



ADVANCED MULTIMEDIA APPLICATIONS

Feliciaan De Palmenaer  
Wouter Pinnoo  
Stefaan Vermassen  
Titouan Vervack

ACADEMIC DATA - AUTOMATIC COURSE ASSEMBLY

2015-2016

# Contents

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>General Information</b>                         | <b>3</b>  |
| <b>2</b> | <b>General project decisions</b>                   | <b>3</b>  |
| 2.1      | Decisions . . . . .                                | 3         |
| 2.2      | Responsibilities . . . . .                         | 5         |
| 2.3      | Requirements . . . . .                             | 5         |
| 2.3.1    | Functional requirements . . . . .                  | 5         |
| 2.3.2    | Non-functional requirements . . . . .              | 6         |
| 2.4      | Assumptions . . . . .                              | 7         |
| 2.4.1    | Input . . . . .                                    | 7         |
| 2.5      | Risk list . . . . .                                | 8         |
| 2.6      | APIs & frameworks . . . . .                        | 8         |
| <b>3</b> | <b>Prototype</b>                                   | <b>9</b>  |
| <b>4</b> | <b>Application</b>                                 | <b>18</b> |
| <b>5</b> | <b>Overview of the different meetings</b>          | <b>18</b> |
| 5.1      | Initial meeting with client (24/02/2016) . . . . . | 18        |
| 5.2      | Review with teaching staff (25/02/2016) . . . . .  | 19        |
| 5.3      | Review with teaching staff (10/03/2016) . . . . .  | 19        |
| 5.4      | Review with client (16/03/2016) . . . . .          | 20        |
| 5.5      | Review with teaching staff (17/03/2016) . . . . .  | 20        |
| 5.6      | Extra internal meeting (28/03/2016) . . . . .      | 20        |

|     |   |    |
|-----|---|----|
| 5.7 | Review with client (30/03/2016) . . . . . | 20 |
| 5.8 | Review with client (13/04/2016) . . . . . | 21 |

# 1 General Information

Onsophic, a Silicon Valley based startup focused on transforming learning, offers an intuitive, data driven, online training platform. The platform continuously gathers and analyses learning data and can therefore be seen as the equivalent of Google Analytics for learning by providing organizations with a toolset to measure, analyse and discover what works and, more importantly, what doesn't work in training. In this business process, integrating existing content into the e-learning platform is crucial. The goal in this project is to automate the course assembly process, by transforming raw learning materials into structured courses.

## 2 General project decisions

This section will outline all of the important decisions we have made during the development of the project.

### 2.1 Decisions

#### **Iteration 1 (Course introduction - 18/02/2016)**

- Stefaan will fulfil the role of team lead. (18/02)
- Stefaan will be responsible for the communication with the other parties. (18/02)
- Git will be used as distributed versioning control system for the code, with the UGent GitHub platform as tool. (20/02)
- Titouan will be responsible for the progress report and will make sure it is always up to date. (22/02)
- Wouter will be responsible for git and over the code quality. (22/02)
- Feliciaan will be responsible for the planning of the project. (22/02)
- For scalability purposes, a web service is chosen over a native application (22/02)

#### **Iteration 2 (Starting 24/02/2016)**

- Play web application framework is chosen. Play is written in Scala and Java and is a clean alternative to the legacy Enterprise Java stacks focusing on predictable, minimal resource consumption (25/2)
- Akka is chosen to create the asynchronous task system. Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM (25/2)
- For Named-Entity Recognition, the NERD API was chosen for its abstraction of other NER APIs. (25/2)
- We chose to use jsoup for our HTML parser as it doesn't require us to create the syntax tree ourselves and because it implements the WHATWG HTML5 specification. This on it's own, using ANTR4, would take quite some time. The license used for jsoup is the MIT license, strengthening our choice. (25/2)

### Iteration 3 (Starting 09/03/2016)

- To decrease the impact of the risk "*NER processing is not quick enough for practical usage of our tool*", use several NER services in parallel and only choose the fastest one (12/03)
- Implement threading for parallel processing of the several documents that have to be recognised in one task (12/03)

### Iteration 4 (Starting 16/03/2016)

- We have defined a set of supported input formats. We noticed that most user manuals come with a Toc. The web service will expect this Toc only, and will dynamically decide which formatting is used (nested unordered lists, nested tables, divs,...).
- Error handling implementation
- Get partial JSON before NER to avoid long waiting times before any result
- Chapters are meant to group modules, no other metadata is attached to a chapter. Chapters are optional. So if you are able to detect an extra level on top of modules, you can use chapters to reflect that extra level.
- The only tags that matter for now are the section tags, so we only focus on these.

## 2.2 Responsibilities

| Responsible        | Task  |
|--------------------|---|
| <b>Iteration 2</b> |   |
| Titouan            | <ul style="list-style-type: none"><li>- Responsible for the initial architecture</li><li>- Digitalising the component-connector diagram</li><li>- Initial wireframe of classes</li></ul>  |
| Feliciaan          | <ul style="list-style-type: none"><li>- Library research for input parsing</li><li>- Adding HTML parser to the solution</li><li>- Handle multiple URLs</li></ul>  |
| Wouter             | <ul style="list-style-type: none"><li>- Research on Named-Entity Recognition libraries</li><li>- Making a motivated decision on which NER tool or tools are most suitable for what you need, and which will be the easiest to integrate into the solution, and be most future-proof</li></ul> |
| Stefaan            | <ul style="list-style-type: none"><li>- Setting up the initial Play project</li><li>- Creating an asynchronous task system using Akka to come up with a scalable approach</li></ul>   |
| <b>Iteration 3</b> |   |
| Titouan            | <ul style="list-style-type: none"><li>- Implement threading (parallel functionality for different documents in the same task)</li></ul>   |
| Feliciaan          | <ul style="list-style-type: none"><li>- Use the parsed JSON document to fill in section names in the Onsophic JSON document</li></ul>   |
| Wouter             | <ul style="list-style-type: none"><li>- Risk list, planning and presentation</li></ul>  |
| Stefaan            | <ul style="list-style-type: none"><li>- Research on Onsophic JSON format, progress report</li></ul>   |
| <b>Iteration 4</b> |   |

## 2.3 Requirements

### 2.3.1 Functional requirements

#### Must-haves

- parse the input source (HTML)
- detect the learning modules and their learning activities
- tag these detected modules and activities based on named entity recognition
- estimate the Bloom level for each of the detected activities

### Nice-to-haves

- Detect chapters in the HTML document, as these are optional (property of a module).
- Parse other types of documents than use manuals.

### 2.3.2 Non-functional requirements

#### Must-haves

- Interoperability. Our tool must be able to interchange information with third-party services correctly. See Quality-Attribute Scenario in Figure 1.

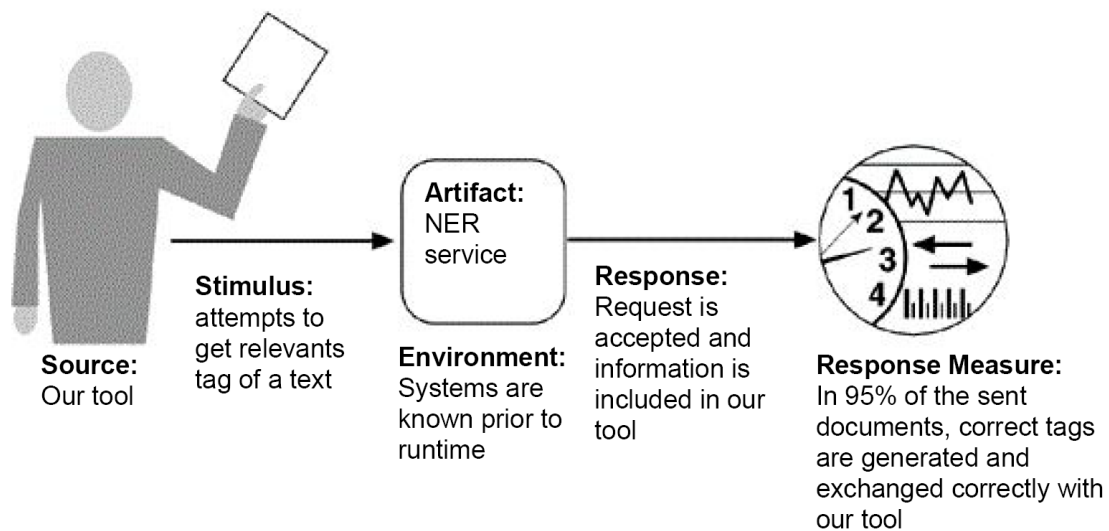


Figure 1: QAS: interoperability

- Modifiability. A developer must be able to change formats of input/output documents without having an impact on other modules. See Quality-Attribute Scenario in Figure 2.

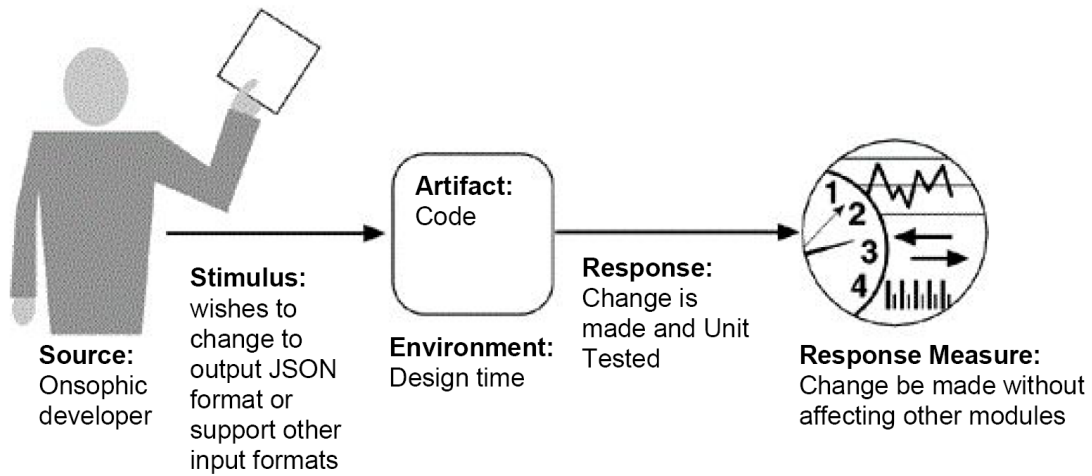


Figure 2: QAS: modifiability

## 2.4 Assumptions

### 2.4.1 Input

After surveying multiple different helpsites we didn't find any similarities that worked for all sites. Some sites can be parsed by counting the depth of the node in the parse tree. Some sites have a first page which is the index, other sites have the index on every page, some even need use javascript to create the whole index of the site.

At the moment we have implemented our parser for sites that have a table of contents. We assume that every html document contains exactly one document.



## 2.5 Risk list

| Risk  | Probability | Impact | Mitigation   |
|---|-------------|--------|--|
| Not finding a generic line in the different input sources                             | M           | H      | Review the assumptions about the input data with the client    |
| No suitable NER tool that is free to use and license to use in closed-source software | L           | H      | Looking into commercial tools                                  |
| Time-management issues due to other projects and master dissertation                  | M           | M      | Discuss planning and requirements with client.                 |
| NER processing is not quick enough for practical usage of our tool                    | M           | M      | Use several services in parallel and only use the fastest one. |
| Not able to fill in all metadata (not fully compatible JSON)                          | H           | L      | Use simplified JSON.   |

## 2.6 APIs & frameworks

1. Play Framework: web application framework, written in Scala and Java, clean alternative to the legacy Enterprise Java stacks, focussing on predictable, minimal resource consumption
2. Akka: toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM
3. For the parsing of HTML we use jsoup. It implements the WHATWG HTML5 specification and creates the same DOM (Document Object Model) as modern browsers create.
4. Named-Entity recognition will be done using the NERD API.

### 3 Prototype

## Iteration 2 (Starting 24/02/2016)

## Architecture

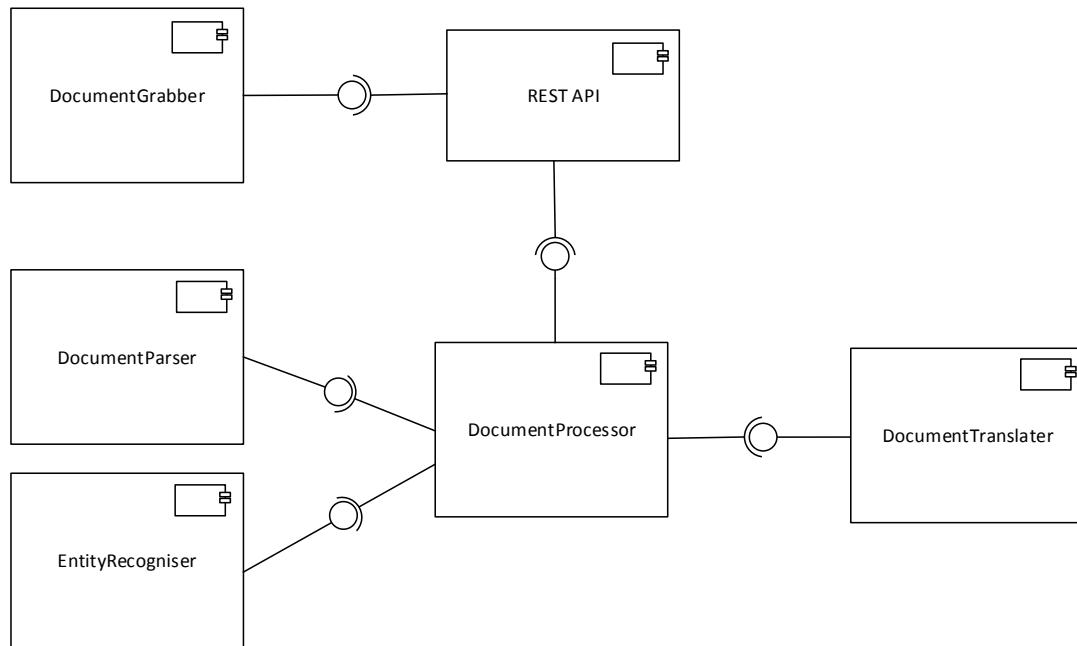


Figure 3: Initial high-level architecture of the Automatic Course Assembly service

Our initial architecture is modular and has as goal to easily allow to extend our application with more features later on. Adding a new type of input or output format for example should be as easy as possible.

First and foremost we have the REST API, this is the starting points of our application. A url will be passed to our REST API after which it will use the right DocumentGrabber to grab the document from the url. An example of a DocumentGrabber would be an HTMLGrabber that knows how to grab HTML documents and returns an unprocessed HTMLDocument.

Throughout the application we will work with Documents to pass around the (un)processed data.

The REST API then passes the Document it grabbed to the DocumentProcessor. The

Processor calls the DocumentParser which will create a parse tree, analyse it and create a new Document, during analysis the learning modules and activities are created. In this Document only the useful parts of the input document are left over.

The DocumentProcessor then passes the parsed Document to the EntityRecogniser which will perform Named Entity Recognition (NER) on the Document. This module will add tags to the Document and pass it back to the DocumentProcessor.

Last but not least the DocumentProcessor passes the Document to a DocumentTranslator which will translate a completely processed Document into a Document fit for outputting. In our current case, this will translate an HTMLDocument into a JSONDocument.

The only document that can be accessed from the outside is the REST API, everything else is hidden from the user. The user can query the API to start a new import, get the status of a previous task or get the generated output document.

## NER

In the assembly process of the JSON-representation of the course, tags will be created in addition to the parsed content of the (HTML-) documents. For each document, these tags will represent the subject of the document. For this purpose, the content of the whole document will have to be scanned, since the title of the document is not always sufficient to serve as tag for the document. For example, user manuals often contain *Prerequisites* as a title for the first section of the manual. This section will probably contain more useful information than only this subject line.

For the automatic scanning of the document into useful tags, Named-Entity Recognition (NER) will be used. NER uses machine learning techniques to find (sequences of) words in the text, i.e. entities, that can be related to concepts from the semantic web. For example, in the sentence *Trump and Clinton are two candidates for the presidential elections of 2016*, the entity *Trump* will be linked to [http://dbpedia.org/page/Donald\\_Trump](http://dbpedia.org/page/Donald_Trump). The used algorithm will make decisions based on relevance of the word to the concept of the semantic web and its confidence.

We think NER is the best solution for the generation of tags, since

- NER finds sequences of words that are relevant (i.e. leave out words like *'and'*, *'or'*, *'the'*, etc.
- NER provides a measure of relevance, which we can use to sort all entities on. Only

the 5 most relevant concepts in the document will be used as tag for the document.

Several NER libraries exist that serve an API for easy access. We chose to use the NERD API<sup>1</sup>. This API acts as an abstraction for other existing NER APIs, one of which is the most commonly used NER API: AlchemyAPI. The NERD API provides easy methods to submit a document, choose a NER library and retrieve the results of the chosen extractor.

This abstraction layer will be a big advantage in this project, because we can easily switch from extractors to tune performance and correctness of the tags.

A disadvantage of the NERD API is its license, which prohibits any commercial use of its services. However, we think that the benefit of the abstraction layer outweighs the disadvantage of the license, since one can use the learned results of all the different used extractors gained in this project course to eventually implement the best performing extractor in the project (which will take more effort than implementing this abstraction layer).

## Demo

- Entering new URL in the system:

```
$ http http://localhost:9000/start\
?url=https://docs.oracle.com/cd/E18727_01/doc.121/e13522/toc.htm
HTTP/1.1 200 OK
Content-Length: 27
Content-Type: application/json; charset=utf-8
{
  "id": 1,
  "status": "QUEUED"
}
```

- Checking progress of a task

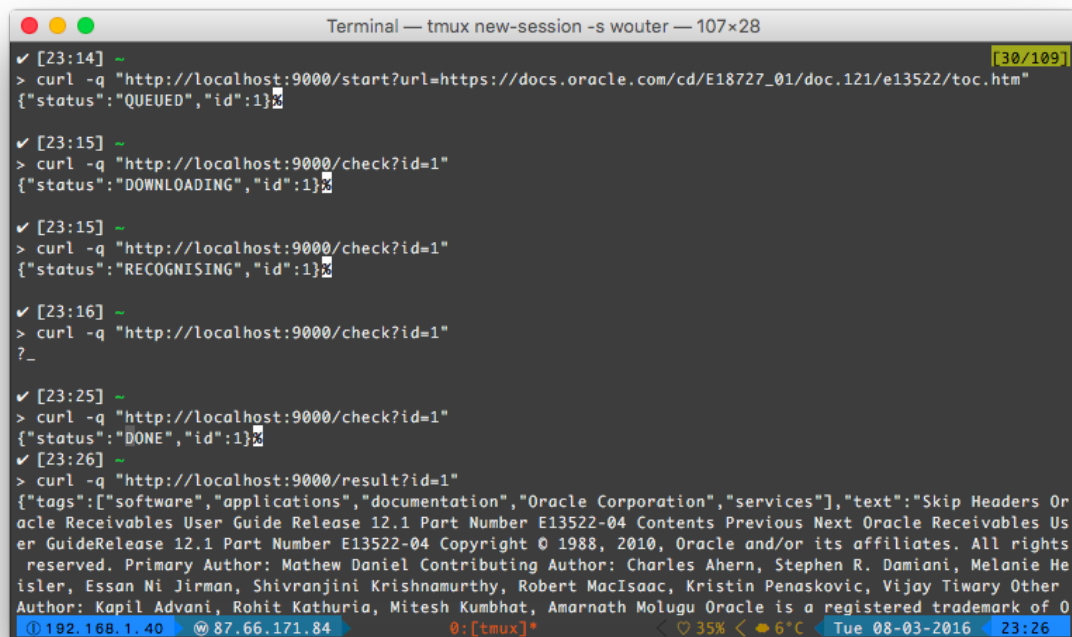
```
$ http localhost:9000/check?id=1
HTTP/1.1 200 OK
{
  "id": 1,
  "status": "RECOGNISING"
}
```

---

<sup>1</sup>see <http://nerd.eurecom.fr/>

- Retrieving the results of a task

```
$ http localhost:9000/result?id=1
HTTP/1.1 200 OK
{
  "tags": [ "tag1", "tag2" ],
  "text": "..."
}
```



```
Terminal — tmux new-session -s wouter — 107x28
[30/109]
✓ [23:14] ~
> curl -q "http://localhost:9000/start?url=https://docs.oracle.com/cd/E18727_01/doc.121/e13522/toc.htm"
{"status":"QUEUED","id":1}
✓ [23:15] ~
> curl -q "http://localhost:9000/check?id=1"
{"status":"DOWNLOADING","id":1}
✓ [23:15] ~
> curl -q "http://localhost:9000/check?id=1"
{"status":"RECOGNISING","id":1}
✓ [23:16] ~
> curl -q "http://localhost:9000/check?id=1"
?_
✓ [23:25] ~
> curl -q "http://localhost:9000/check?id=1"
{"status":"DONE","id":1}
✓ [23:26] ~
> curl -q "http://localhost:9000/result?id=1"
{"tags":["software","applications","documentation","Oracle Corporation","services"],"text":"Skip Headers Or
acle Receivables User Guide Release 12.1 Part Number E13522-04 Contents Previous Next Oracle Receivables Us
er GuideRelease 12.1 Part Number E13522-04 Copyright © 1988, 2010, Oracle and/or its affiliates. All rights
reserved. Primary Author: Mathew Daniel Contributing Author: Charles Ahern, Stephen R. Damiani, Melanie He
isler, Essan Ni Jirman, Shivranjini Krishnamurthy, Robert MacIsaac, Kristin Penaskovic, Vijay Tiwary Other
Author: Kapil Advani, Rohit Kathuria, Mitesh Kumbhat, Amarnath Molugu Oracle is a registered trademark of O
192.168.1.40 87.66.171.84 0:[tmux]* < 35% < 6°C < Tue 08-03-2016 23:26
```

Figure 4: Screenshot of the queueing process

## Iteration 3 (Starting 09/03/2016)

### Architecture

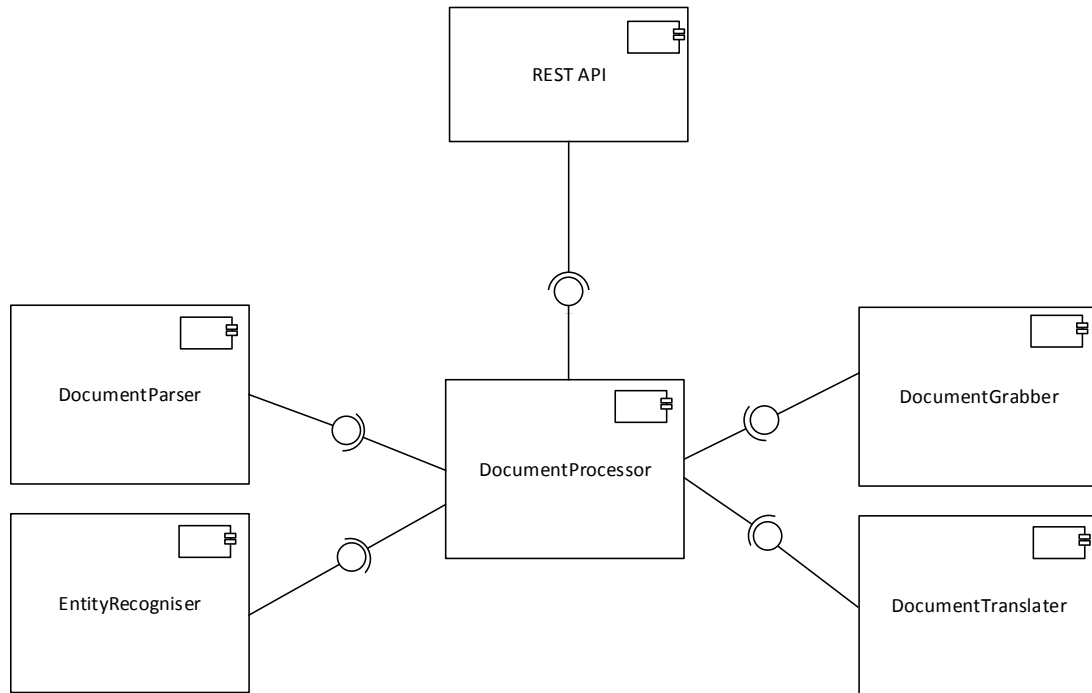


Figure 5: Revised high-level architecture of the Automatic Course Assembly service

A few changes were made in this iteration because we noticed that the document we receive as input has links to other documents that should also be grabbed and parsed.

In our second architecture the DocumentGrabber has moved to the DocumentProcessor. The API just passes the url of the first url to the DocumentProcessor. In the DocumentGrabber two new thread are created, one in which the DocumentGrabber grabs new documents and one in which the DocumentParser parses the grabbed Document. In the latter more urls can be derived from the document and these are added to the list of documents to grab.

The EntityRecogniser and DocumentTranslator remain the same but they are called once for every document that went through the parser.

## Retrieve section headers

As a first attempt in parsing, section headers from a HTML manual document were retrieved. Those headers are used in the `embeddedSections-title` field in JSON format of Onsophic, and the tags (retrieval implementation made in iteration 2) are used in the corresponding JSON field. The code below shows the output of our tool at the end of this iteration.

```
$ http localhost:9000/result?id=1
HTTP/1.1 200 OK
{
  "@type": "CourseEdit",
  "title": "AMMA test course",
  "embeddedSections": [
    {
      "@type": "SectionEdit",
      "title": "Oracle Receivables User Guide",
      "activity": {
        "url": "https://docs.oracle.com/cd/E18727_01/doc.121/e13522/toc.htm",
        "modality": "text",
        "activityType": {
          "id": "READ",
          "title": "Read"
        },
      },
      "title": "Oracle Receivables User Guide"
    },
    {
      "tags": [
        "Receipts",
        "Transactions",
        "Bills Receivable",
        "Customer",
        "Receivables"
      ],
      "visible": true,
      "optionality": "required"
    }
  ],
  "@type": "SectionEdit",
```

```

        "title": "...",
        ...
    }
]
}

```

## Iteration 4 (Starting 16/03/2016)

We have defined a set of supported input formats. We noticed that most user manuals come with a table of contents. The web service will expect this table of contents only, and will dynamically decide which formatting is used (nested unordered lists, nested tables, divs,..).

As Onsoptic indicated that they would like to test the service during the development to give more feedback and evaluate the integration in the learning platform, we decided to expose stable versions of this web service on a web server. For this purpose, an Azure web service has been created that always reflects the stable version on the master branch and is accessible on the following endpoint: **<http://13.94.197.112:9000>**.

The supported input formats will be explained in detail in the sections below. For each format, an example is provided.

### HTML unordered lists

The table of content lists all the sections in the form of a nested unordered list.

```

<ul>
  <li>List item one</li>
  <li>List item two with subitems:
    <ul>
      <li>Subitem 1</li>
      <li>Subitem 2</li>
    </ul>
  </li>
  <li>Final list item</li>
</ul>

```

Example: <http://13.94.197.112:9000/assets/examples/lists.html>.



```
Optional<Element> toc = document.getDocument()
.select("ul").stream().filter(l -> !l.parent().tag().equals("li"))
.sorted((e1, e2) ->
    e1.childNodes().size() > e2.childNodes().size() ? -1 : 1).findFirst();
```

To find the table of contents on the page, we first get all the lists that are not nested inside another list. The list with most childnodes will be selected. Each child in the parent list will be treated as a Module. If it seems that a Module node has no children, an Section and Activity with the same name will be created artificially. The URL of the activity will represent the URL of the Module node.

```
//No sublevels found, create new activity with this element's content
//If there is a link assigned to this listelement
if (!e.select(":root > a").isEmpty() &&
    !e.select(":root > a").get(0).attr("href").isEmpty()) {
    Section section = new Section();
    Element linkElement = e.select(":root > a").get(0);
    section.setTitle(linkElement.ownText());
    setActivity(linkElement, section);
    module.addSection(section);
}
```

However, as lists can be nested, the children of the Module nodes will be treated as sections. If the Section node has no children, an activity node will be created artificially. The children of Section nodes will be treated as activities.

## Description lists

The table of content lists all the sections in the form of a description list.

```
<dl>
  <dt>Firefox</dt>
  <dt>Mozilla Firefox</dt>
  <dt>Fx</dt>
  <dd>A free, open source, cross-platform, graphical web browser
    developed by the Mozilla Corporation and hundreds of volunteers.</dd>
</dl>
```

Example: `http://13.94.197.112:9000/assets/examples/definitionlists.html`.

## Raw weblinks

The table of content lists all the sections in the form hyperlinks. The parser will automatically follow each link and parse the pages on the deeper levels.

```
<a href="http://help.apple.com/ipad/5/voiceover/en/iPad73fccd85.html"
  alt="At a Glance" class="voiceoverListLink">At a Glance</a><br />
<a href="http://help.apple.com/ipad/5/voiceover/en/iPad741db878.html"
  alt="Getting Started" class="voiceoverListLink">Getting Started</a><br />
<a href="http://help.apple.com/ipad/5/voiceover/en/iPad743b0e91.html"
  alt="Basics" class="voiceoverListLink">Basics</a>
```

Example: `http://13.94.197.112:9000/assets/examples/rawlists.html`.

## Error handling

Since we decided that the parsing will be handled asynchronously, it is possible that an error occurs of which the web service consumer doesn't know about. Therefore, we implemented better error handling. If an unhandled exception occurs in the parsing process, this will be visible for the consumer as the error message will now be shown in response of the check method.

Before this implementation, the status of a task would not change and the consumer could wait forever.

## NER after writing out the JSON

As the NER lookup is the slowest operation in the process, we decided to make the JSON result accessible before the NER is completed. As soon as the parsing part is done, querying the `result` endpoint will result in the status `DONE_WITHOUT_TAGS`. The returned JSON document will be complete except that the tags array will be empty. As soon as the NER processing is done, the tags array will be filled and the status will be `DONE_WITH_TAGS`.

## Prefix handling

We encountered some problems with documents that contain relative links to other documents. Instead of having to modify all links in a document before submitting it to the system, the prefix of the links can be passed to the system in the **start** endpoint. For example, using `?prefix=http://example.com/` will make sure that relative links (e.g. `subpage/page1.html/`) are retrieved correctly.

In case no prefix is provided in the **start** endpoint, a prefix will be automatically generated based on the URL of the page that contains the table of contents. For example, submitting `http://example.com/subpage/toc.html` without **prefix** parameter will cause the system to use the prefix `http://example.com/subpage/`.

## 4 Application

The application we create is a web API that takes an url of an HTML document. The document is then processed and a JSON compatible with the JSON format used in Onsophic is created. Multiple requests can be issued simultaneously and the status of a document can also be queried.

## 5 Overview of the different meetings

### 5.1 Initial meeting with client (24/02/2016)

- **Present:** Everyone except Feliciaan
- **Goal:** Getting to know the specific details and requirements for the project.
- **Meeting notes:** We met with Davy, the CTO of Onsophic. Onsophic has seven employees, of which two to three are technical employees. The platform that Onsophic produces is an educational platform, aimed at different companies such as: software, retail, ... They are based in Europe & the USA.

The platform offers courses and learning activities. A course exists of modules, each module contains learning activities, e.g. youtube clip, drive document, ... Onsophic hosts nothing by itself, everything is hosted somewhere else. Documents that have to be imported into the platform are for example user manuals about products. The

support can than learn more about the products through the courses. We have to convert our input documents to a series of modules and learning activities.

- **Decisions:** This week we received a sample JSON file and test logins for the On-sophic platform. We will have a meeting with the client weekly, the day before meeting with the teaching staff.

## 5.2 Review with teaching staff (25/02/2016)

- **Meeting notes** Make sure to discuss these topics next time:
  - Overview of architecture
  - Who is doing what
  - Risk list
  - What can go wrong, what problems do you have to tackle first, prioritize
  - What features are we going to offer
  - Use your client
  - Include assumptions about inputs in the report!
  - Agile approach, get a demo ready as fast as possible
  - Planning in a powerpoint, what technologies to work together
- **Decisions:** Next time, come up with a presentation that gives a summary of the above subjects.

## 5.3 Review with teaching staff (10/03/2016)

- **Meeting notes:**
  - Define more risks. Not only encountered events, but also risks of events that may happen in the future.
  - Make sure a good planning for the next 6 weeks is defined and presented on the next meeting (17/03/2016).

## 5.4 Review with client (16/03/2016)

- **Meeting notes:**

- An external NER service can be down. Think about mitigations for this risk.
- Think about possibilities to use our system in a more user-friendly way.

## 5.5 Review with teaching staff (17/03/2016)

- **Meeting notes:**

- A more detailed sprint planning must be added to the progress report.
- A self-reflection on the proposed sprint must be discussed during the next meeting.

## 5.6 Extra internal meeting (28/03/2016)

- **Meeting notes:**

- Several sources were researched, and a categorisation is made to
  1. sources with navigation based on simple HTML list tags;
  2. sources with navigation places in HTML tables;
  3. sources with raw (hierarchical) links.

- **Decisions:**

- Stefaan implements a parser based on category 1.
- Titouan implements a parser based on category 2.
- Wouter implements a parser based on category 3.
- Feliciaan implements the detection of each category and executes the corresponding parser.

## 5.7 Review with client (30/03/2016)

- **Meeting notes:**

- There're a lot of possible formats. Make a list of the format we want to support and provide some examples for it.

- The parsing algorithm should be described more with all relevant constraints.
- Sometimes a *parent* in the hierarchical tree has content too. In that case, make a separate section that contains that content, e.g. an “*Introduction*” section.
- Make a better error handling system, where a user gets notified if things go wrong.
- Units are not yet implemented. Without units, we can not import our JSON into the client’s system.
- Both *Bloomlevels* and *Modalities* have to be implemented.

- **Decisions:**

- Stefaan provides the client with a link to the Azure production server.
- Wouter checks whether the used NER service has a usage limitation.
- The client provides the team with correct user permissions on his platform.

## 5.8 Review with client (13/04/2016)

- **Meeting notes:**

- The proposed categorisation of Bloomlevels is too wide. The client will provide the team with the exact list of keywords that will have to be used in the application.
- Testing is not yet possible because there was an error while deploying on Azure. This will be fixed as soon as possible.
- For the Bloom level detection: the title of an activity is more important than keywords in the content of that activity.

- **Decisions:**

- As temporary solution for the Bloom levels, use only the first two categorisation columns from the proposed levels.
- A draft of the progress report will be sent to the client before next week.
- Set a priority: deploy on Azure.