# Language Detection Web Application – Duck Soft Works § Co.

# US 2 – As a user I want to create a text language analysis task, passing a URL (which points to a .txt file), a category and a timeout.

## 1. Requirements Engineering

### 1.1. Customer Specifications and Clarifications

### Q: What does the client mean when he says "to obtain information about the language in which the text was written"? R: It is the purpose of the task launched by the user. The user wants to identify the language of a text.

**Q:** Is there a minimum or maximum size of texts to be analyzed?

**R:** Does not exist a limit.

**Q:** How many languages should be covered, and if so, which ones?

**R:** At least 3 languages must be covered: Portuguese, Spanish and English

**Q:** Is there any maximum limit for the Task Timeout?

**R:** This question has already been asked. Since you are being so insistent, consider the maximum limit of 5 minutes.

### And from the specifications:

### The text to be validated must be stored in a text file and must be identified by the URL that allows it to be located, and by one of the existing categories in the information system.

### 1.2. Acceptance Criteria

- The analysis of the text must be performed asynchronously so that the user is not left waiting for the analysis to be completed

- For language identification services, client suggests the use of dictionaries from the Aspell project and the Lucene project and the use of the concepts of similarity between documents.

### 1.3. Found out Dependencies

To create a task, it is necessary to insert a category previously created by the system administrator and that the URL be with the extension .txt

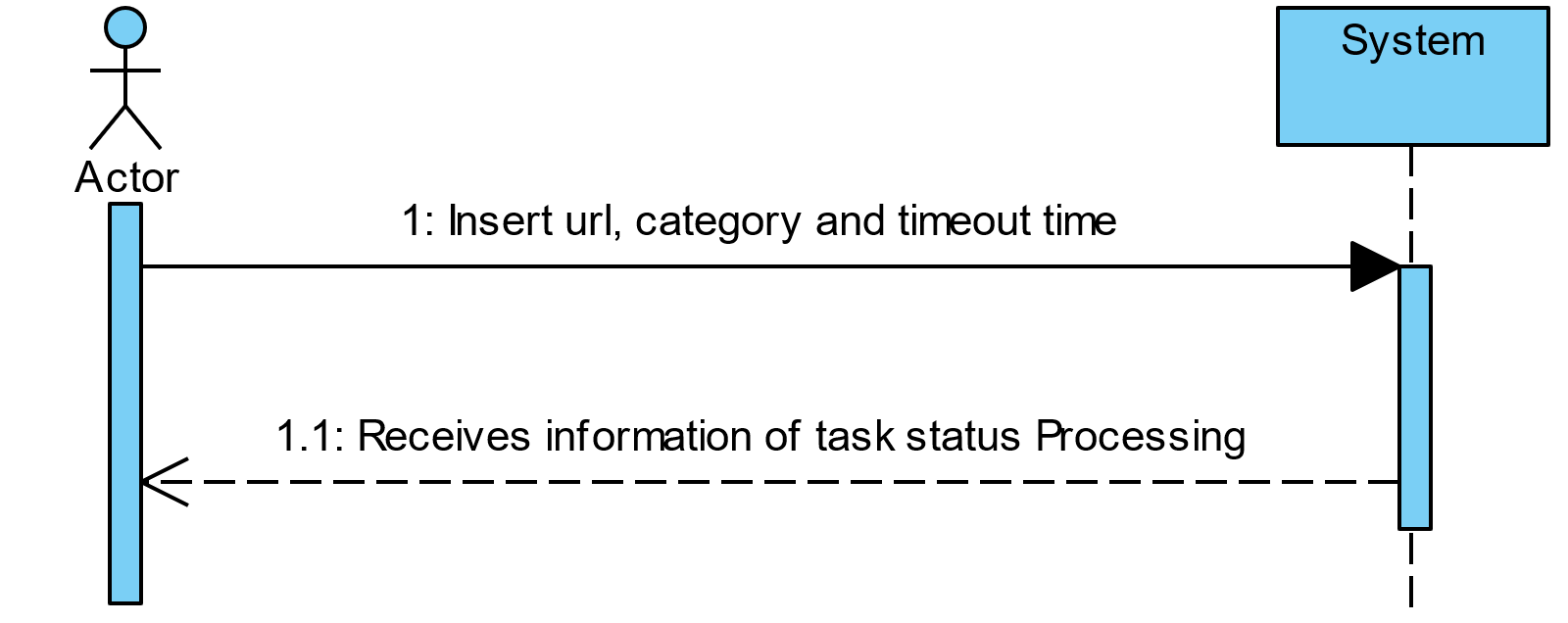
To be able to create a task, a user must provide a valid text url that is not contained in the BlackList. For that reason, the url must always be compared to all the elements in that list to make sure it has permission to continue to the language detection analysis.

It is also always necessary for the user to send a time limit (timeout) for the analysis of the language. This time should be between 1 and 5 minutes.

### 1.4. Input and Output Data

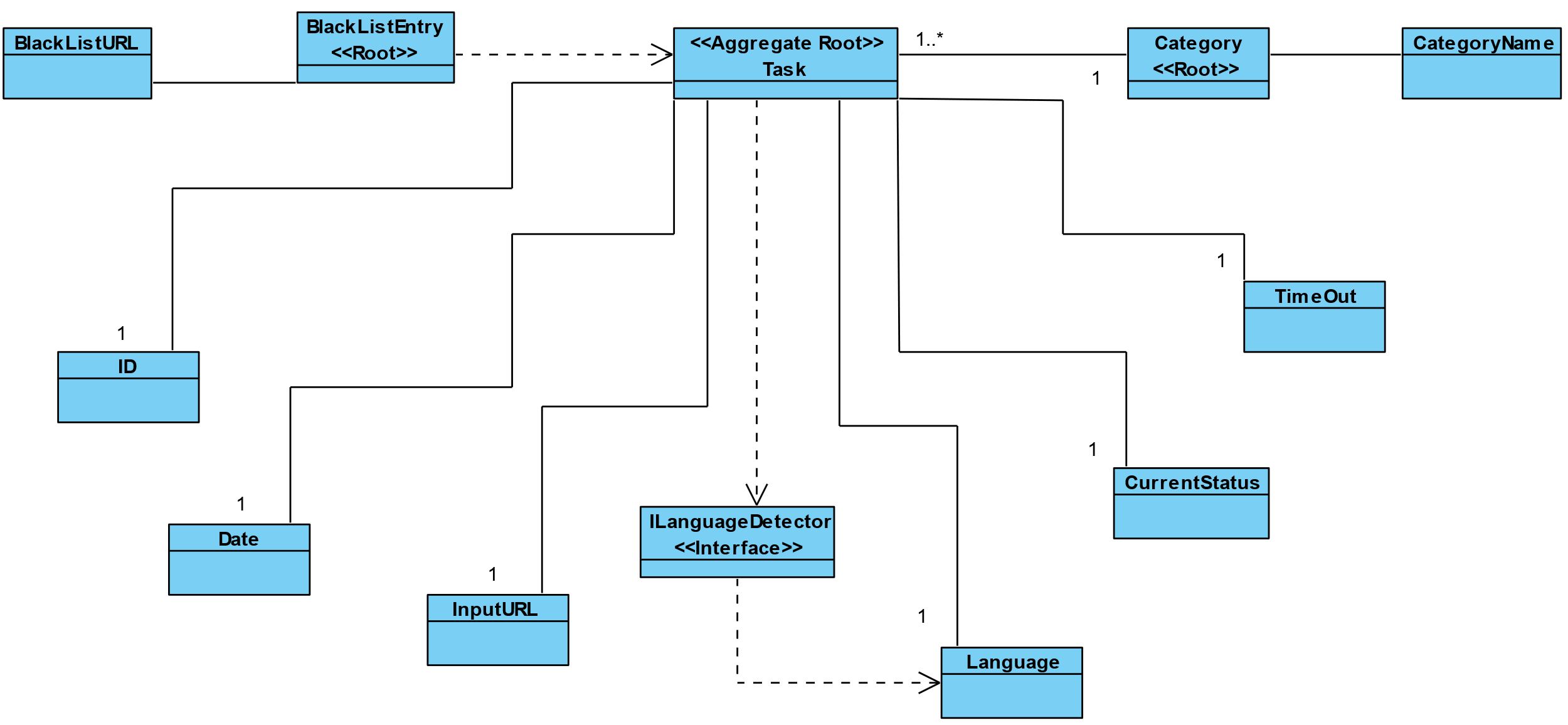
### To be able to create an analysis task, it is necessary to introduce a URL link with a .txt extension that points to a text file, a time limit, and a previously registered category.

### 1.5. System Sequence Diagram (SSD)

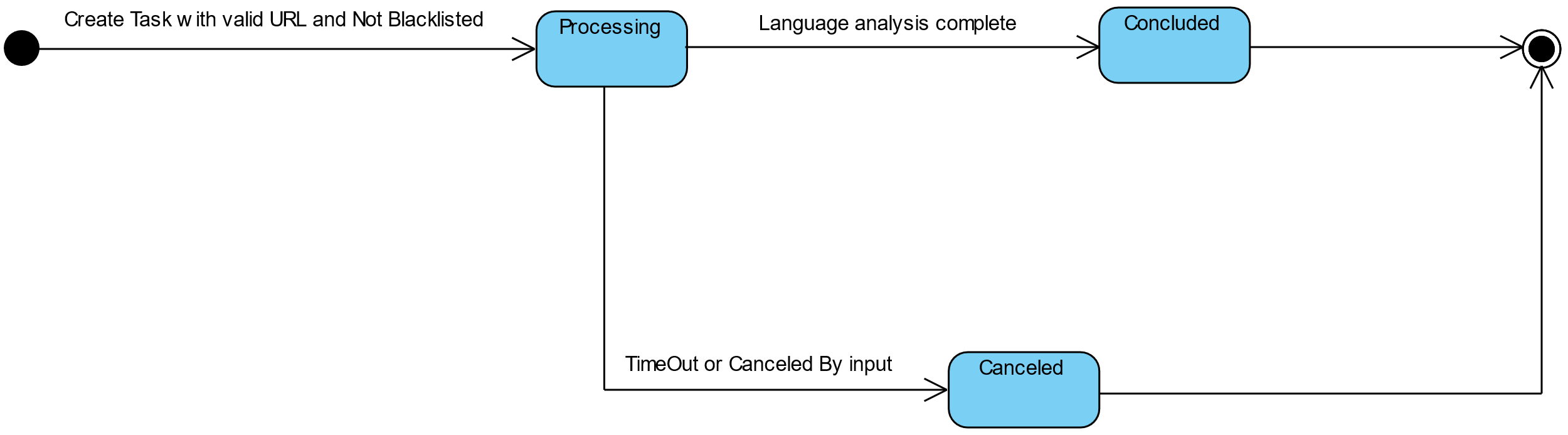


## 2. OO Analysis

### 2.1. Relevant Domain Model Excerpt



### 2.2. Other Remarks



This diagram represents the change of task status.

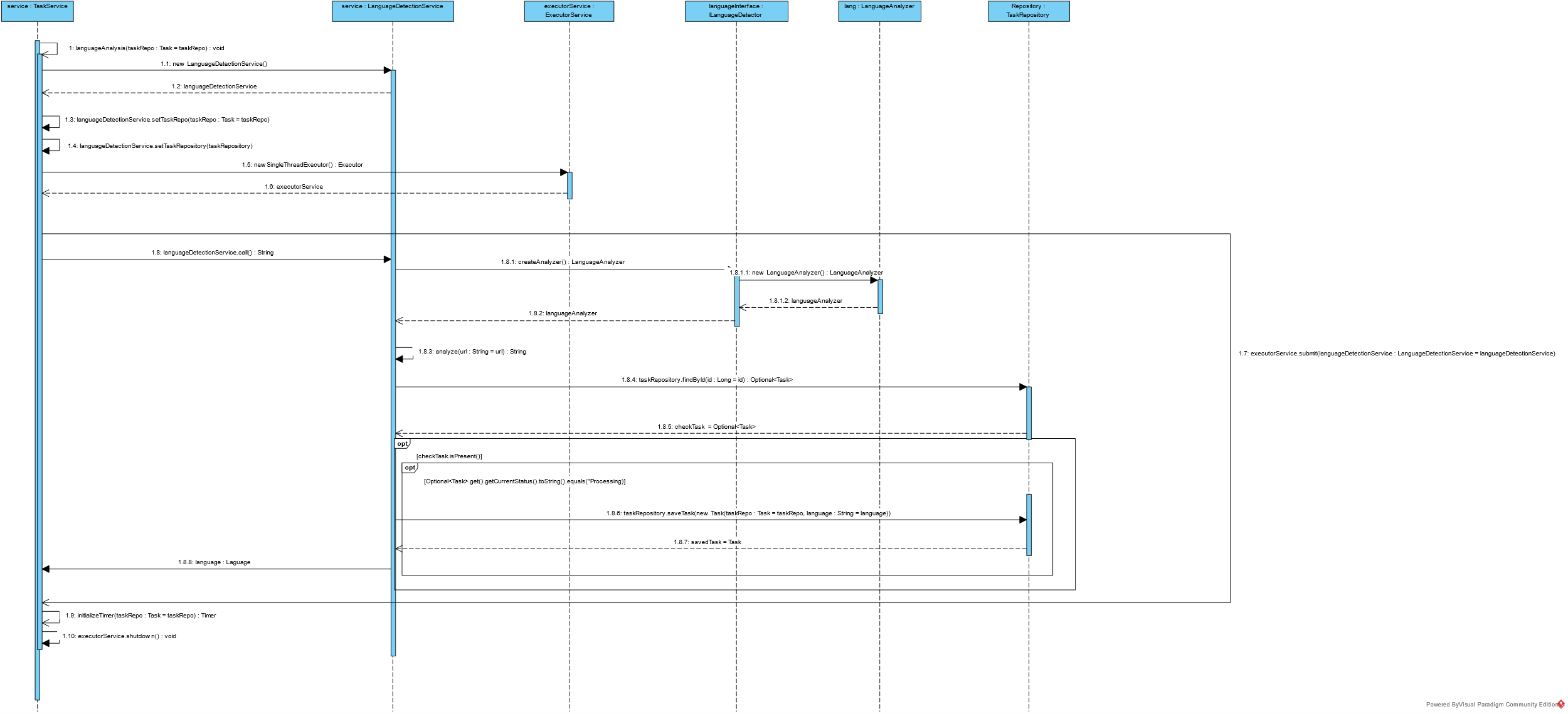
## 3. Design - User Story Realization

## 3.1. Sequence Diagram (SD)

## 

This diagram represents task creation without asynchronous process of language identification.

Below it’s possible to check the asynchronous process of language identification.



## 3.3. Class Diagram (CD)

In this US, as it depends of almost every class on our project, it’s possible to check our general class diagram in our CD folder.

# 4. Tests

@org.junit.jupiter.api.Test  
void createAndSaveTaskShouldReturnOptionalEmptyIfCategoryExistsAndUrlIsBlackListed() throws IOException {  
 NewTaskInfoDTO infoDTO = new NewTaskInfoDTO("http://www.textexample.com/text/text.txt", "Sports", 2);  
 NewBlackListInfoDTO blackListInfoDTO = new NewBlackListInfoDTO("http://www.textexample.com/text/text.txt");  
 Category category = new Category("Sports");  
  
 *when*(blService.isBlackListed(blackListInfoDTO)).thenReturn(true);  
 *when*(categoryService.findById(category)).thenReturn(Optional.*of*(category));  
  
 Assertions.*assertEquals*(taskService.createAndSaveTask(infoDTO), Optional.*empty*());  
}

# 5. Construction (Implementation)

We identified a clear difficulty when canceling the task and trying to cancel the correspondent thread. As for, we would like to improve it in the future. Maybe with the implementation of another process of detecting the language it would be possible to use a different interrupting method as for, in this project, by the way it is constructed and how it detects the language, it prevents the termination of the thread in a graceful and safe way.

# 6. Integration and Demo

The main difficulty identified was performing the tasks asynchronously. The studies involved in creating the tasks in this way took a lot of time and still have flaws that can be improved in the future.

# 7. Observations

In this section, it is suggested to present a critical perspective on the developed work, pointing, for example, to other alternatives and or future related work.