Documentation – A crowdsourcing plugin for the validation of ontologies for the Protégé Ontology Editor

The crowdsourcing/GWAP plugin for the Protégé Ontology Editor allows users to validate parts of an ontology with the help of crowdsourcing – using the uComp Human Computation (HC) services. The uComp HC services rely on so-called Games With a Purpose (GWAP) and virtual labor markets such as CrowdFlower.

GWAPs are interactive games which are run as applications on Social Web platforms (such as Facebook) or as stand-alone applications. Players solve certain problems which cannot be solved by algorithms (with acceptable accuracy), GWAP users are motivated by scoring system or other forms of rewards, whereas users on virtual labor markets are paid for their input.

To use the plugin, users select the part of the ontology to validate and define some parameters (for details see below). The plugin sends the request to the HC API (uComp API), which delegates the task to a GWAP or CrowdFlower. After enough HC users given their input on the task, the results are sent back to the plugin, which then displays them to the Protégé user. Depending on the result, the user can perform further actions like deleting negatively validated parts of the ontology.

Installing the plugin

Step 1: Installing Protégé 4.3

To run the plugin, you need a recent version of Protégé installed. The current version (July 2014) is Protégé 4.3 (build 304), the download and installation instructions are found on http://protege.stanford.edu/download/protege/4.3/installanywhere/Web_Installers/.

The installer is available for various platforms and also with and without the Java VM. Since the version including the Java VM is not updated with every Java Update, it is recommended to install the latest Java VM separately and then download and install the Protégé version without Java. The latest Java VM is available at the Java website http://www.java.com/de/download/. We recommend to install the 32-bit version of Protégé. The plugin has been tested with Protégé 4.3 and 4.2.

Step2: Installing the Crowdsourcing plugin

Every Protégé plugin consists of a single jar-File (Java archive). All required third party libraries have to be included into that jar-File, but it is possible to reference to other installed Protégé plugins since Protégé is built very modular.

The Crowdsourcing plugin (current version is 1.1.1) can be downloaded from within Protégé. The relevant menu item for this functionality is found under *File → Check for plugins.*

Step3: Adding the uComp-API settings

To use the plugin, you need to provide some settings (uComp-API key, number of needed judgements per unit, payment in cents per unit if you want to publish them onto crowdflower) in a file named **ucomp_api_settings.txt** in the folder **.Protege**, which is automatically created by Protégé in your home-directory.

You have to provide the settings in one line in the following format:
<ucomp_api_key>,<number_of_judgements_per_unit>,<payment_in_cents>
(Example file contents of ucomp_api_settings.txt:

```
abcdefghijklmnopqrst,5,2
```

This means that per unit (evaluation task) 5 different judgments will be collected, and for each judgment 2 USD-cents are paid.
)

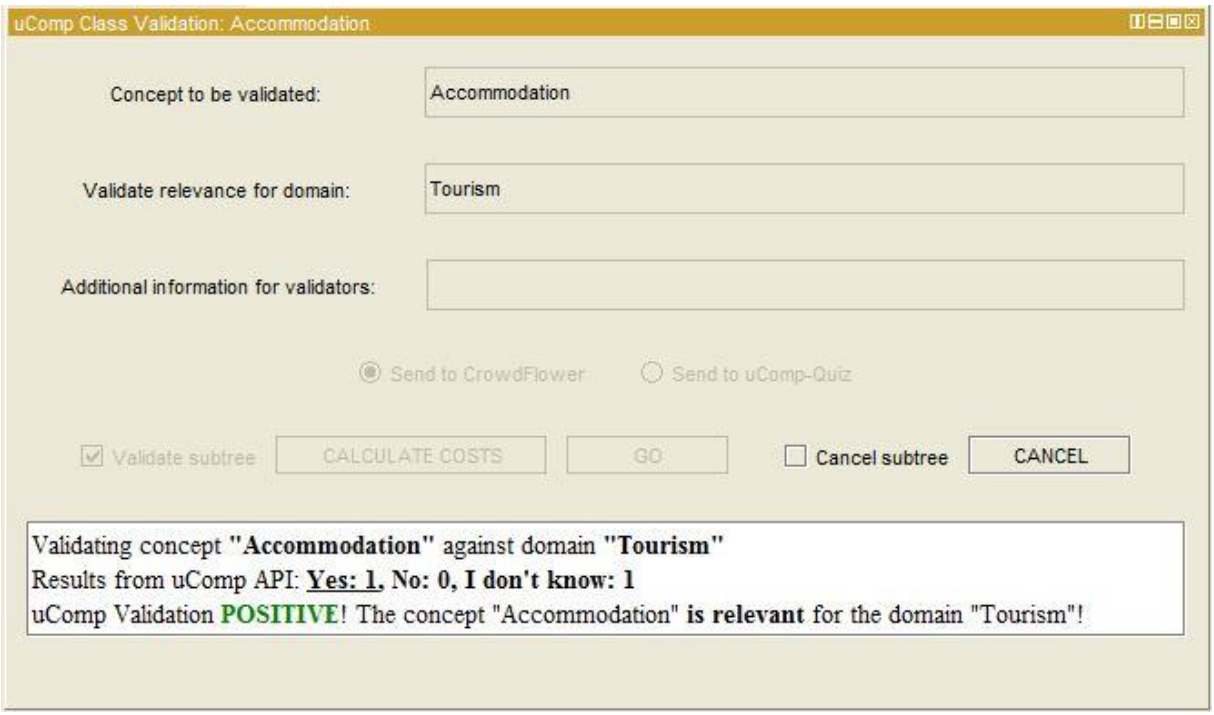
To get a uComp-API key, request it from the uComp team, see http://soc.ecoresearch.net/facebook/election2008/ucomp-quiz-beta/api/v1/documentation/.
You need to specify your CrowdFlower key which will then be used by the uComp API.

General Information

Upon start, Protégé first loads the core framework which then checks and loads all plugins. In addition to the user interface of Protégé an associated console windows opens at every start. It works as standard output and log for Protégé. This helps to see which plugins are loaded by Protégé and if there are any errors.
At every start Protégé checks for updates for all installed plugins. By default they are installed into the home directory of the current user. (*For Windows users:* By giving the security principal „Everyone“ full control in the NTFS-permissions of the plugins-Folder (e.g. C:\Program Files\Protege_4.3\plugins) it’s possible to allow Protégé to overwrite the original jar-File.)

Until now, five types of validation tasks have been implemented, which are described in the following.

Crowdsourcing Class Validation



Crowdsourcing Class Validation lets users validate one or more ontology classes against a domain at the same time, ie. check if a class (concept) is relevant for the domain given.

Usage:

1. Load the ontology and select the "Classes" tab. From the menu, select "Window" → "Views" → "Class Views" → "uComp Class validation" and add it to the user interface.
2. The user selects an ontology class by clicking on it in the class hierarchy (i.e. „Accommodation").
3. Define the domain against which the class is to be validated (i.e. „Tourism") and optionally give some additional information for the validators. This additional information helps the HC users (the validators) to better understand the task and will be displayed to them, in addition to the basic task description which is included in the Plugin.
4. The user decides if the task should be sent to CrowdFlower or the uComp-Quiz (GWAP).
5. With a click on the GO-button the validation is started. Status information is displayed in the text area below.
6. The plugin sends all the necessary data to the crowdsourcing API, which creates a new crowdsourcing task. After the task has been finished, the plugin receives the results from the uComp API and displays it to the user in the text area.
7. During the whole task the user can cancel the current validation (also with the option to cancel the whole subtree). In this case, the user interface is reseted and a cancellation request is sent to the API.

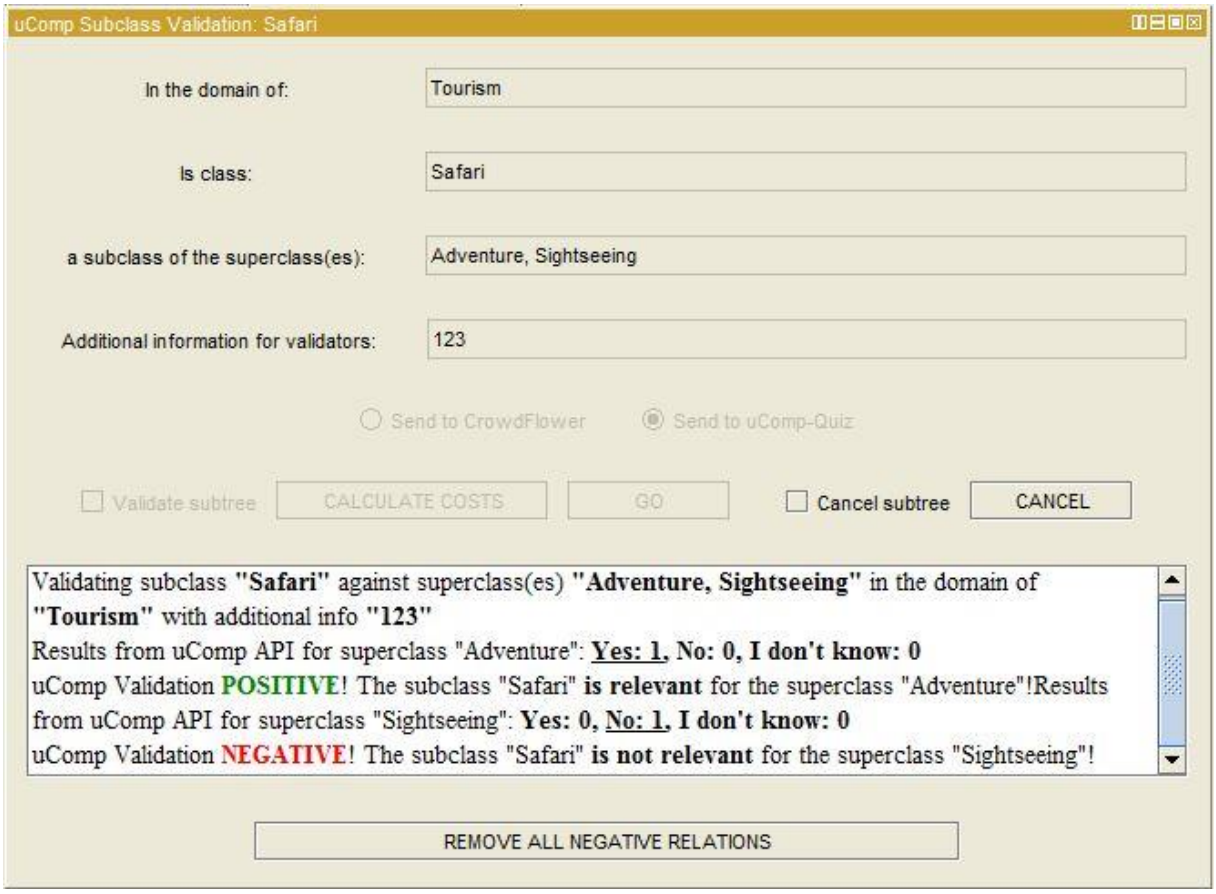
Remarks:

- During the first validation, the given domain is saved in the ontology for further validations. That means if the user decides to validate another class, the textfield for the domain is automatically populated by the plugin (of course it's editable to give the user the opportunity to change it).
- If the user wants to validate **more than a single class**, (s)he can select the checkbox for the validation of the class subtree. In this case, every subclass of the current class also gets validated (recursively!) If you want to validate the **whole ontology,** select the *uppermost class ("Thing")*, and select option "Validate subtree". Then, all concepts will be validated.
- If "Send to CrowdFlower" is selected, you can calculate the expected costs based on the number of units included in this task and the cents paid per judgement by clicking on the Calculate Costs-Button. No data is sent to the API until you click on the Go-Button.
- The information shown in the user interface only refers to the currently selected class. Therefore, when a user has chosen to validate the whole subtree, he has to select another class in the subtree to show the status and results for that class.

The following text will be sent to the API as task description when starting a Class Validation:

Check word relevance for a domain
Please decide whether the given word (also known as concept) is relevant for the mentioned domain.
Generally, *relevant* means that the word comes to ones mind when thinking about that domain.
Class Relevance Check:
Your task is to decide if a given concept (also called class) is relevant for a given domain.
Examples:
Is racket relevant to the domain of tennis? - Correct answer: Yes
Is game relevant to the domain of tennis? - Correct answer: Yes
Is election relevant to the domain of politics? - Correct answer: Yes
Is racquet relevant to the domain of politics? - Correct answer: No
Is party system relevant to the domain of politics? - Correct answer: Yes
Is partysystem relevant to the domain of politics? - Correct answer: No, this is no English term!
Sometimes there is no answer that is clearly correct, because the concept may be slightly relevant, too generic, or too specific.
Examples:
Is human relevant to the domain of politics? - Unclear, probably: Yes, but very generic.
Is weather relevant to the domain of politics? - Unclear, probably: No, only slightly relevant.
Is event relevant to the domain of politics? - Unclear, but probably: Yes.
Please consult the Web or any external source for additional information you might need for completing this task (for example, checking the definition of climate related terms on Wikipedia).

Crowdsourcing Subclass Validation



The Crowdsourcing Subclass Validation enables a user to validate **subclassof** relations between a class and its superclasses and works similar to the GWAP Class Validation. Therefore we will mention only the differences between the two.

Usage:

1. Load the ontology and select the "Classes" tab. From the menu, select *"Window"* → *"Views"* → *"Class Views"* → *"uComp SubClass validation"* and add it to the user interface.
2. When the user selects a class from the class hierarchy, the plugin automatically looks for all of its **superclasses** and puts the labels of the class and all of its superclasses into the first two textfields.
3. Again, users can provide additional information and have the option to select the whole subtree to be validated.
4. When a class has more then one superclasses (as in the example above), a validation task for each superclass will be created.
5. All results are displayed in the text area. The example also shows what happens if the result includes a negative validation: A button is displayed, which allows the user to delete all subclassof relations validated as „not relevant".
6. If you want to **validate all subclass relations** in the ontology, go select the uppermost class ("Thing") and select option "validate subtree".

The following text will be sent to the API as task description when starting a Subclass Validation:

Verify that a term is more specific than another

Task: You task is to decide if a term A is more specific than a term B.

A term A (e.g., cat) is more specific than a term B (e.g., animal) if it can be said that *every A is a B* but not the other way around. Indeed, every cat is an animal but not every animal is a cat.

If A (e.g., cat) is more specific than B (e.g., animal), we say that A is a subClass of B

Attention! The order of the terms is important:

cat - is a a subClass of – animal is TRUE because every cat is also an animal (cats are types of animals)

animal - is a a subClass of – cat is FALSE because not every animal is a cat (animals are not types of cats)

Examples: Is human a subClass of mammal? Correct answer: Yes

Is human a subClass of car? Correct answer: No

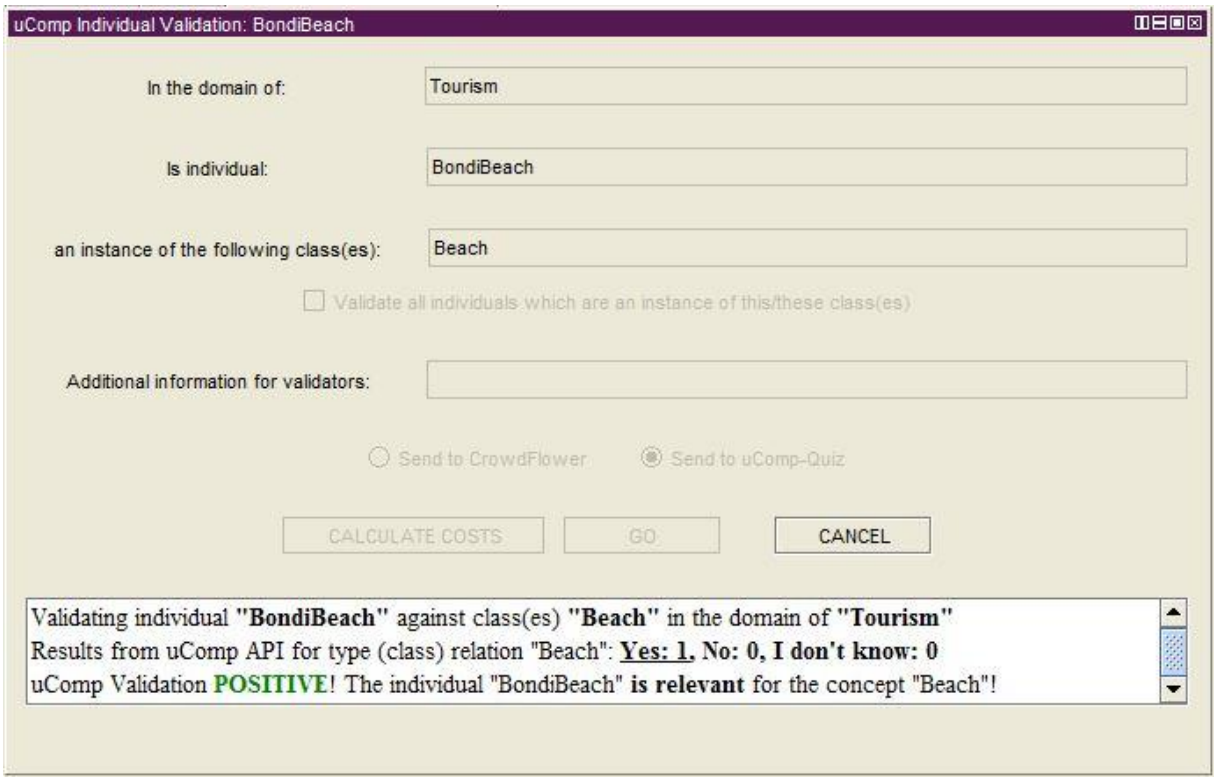
Is politician a subClass of human? Correct answer: Yes

Is water a subClass of liquid? Correct answer: Yes

Is water a subClass of car? Correct answer: No

Please consult the Web or any external source for additional information you might need for completing this task (for example, checking the definition of climate related terms on Wikipedia).

Crowdsourcing Individual Validation

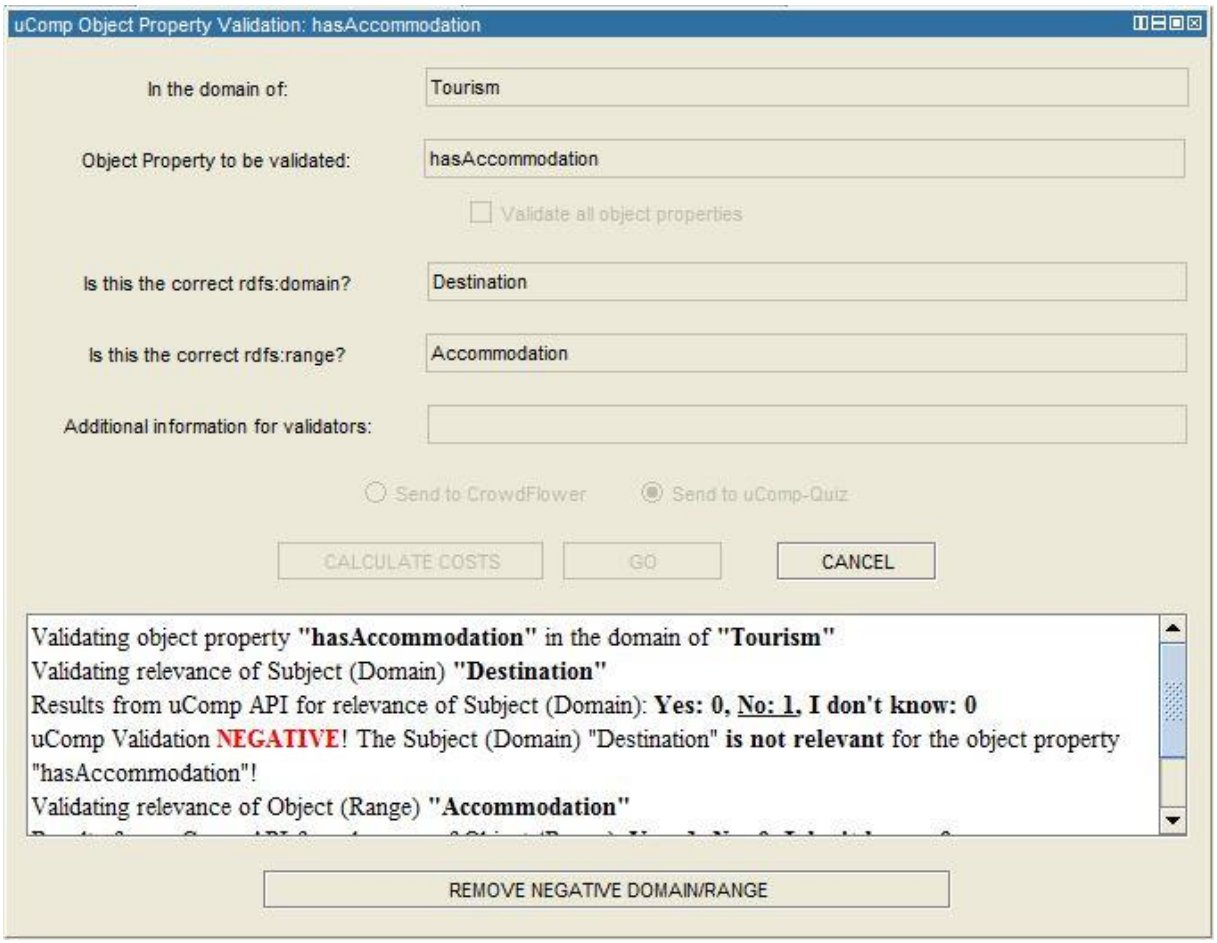


The Crowdsourcing Individual Validation lets users validate the relation of an OWL individual to its corresponding OWL class, ie. if the instanceOf relation is valid. This task can be seen as a mixture of the two previous validation tasks when looking at the validation scenarios.

Usage:

1. When a Protégé user selects an individual, the plugin automatically puts its name and the corresponding class in the first two text fields.
2. As in Subclass Validation it is possible that an individual belongs to more than one class. In this case a validation for each of the classes will be triggered. The difference to the class validation is that no subtree validation is possible, as an individual can have no sub-individuals.
3. It is possible to validate all individuals which are also an instance of one of these classes at the same time. Simply select the respective checkbox to activate this feature.
4. Again, it is possible to cancel the current validation in progress.
5. If a validation task has a negative result, the respective individual can be deleted.

Crowdsourcing Object Property Validation



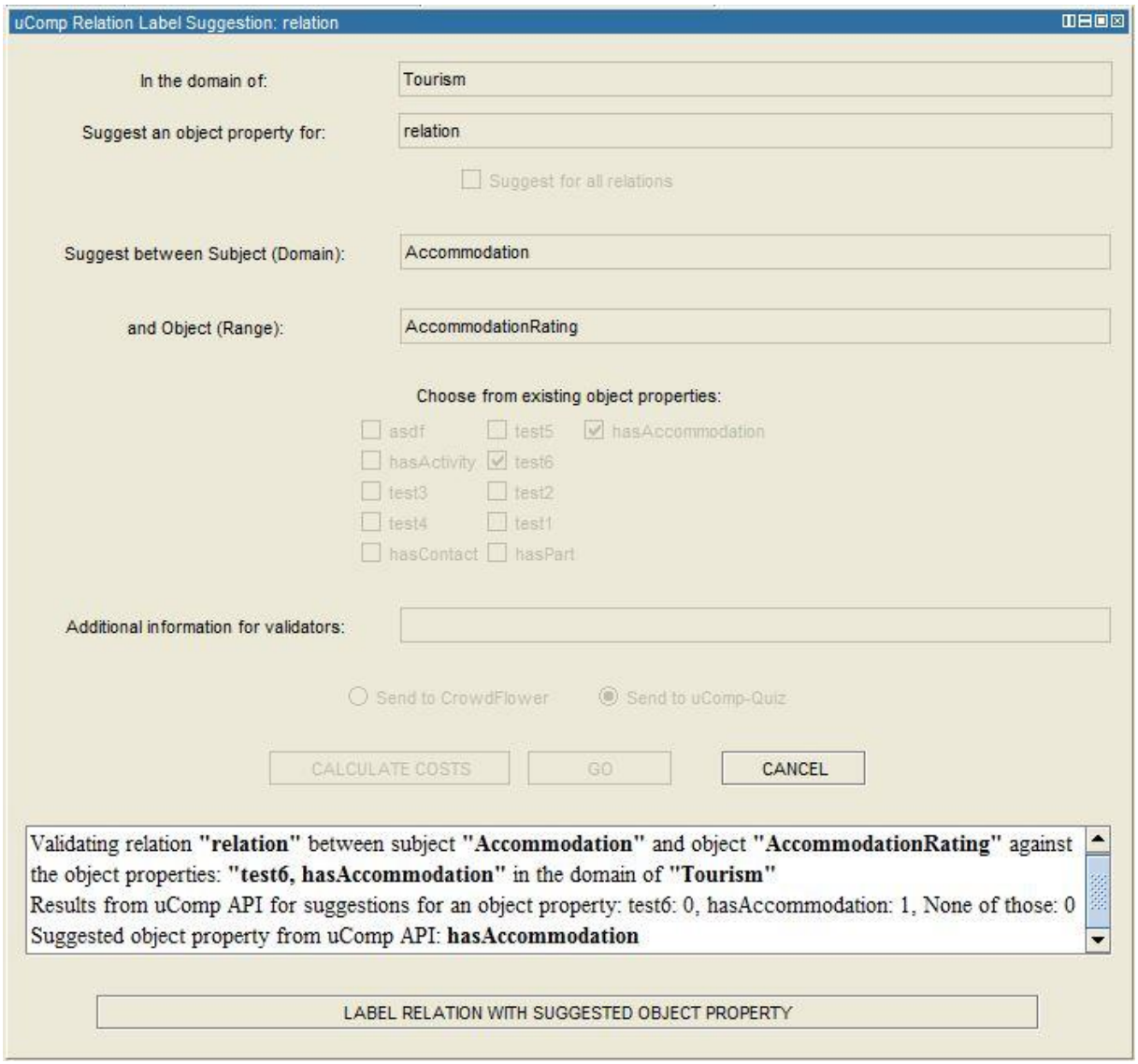
With the Crowdsourcing Object Property Validation users can validate the **domain/range** restrictions of a given object property. It differs from the previous task by having more elements and also in the way the question for the Crowdsourcing users is formulated. In contrast to the other validation tasks there are two questions per object property (to check the correctness of both the domain and the range restriction).

One must distinguish between the general domain (of knowledge) of the ontology and the **Domain restriction** of a class, which should be trivial for an ontology engineer.

Usage:

1. The Protégé user selects an object property, the plugin automatically fills out the text fields with the label of the object property and its *subject (domain) and object (range)*.
2. If the general domain is already defined for the ontology, the plugin puts it into the respective text field (“In the domain of”).
3. By selecting the respective checkbox it is possible to validate all available object properties at the same time.
4. When the validation is finished, the plugin displays the results of the two validations. Again, if there was a negative validation result, the Protégé user has the option to delete the respective relations.

Crowdsourcing Relation Validation



This task is relevant especially for ontologies which have been created by automated methods, such as ontologies from ontology learning systems. Often ontology learning systems can only detect an unlabeled relation between two classes. The goal of this Crowdsourcing task is to select an appropriate relation label between the two classes from a given list of labels (object properties).

Usage:

1. The plugin allows Protégé users to select one or all of the mentioned unlabeled relations. The user also selects which of the existing object properties should be sent to the HC API as canidate relation labels for the unlabeled relations.
2. The plugin then sends this information to the HC API. The HC users now have to choose which of the available object properties is most appropriate for the relation between the two objects.
3. By selecting the respective checkbox it is possible to validate all unlabeled relations together.
4. The result is sent back to the plugin and displayed to the user, who can then accept the decision in order to replace the unlabeled relation with the chosen object property.