legal Information

Nuno Luz[1], Nuno Silva[1], Paulo Novais[2]

[1] GECAD (Knowledge Engineering and Decision Support Group), Polytechnic of Porto
Porto, Portugal
{nmalu,nps}@isep.ipp.pt
[2] CCTC (Computer Science and Technology Center), University of Minho
Braga, Portugal
pjon@di.uminho.pt

Abstract. With the growing popularity of micro-task crowdsourcing platforms, new workflow-based micro-task crowdsourcing approaches are starting to emerge. Such workflows occur in legal, political and conflict resolution domains as well, presenting new challenges, namely in micro-task specification and human-machine interaction, which result mostly from the flow of unstructured data. Domain ontologies provide the structure and semantics required to describe the data flowing throughout the workflow in a way understandable to both humans and machines. This paper presents a method for the construction of micro-task workflows from legal domain ontologies. The method is currently being employed in the context of the UMCourt project in order to formulate information retrieval and conflict resolution workflows.

Keywords: Legal Crowdsourcing, Micro-Tasks, Workflows, Relational Law

1 Introduction

Several experiments in different domains have shown that micro-task crowdsourcing has great potential for solving large scale problems that are often difficult for computers to solve automatically, on their own [1]. These problems usually require a degree of creativity or just common sense plus some background knowledge [2, 3]. The interpretation and recognition of images and natural language are two examples of these kinds of problems.

Crowdsourcing platforms like Mechanical Turk, CloudCrowd, ShortTask and CrowdFlower are widely used for tasks such as (i) categorization and classification, (ii) data collection (e.g., finding a website address), (iii) moderation and tagging of images, (iv) surveys, (v) transcription from multimedia content (e.g., audio, video and images), and (vi) text translation.

More recently, a special interest in employing crowdsourcing towards solving complex tasks has emerged [4–9]. Following the trend of the current crowdsourcing platforms, which feature the execution of single micro-tasks, this interest has led to

the emergence of new approaches built upon workflows of micro-tasks. The modelling of such workflows allows the crowdsourcing of a new kind of more complex tasks (e.g., selecting and buying a video camera, recommending points of interest), which require the ordered execution of multiple types of micro-tasks.

Among these complex tasks are mediation processes often employed in relational law, which focuses on "justice produced through cooperative behavior, agreement, negotiation, or dialogue among actors in a post-conflict situation" [10]. The ordered execution of micro-tasks by individuals and groups selected from crowds not only results in cooperative solutions, but can also be used to implement conflict resolution and negotiation strategies in a wide scale. As a form of collective intelligence, the resulting data can be interpreted as a wide scale consensus or truth regarding a specific domain or topic, relevant to the law or case under scrutiny.

Micro-task workflows present new challenges at different dimensions of the crowdsourcing process, namely in micro-task specification and human-machine interaction [4, 5]. In particular, micro-task workflow approaches like CrowdForge [5], Jabberwocky [4] and Turkomatic [6] employ divide-and-conquer and map reduce strategies to build workflows. This usually involves workflows that include tasks for (i) the partitioning of the complex task (partition tasks), (ii) the execution of the partitioned tasks (map tasks), and (iii) the aggregation of results (reduce tasks).

However, in most cases, task (or micro-task) responses are unstructured and in natural language. Furthermore, micro-task interfaces are built using markup languages that contain little or no meta-data, making it difficult for machine micro-tasks to be included in the workflow.

The unstructured nature of micro-tasks in terms of domain representation makes it difficult (i) for task requesters not familiar with the crowdsourcing platform to build complex micro-task workflows and (ii) to include machine workers in the workflow execution process [11]. Furthermore, while some of the micro-tasks in the workflow are better performed by humans, others are better performed by a machine, which is seldom explicitly defined.

As stated by Obrst et al. [12], ontologies "represent the best answer to the demand for intelligent systems that operate closer to the human conceptual level". Domain ontologies are not only able to describe the domain knowledge, but also to describe workflow micro-tasks and the data flowing through them in a way understandable to both humans and machines.

Considering these, a method for the construction of human-machine micro-task workflow ontologies is proposed. Although the method is intended for the construction of crowdsourced micro-task workflows, it can be employed to build workflow ontologies for other types of applications. Possible domains of application include legal information retrieval and legal conflict resolution [13–16]. In the particular case of mediation in relational law, the essential requirements are (i) to harness structured and semantically enriched information (ii) from a crowd or group of actors. While current crowdsourcing approaches, like CrowdForge and Jabberwocky, tackle the distribution and crowdsourcing of micro-tasks, the resulting data is often found poorly structured or in natural language.

In this sense, the ultimate goal of this method is to define a set of ground rules for the assisted construction of workflow definition ontologies from domain ontologies. A top-level workflow definition ontology is presented, upon which any workflow execution and task distribution engine can be implemented. The resulting workflow definition ontology defines the domain and rules for each task, along with the relationships between the input and output data in and between tasks.

The following sections of this paper start with a brief overview of crowdsourcing terminology and ontology-related background knowledge. Section 3 describes the proposed workflow construction method in four parts: (i) domain ontologies, (ii) the Onto2Flow ontology, (iii) micro-task specification, and (iv) workflow specification. Finally, conclusions are given along with some remarks on the future directions of this work.

2 Background Knowledge

2.1 Micro-Task Workflows in Crowdsourcing

The terminology employed in the crowdsourcing domain often varies from platform to platform. In the context of this paper, a *job* (or a complex task) contains a workflow of *tasks* (or micro-tasks), along with all the data required for its execution. Micro-tasks (e.g., tag an image), as seen by the crowdsourcing community, have one or more *units* of work as input (e.g., the images to be tagged). Each of these units will be assigned to one or more *workers*, which must then submit a *response* (e.g., the tagging of the image). Multiple *assignments* of the same unit to different workers allow redundancy and quality improvements of the overall result after the aggregation of responses is performed.

Furthermore, the aggregation of responses often takes into account units for which a correct response is already known. These units are often referred to as *reference units*. Workers that give incorrect or invalid responses to reference units suffer credibility penalties, and their responses have significantly less impact in the final result.

Typically, in crowdsourcing platforms such as Mechanical Turk, human workers choose whether to perform the specific task according to the given (often monetary) *reward*. In some cases, the *requesters* of the task may require workers with certain expertise and *qualifications*, which are given after the worker successfully solves a qualification task.

Through the analysis of the evolution of crowdsourcing platforms, it is possible to conclude that an effort towards structured (sets of) tasks is being made. While early crowdsourcing platforms such as MTurk, CrowdFlower, MicroWorkers and CloudCrowd have added template construction features, more recent platforms and frameworks such as CrowdForge, Jabberwocky, Turkomatic and Turkit have tackled this emerging need through different workflow representations and construction strategies.

Table 1 presents a comparison of several crowdsourcing approaches. Each approach is compared according to five different dimensions. These dimensions reflect if the approach (i) relies on its own crowd or in multiple (possibly external) crowds, (ii) supports complex tasks, (iii) employs any task construction strategy, (iv) employs

worker and result assessment strategies, and (v) employs result aggregation strategies when redundancy (multiple responses for the same unit) is found.

Table 1. Comparison of crowdsourcing platforms.

System	Relies on	Complex Tasks	Task Strategy	Worker Assessment	Aggregation
MTurk	Self	No	Task Templates	Qualification Tests	Manual
CrowdFlower	Several	No	Task Templates	Gold Units	Yes
ShortTask	Self	No	Task Templates	Manual	Manual
MicroWorkers	Self	No	Task Templates	Manual	N/A
CloudCrowd	Self	-	-	Credential Tests and Credibility	-
CrowdForge	MTurk	Workflows	Map Reduce	(MTurks')	Yes
Jabberwocky	Self/Several	Workflows	Map Reduce	User Profiles	Yes
Turkomatic	MTurk	Workflows	Divide and Conquer	(MTurks')	Yes (Workers)
Turkit	MTurk	Workflows	Crash and Rerun	(MTurks')	Yes (Workers)

2.2 Ontologies in Description Logics

In this paper, Description Logics (DL) knowledge bases and ontologies with $\mathcal{ALCOQ}$ expressivity are considered (see fig. 1). A DL knowledge base contains a TBox (terminological box) and an ABox (assertion box) [17], where the TBox contains all the concepts and relationships that define a specific domain, and the ABox contains the instances or individuals defined according to the elements in the TBox. It is assumed that ontology is synonym of TBox.

Each concept (e.g., C, D) is defined according to other concepts (e.g., $C \sqcup D$) and property restrictions (e.g., $\exists R.D$) that define the necessary (e.g., $C \sqsubseteq \exists R.D$), and necessary and sufficient (e.g., $C \equiv \exists R.D$) conditions for an individual to be an instance of the concept.

There are two main types of properties: object properties and data-type properties. While object properties relate instances (or individuals) with other instances, data-type properties relate instances with "primitive" type values (e.g., integer, string, double, date, time).

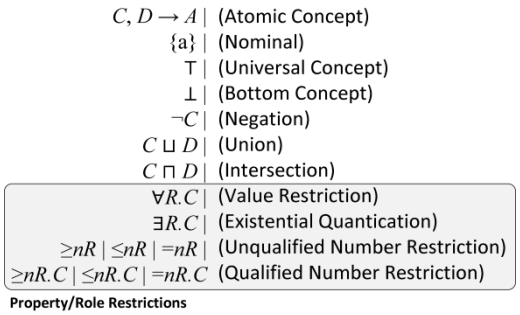

Fig. 1. TBox concept description syntax and rules with $\mathcal{ALCOQ}$ expressivity.

3 The Workflow Specification Method

Micro-tasks, whether they involve physical actions or not, can be seen as a process that, in a specific context, results in the emergence of new data (responses) from the presentation of other particular pieces of data (units) to a worker. Analogously, a workflow of micro-tasks is the continuous ordered increment of new (different types of) data, in a specific context or domain.

The proposed method suggests that micro-task responses correspond to new instances of concepts (or classes) in the domain ontology, associated with input (unit) instances of domain ontology concepts. Thereafter, a micro-task can be considered to be *the instantiation of domain classes and the specification of new relationships between instances* according to the domain ontology. A workflow of micro-tasks is then considered as *the incremental instantiation of the domain ontology according to its structure and semantics*.

With the assumption that domain ontologies represent the structure and semantics of the data that must be presented and retrieved from workers during the execution of a task, workflow ontologies extend both the Onto2Flow and domain ontologies (see fig. 2).

Workflow ontologies are instantiated and executed by a workflow engine that is able to interpret the ontology according to the ground rules established by the proposed method. During the workflow execution, the input is given as an ABox described by the domain ontology. The output of the workflow will be described by the domain ontology and, in some situations, operational concepts and properties of the workflow ontology.

The ground rules established by the proposed method must be employed during the workflow construction step (1) and followed during the instantiation and execution step (2).

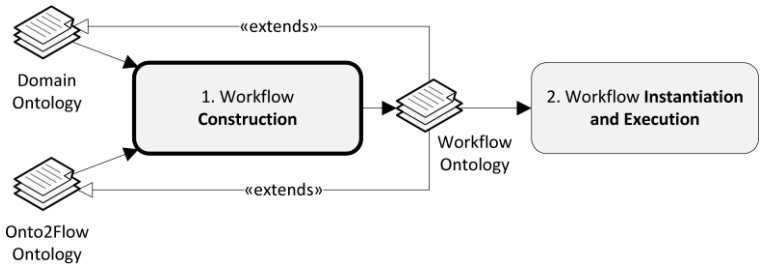

Fig. 2. Workflow construction and execution steps.

3.1 Domain Ontologies

Workflow ontologies capture the tasks/operations of a certain process, and the dynamic nature of a domain. The static structure and semantics of the specific knowledge domain, on the other hand, are captured by domain ontologies in the form of concepts and their relations.

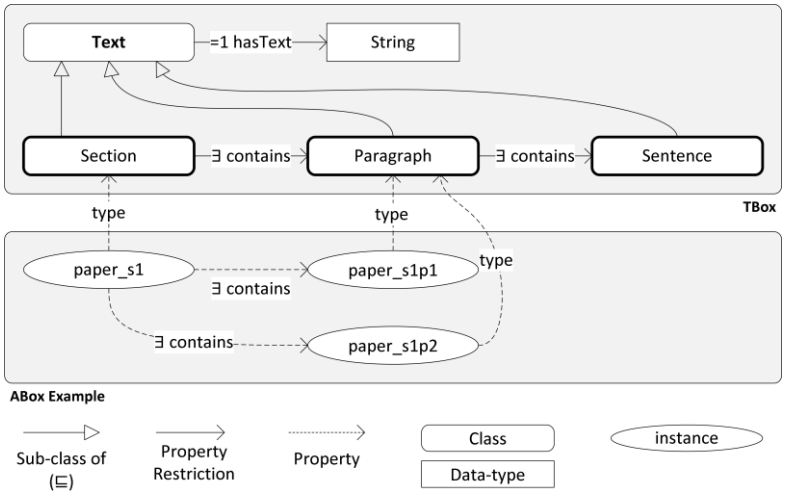

Fig. 3. The document ontology (TBox only) with a possible example ABox (or instantiation). The TBox is an adaptation from the DoCO (Document Components) ontology[1].

Unlike workflow ontologies, domain ontologies are very common and accessible. Inclusively, their structure can be analysed and employed in the construction of workflow ontologies.

Consider the document ontology and example ABox presented in fig. 3. The graph structure of the TBox defines the known properties of instances in the ABox. Following the restrictions specified in this structure, the incremental filling of the ABox is possible through the execution of several atomic operations (micro-tasks). In the spe-

[1] DoCO: http://purl.org/spar/doco/

cific case of the document ontology, an initial ABox with English sections may be supplied as input to the workflow, resulting in translated Portuguese sections. Since the ontology contains the semantics for the subdivision of sections, some of the workflow micro-tasks may consider their subdivision into smaller units (e.g., paragraphs, sentences).

Translation is a typical domain of application in crowdsourcing, however, the proposed method can be applied in other domains that may or may not be currently in the scope of crowdsourcing. A partial simplification of a possible legal ontology, depicted in fig. 4, describes such a domain. The concepts and relationships in this ontology can be used to establish workflows that inquire a crowd about past legal cases (e.g., abusive discharge cases) in order to gather information for new ones.

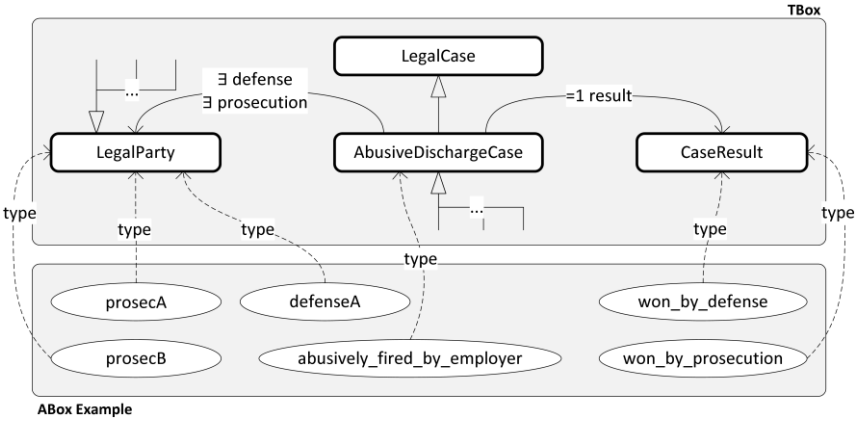

Fig. 4. A possible partial legal ontology (TBox only) with an example ABox (or instantiation).

3.2 The Onto2Flow Ontology

The Onto2Flow ontology captures the structure and semantics of workflows (see fig. 5). The main concepts are: Task, Assignment, Requester and Worker (either Machine or Person). This ontology is further extended and its concepts refined in the workflow ontology as required by the domain of application.

Assignments correspond to the execution of a task by a worker, for a single unit of work. The properties that define the domain of a task are:

- *unit* – defines the set of instances (class) that constitute the input of the task (only one property restriction allowed);
- *unitContext* – defines the input context classes of the task;
- *response* – defines the set of instances (class) that constitute the output of the task (only one property restriction allowed);
- *responseContext* – defines the output context classes of the task.

The different types of atomic operations (or micro-tasks) that can be performed are specified in the ontology through sub-classes of Task. As presented in fig. 5, the On-

to2Flow ontology currently defines four atomic operations associated with the classes: CreateAndFillTask, FillTask, SelectionTask and AggregationTask.

CreateAndFillTask instances will result in new instances of the response class, for which all data-type property values will be requested to the worker.

A FillTask will request data-type property values for already existent instances of the unit class.

SelectionTask instances will result in the definition of new relationships between already existent instances in the ABox, i.e., no new response instances will be created. Instead, they will be selected by the worker from a set of possible responses.

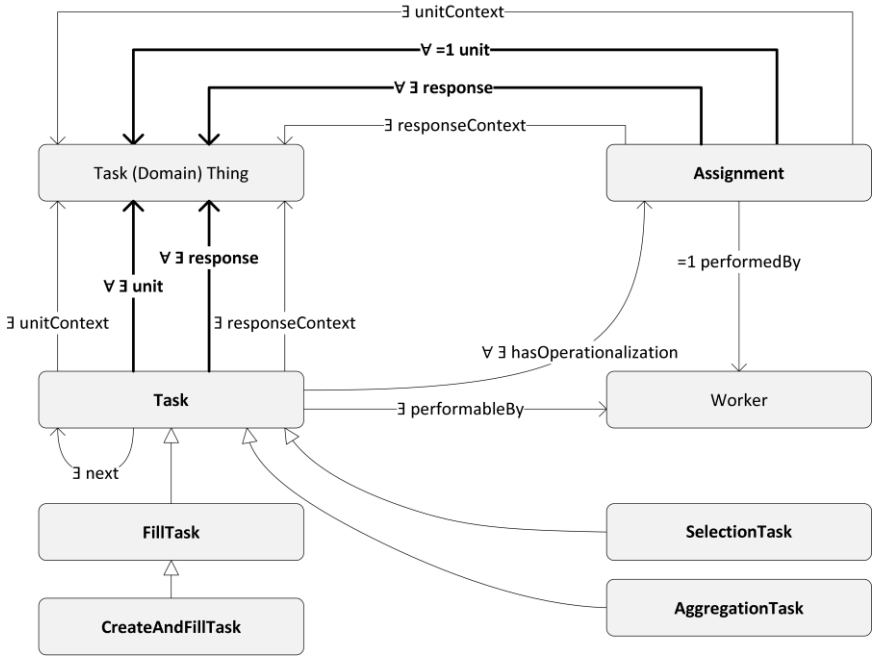

Fig. 5. Partial representation of the Onto2Flow ontology with Task sub-classes for atomic operations.

If more than one assignment per unit is demanded, the execution of the task will result in several possible Output ABoxes for each unit. In these situations, an aggregation of the responses must be performed through an AggregationTask. AggregationTasks consider the context, unit and response classes of the previous task. Furthermore, any number of AggregationTask sub-classes may be included in order to implement different aggregation strategies (e.g., majority voting, assessment-based).

Requesters may define the set of workers that may participate in the task through the *performableBy* property. In order to restrict or create worker roles, new Worker (Person or Machine) sub-classes may be created with restrictions applied to their properties (e.g., ∃*country*.*{portugal}*).

The *performedBy* property is established only after the worker accepts to participate in the task.

3.3 Defining Micro-Tasks

A workflow ontology describes a workflow that can be instantiated multiple times. The workflow ontology must import and extend the Onto2Flow ontology. Domain concepts must either be defined in the workflow ontology, or imported from a domain ontology (as depicted in fig. 2). The following parts of this document assume that domain concepts are always imported from an external domain ontology.

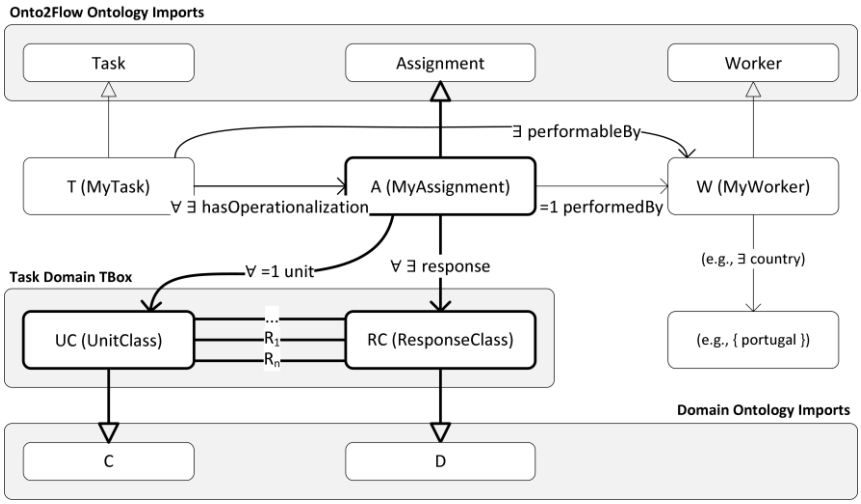

Fig. 6. Structure of a basic micro-task in a workflow. Relationships between *T* and *UC*/*RC* were omitted from the figure since they are similar to those between *A* and *UC*/*RC*.

In order to build the workflow ontology, the requester must extend the Task, Assignment and Worker classes from the Onto2Flow ontology, and any class from the domain ontology. Fig. 6 depicts the ontological structure of a simple micro-task.

A micro-task specification is an explicit partial TBox in the workflow ontology with, at least, the following terminological axioms:

- $T \sqsubseteq Task \sqcap \exists response.RC \sqcap \forall response.RC \sqcap \exists assignment.A$
- $A \sqsubseteq Assignment \sqcap \exists response.RC \sqcap \forall response.RC$
- $RC \sqsubseteq D$

The specification of an *UC* is not mandatory and is done through the following terminological axioms:

- $T \sqsubseteq \exists unit.UC \sqcap \forall unit.UC$
- $A \sqsubseteq= 1unit.UC \sqcap \forall unit.UC$
- $UC \sqsubseteq C$

- $UC \sqsubseteq \exists R.RC$ or $RC \sqsubseteq \exists R.UC$ (optional)

C and *D* are classes in the domain ontology. *UC* represents the subset of *C* instances that constitute the input of *T*. *RC* represents instances of *D*, which are output of *T*. If no additional property restrictions are defined on *UC*, any instance of *C* in the input ABox is also considered to be an instance of *UC*.

A establishes an n-ary relationship between *UC* and *RC*, which reflects the operational semantics of all *R*. *R* are object property restrictions (from properties and restrictions typically present in the domain ontology) that establish a direct correspondence between *UC* and *RC* (or vice-versa).

If the requester needs to select specific target workers for the task, a sub-class of Worker ($W \sqsubseteq Worker \sqcap C_1 \sqcap C_2 \sqcap ... \sqcap C_n$, where *C* represents a property restriction onto the *W* class) must be created.

The Task Domain TBox represents a partial copy of the domain ontology containing only the necessary classes and relationships: those required as input and those for which new instances and relationships will be established.

Unit Context Classes. In some situations, the requester needs to provide additional contextual information, given through related domain classes, to the worker. For these tasks, unit context classes (*UCC*s) may be specified. The set of all UC, UCCs, and their relationships form the Input TBox. The Input TBox defines the set of rules that will filter the input data from the given ABox. For instance, the following rule would filter the input of the task according to the Input TBox structure presented in fig. 7:

$$\forall x \forall y \big(D(x) \wedge C(y) \wedge S(x,y) \rightarrow UCC(x) \wedge UC(y) \big)$$

Any number of *UCCs* may be included in the Input TBox, with any type of relationships between them and to/from the *UC* or *RC*.

Relationships to/from the *RC* (e.g. *T*) are established during the execution of the task.

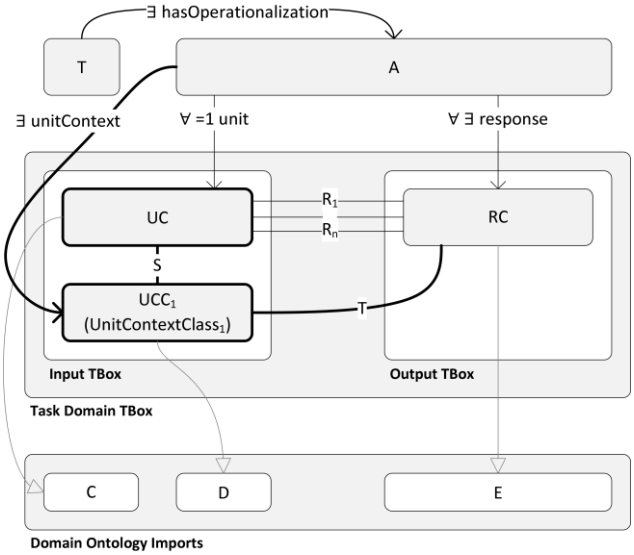

Fig. 7. Structure of a micro-task with unit context classes.

Response Context Classes. Response context classes (*RCC*s) establish property restrictions onto the *RC* that must be followed by the worker (as in fig. 8). An *RCC* represents a subset of input instances (it is a sub-class of an *UCC*) that were chosen by the worker as property values for an *RC* instance. The mandatory sub-class-of relationship between the *RCC* and the *UCC* is considered a dependency.

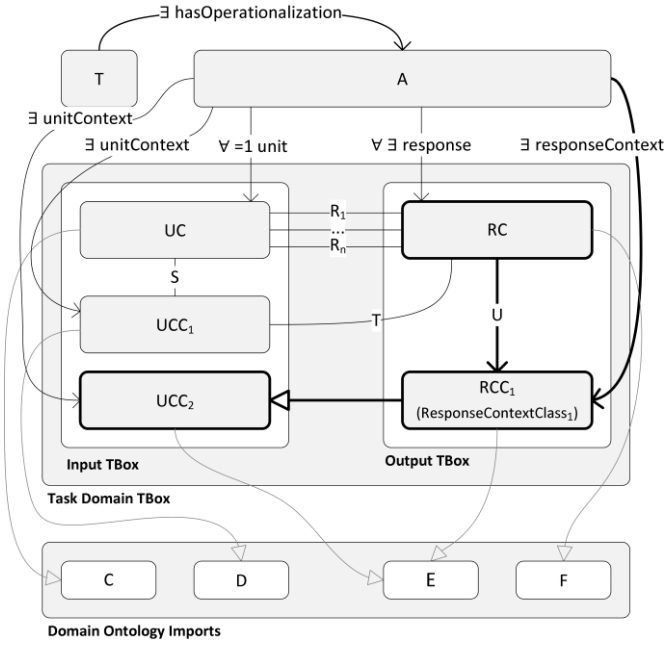

Fig. 8. Structure of a micro-task with response context classes for establishing *RC* property restrictions.

Dependency relationships indicate that instances of the *UCC* are candidate instances of the *RCC*. In this sense, the worker will have to select which instances of *UCC* will become instances of *RCC*, related to *RC* through the property in the specified restriction, *U*.

An *RCC* is defined through an *UCC* (where *E* is a class from the domain ontology) as:

- $RC \sqsubseteq \exists U.RCC$
- $RCC \sqsubseteq UCC \sqcap E$
- $UCC \sqsubseteq E$

Dependencies on the Response Class. When building SelectionTask and FillTask tasks, the requester must establish a dependency between the *RC* and one of the Input TBox classes (the *UC* or an *UCC*) (see fig. 9 for an example with a SelectionTask).

For SelectionTask tasks, the dependency must be established between the *RC* and an *UCC*. Analogous to paths established through RCCs, it means that the worker will have to select the appropriate *RC* instance(s) from the set of instances given by the *UCC*. The selected instance(s) will become the response of the assignment.

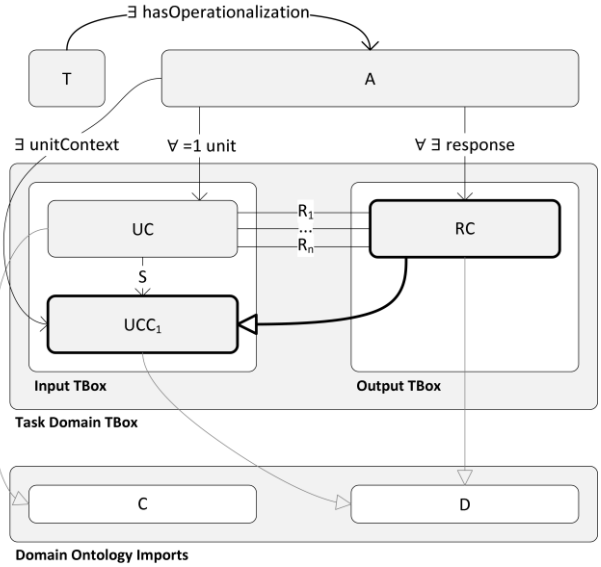

Fig. 9. Structure of a SelectionTask task with a dependency between the *RC* and an *UCC*.

An *RC* dependency for SelectionTasks is defined (where D is a class from the domain ontology) as:

- $RC \sqsubseteq UCC \sqcap D$
- $UCC \sqsubseteq D$

For FillTask tasks, the dependency can be established between the *RC* and either the *UC* or an *UCC*. It means that the worker will have to fill the data-type properties for existent instances of the *UC* or *UCC*.

Considering *IC* any class that may be the *UC* or an *UCC*, an *RC* dependency for FillTasks is defined (where *D* is a class from the domain ontology) as:

- $RC \sqsubseteq IC \sqcap D$
- $IC \sqsubseteq D$

3.4 Defining Workflows of Micro-Tasks

Workflows of micro-tasks are defined through dependency relationships between Task Domain TBoxes. A micro-task *A* is dependent (or follows) a micro-task *B* if there is at least one dependency relationship between the Input TBox of *A* and the Task Domain TBox of *B*.

Dependency relationships between micro-tasks are used to infer the *next* relationship and to optimize the resulting workflow. The optimization process includes the parallelization of independent tasks.

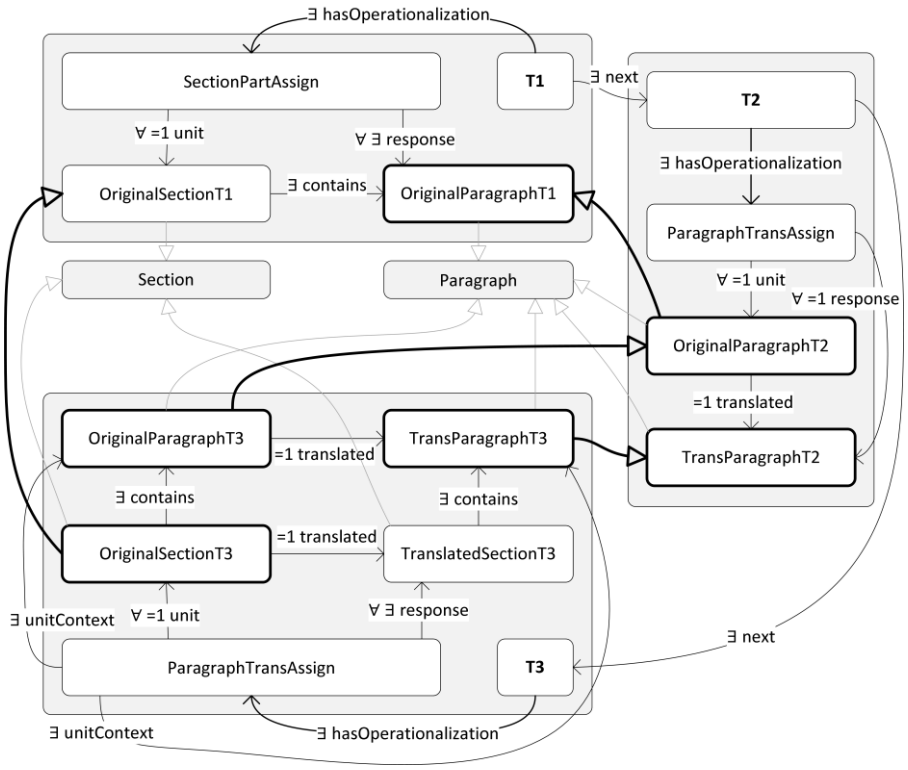

Fig. 10. Example of a CreateAndFillTask micro-task workflow built according to the translation ontology that (i) partitions sections into paragraph, (ii) translates paragraphs, and (iii) assembles paragraph translations into translated sections.

Fig. 10 depicts a section translation micro-task workflow that applies a divide-and-conquer strategy. The Section and Paragraph domain classes from the translation ontology, along with their relationships, are exploited in the workflow ontology in order to define each of the CreateAndFillTask tasks *T1*, *T2* and *T3*. Dependencies exist between *T2* and *T1*, between *T3* and *T2*, and between *T3* and *T1*. The transitive closure of the inter-task dependency relation results in the workflow structure reflected by the next relationship. In this case, the three tasks are executed in the sequential order: *T1*, *T2* and *T3*.

The legal information retrieval workflow, presented in fig. 11, is built from the partial legal ontology in fig. 4. It depicts a situation where an expert (e.g., a lawyer) is assessing the possibilities to take legal action against a company on behalf of a customer [18].

The first task, *T1*, is a CreateAndFillTask micro-task that asks an entity or crowd for instances of abusive discharge cases. For each given case, the worker(s) must also fill all datatype properties of the AbusiveDischargeCase concept. The second task, *T2*, is a CreateAndFillTask micro-task where the entity or crowd must, for each case previously submitted, provide information on the defence and prosecution parties in-

volved. Finally, on task *T3*, workers submit information on reported abuses for each case submitted in *T1*.

These types of information retrieval workflows allow legal parties to collect information on previous instances of legal procedures. The retrieved information is structured and enriched with the semantics of legal domain ontologies.

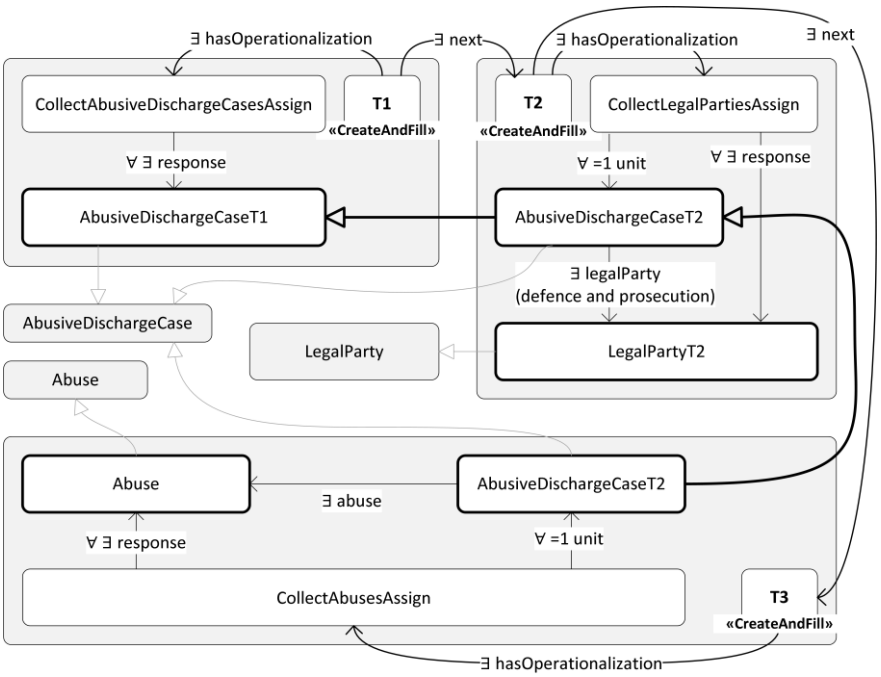

Fig. 11. Example of a micro-task workflow built according to a legal ontology that (i) asks for instances of abusive discharge cases, (ii) requests information about the legal parties involved in each case and (iii) asks for reported abuses in each case.

4 Conclusions and Future Work

The proposed method tackles the challenge of building micro-task workflows while promoting human-machine cooperation through high-level, declarative and semantically explicit domain ontology models.

Although the process of manually building micro-task workflows requires some degree of domain expertise and knowledge of the Onto2Flow ontology, the ground rules for creating an assisted workflow construction process were defined.

Since domain ontologies are interpretable by humans and machines, micro-tasks can be solved by either or both human and machine workers.

Future work includes the creation of an assisted micro-task workflow construction process, which automates the construction of workflows through the detection of ontology patterns and their aggregation into different strategies. The evolution of the

Onto2Flow ontology, in order to assimilate concepts often found in workflow definition languages (e.g., Gateway, Loop, Event), is also considered. Furthermore, the proposed method is being employed in the context of the UMCourt project [13, 14] in order to further evaluate its impact in legal use cases.

Acknowledgements. This work is part-funded by FEDER Funds, by the ERDF (European Regional Development Fund) through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within the project FCOMP-01-0124-FEDER-028980 (PTDC/EEI-SII/1386/2012). The work of Nuno Luz is supported by the doctoral grant SFRH/BD/70302/2010.

References

1. Von Ahn L (2009) Human Computation. 46th ACM IEEE Des. Autom. Conf. pp 418–419
2. Chklovski T (2003) Learner: A System for Acquiring Commonsense Knowledge by Analogy. Proc. 2nd ACM Int. Conf. Knowl. Capture. Sanibel Island, FL, USA, pp 4–12
3. Singh P, Lin T, Mueller ET, et al. (2002) Open Mind Common Sense: Knowledge Acquisition from the General Public. Move Meaningful Internet Syst. 2002 CoopIS DOA ODBASE. Springer, pp 1223–1237
4. Ahmad S, Battle A, Malkani Z, Kamvar S (2011) The Jabberwocky Programming Environment for Structured Social Computing. Proc. 24th Annu. ACM Symp. User Interface Softw. Technol. Santa Barbara, CA, USA, pp 53–64
5. Kittur A, Smus B, Khamkar S, Kraut RE (2011) Crowdforge: Crowdsourcing Complex Work. Proc. 24th Annu. ACM Symp. User Interface Softw. Technol. Santa Barbara, CA, USA, pp 43–52
6. Kulkarni AP, Can M, Hartmann B (2011) Turkomatic: Automatic Recursive Task and Workflow Design for Mechanical Turk. Proc. 2011 Annu. Conf. Ext. Abstr. Hum. Factors Comput. Syst. Vancouver, BC, Canada, pp 2053–2058
7. Little G, Chilton LB, Goldman M, Miller RC (2010) Turkit: Human Computation Algorithms on Mechanical Turk. Proc. 23rd Annu. ACM Symp. User Interface Softw. Technol. New York, NY, USA, pp 57–66
8. Luz N, Silva N, Maio P, Novais P (2012) Ontology Alignment through Argumentation. 2012 AAAI Spring Symp. Ser.
9. Sarasua C, Simperl E, Noy NF (2012) CrowdMap: Crowdsourcing Ontology Alignment with Microtasks. Semantic Web – ISWC 2012. Springer, pp 525–541
10. Casanovas P (2009) The Future of Law: Relational Justice, Web Services and Second-generation Semantic Web. Legal Information and Communication Technologies 7:137–156.
11. Quinn AJ, Bederson BB (2011) Human Computation: A Survey and Taxonomy of a Growing Field. Proc. SIGCHI Conf. Hum. Factors Comput. Syst. ACM, New York, NY, USA, pp 1403–1412
12. Obrst L, Liu H, Wray R (2003) Ontologies for Corporate Web Applications. AI Mag 24:49.
13. Carneiro D, Novais P, Andrade F, et al. (2013) Using Case-Based Reasoning and Principled Negotiation to provide decision support for dispute resolution. Knowl Inf Syst 36:789–826.

14. Novais P, Carneiro D, Gomes M, Neves J (2013) The relationship between stress and conflict handling style in an ODR environment. New Front. Artif. Intell. Springer, pp 125–140
15. Casanovas P (2012) Legal crowdsourcing and relational law: What the semantic web can do for legal education. J Australas Law Teach Assoc 5:159–176.
16. Poblet M, Casanovas P, Cobo JML, Casellas N (2011) ODR, Ontologies, and Web 2.0. J UCS 17:618–634.
17. Baader F, Calvanese D, McGuinness DL, et al. (2007) The Description Logic Handbook: Theory, Implementation, and Applications, 2nd ed. Cambridge University Press
18. Gangemi A, Presutti V, Blomqvist E (2011) The Computational Ontology Perspective: Design Patterns for Web Ontologies. Approaches Leg. Ontol. Springer, pp 201–217