ÉCOLE POLYTECHNIQUE DE L'UNIVERSITE FRANÇOIS RABELAIS DE TOURS
Spécialité Informatique
64 av. Jean Portalis
37200 TOURS, FRANCE
Tél +33 (0)2 47 36 14 31
www polytech univ-tours fr

# SPECIFICATION

| | | |
|---|---|---|
| **Project :** CDS03 | Maquette détaillée d'un cahier de spécification | |
| **Emitter:** | N. Ragot | **Owner :** EPU-DI |
| **Date of issue :** | 28/10/2015 | |

**Validation**

| Name | Date | Valid (Y/N) | Comments |
|---|---|---|---|
| | | | |
| | | | |
| | | | |

**History of changes**

| Version | Date | Description of the change |
|---|---|---|
| 00 | 11/2008 | Initial version: synthesis of different documents |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# TABLE OF CONTENTS

## SPECIFICATIONS BOOKLET

This document aims to raise the technical and methodological requirements and for the project. After putting the project into context, it will expose state-of-the-art tools to use, analyse the solution at each stage, define test plans through scenarios and test sets and thus validate the proposal.

The actors of this project are:

- the client, which here are Centre for Advanced Renaissance Studies (fr. Centre d'études supérieures de la Renaissance) (CESR), for which a contact is Rémi Jimenes, lecturer and researcher.
- the Project/Product Owner (fr. Maître d'ouvrage) (fr. MOA), who is Jean-Yves Ramel, professor of computer science, director of Laboratory of Fundamental and Applied Computer Science of Tours (fr. Laboratoire d'Informatique Fondamentale et Appliquée de Tours) (LIFAT) and academic tutor for this project.
- the Project Manager / Scrum Master (fr. Maître d'œuvre) (fr. MOE), Théo Boisseau, an engineering student in his final year of study. He decide on the technical means used to design the product by what was defined by the product owner.

The product owner is responsible for representing the client by ensuring that the deadlines are met and that the product conforms. Thus, he is in charge to review documents such as this one.

The implementation phase starts on 4 January and ends with a final presentation around 3 March.

## 1. Context of the project

The client expressed the need for easy-to-use interactive software so that its users, CESR researchers and historians, could create their own scenarios for extracting elements of content (EOCs) from images of historical documents. These historical documents are mainly Renaissance corpora, accessible from the CESR database, and contain mainly printed or manuscript text, illustrations and page ornaments.

To convert these historical books into accessible digital libraries, LIFAT is developing software that participates in a complete processing chain, including layout analysis, text/illustration separation (i.e. segmentation of elements of content), optical character recognition (i.e. OCR) and text transcription. This project focuses on layout analysis and segmentation of elements of content of historical documents.

### 1.1. Objectives

This project aims to propose a new approach based on deep learning neural networks to solve this image segmentation problem.

To this end, the Deep-Agora R&D project aims to build a prototype of an optimisation software capable of extracting textual and decorative elements of content from images of historical documents.

The user should not be responsible for training neural network models. Therefore, several deep learning models can be created and trained to extract the elements of content required in the different use cases of the software.

Due to its nature as a prototype, the system will need to be composed of computational documents combining scripts and good documentation. It must also provide access to training datasets and parameter storage files to reproduce the deep learning models created.

If the objective is achieved, the project can be continued and a scenario creation subsystem can be implemented to deploy the models created within it.

### 1.2. Hypotheses

For this project, we suppose that they are no different typefaces in blocks of text of a document. However there are, so it will be taken into account in the future of this project.

State-of-the-art DL frameworks are not good enough to segment handwritten characters in images of historical documents. If one appears during the development in the deep learning lab, it should definitely be used in the project.

We suppose that the end users will only look for these elements of content:

- Blocks of texts
- Printed and handwritten text-lines
- Handwritten annotation
- Initial capitals
- Banners
- Illustrations/decorations

And will not look for more modern or scientific ornaments, such as:

- photographs
- tables
- graphics
- formulas

Either way, new data sets should be used to train new neural network models.

No other methods than grouping connected black pixels exist to post-process the binary mask of predictions. If this is wrong, then it could be a solution to make state-of-the-art DL frameworks good enough to segment handwritten characters in images of historical documents. Appropriate new data sets with each character labelled individually should be used to train new neural network models. ALTO files could also identify each character by a Glyph tag.

The DL frameworks do not use binarisation algorithms as a pre-processing step. If they all do, the efficiency will not be as good, but there is nothing to be done.

The last version of ALTO must be used. If it wastes time on the project, a former version can be used.

Agora will continue to evolve over the years and new needs may arise. Thus, documentation should be very good in order to ensure a successful takeover of the project.

A long period of time will be devoted to understanding the frameworks, working on the data and training the first model. If it wastes time on the project, the subject should be referred to the product owner.

### 1.3. Methodological

An Agile project management method will be used to create learning loops to quickly gather and integrate feedback. Therefore, the Scrum method should be preferred in which ideology is to:

- learn from experience
- to self-organise and prioritise
- to reflect on gains and losses to continuously improve

Therefore, contact with the product owner should be maintained as much as possible, as it will help me to improve and learn considerably as the project progresses.

To this end, we set sprints with a fixed duration of 2 weeks, which means there are 5 sprints. At least one deliverable, containing an e-mail, should be sent to the product owner at least every two weeks and preferably once a week. During the implementation phase, a meeting to get feedback about the product should be scheduled at the end of each sprint.

All these sprints aim to prioritise and propose different versions of Deep-Agora:

1. The first major release should semantically segment the layout of pages and return for each page a list of tuples each containing an element of content, with its label and its coordinates.
2. The next major release should orient the notebook towards targeted extractions, which should allow multiple algorithms to target more specific types of elements of content. Most of them will most likely be trained on different data sets.
3. During or after the major release above can be produced another one that will export the outputs to ALTO XML files.
4. An optional fourth major release consist of turning the different models into exploitable modules in the software. It includes creating encapsulated systems for managing scenarios and inputs/outputs (I/O) for end users of Deep-Agora.

GitHub will be used for configuration management, by creating two different repositories:

- Deep-Agora, which contains the source code of the project
- Deep-Agora_DOC, which contains all the deliverables of the specification, analysis and modelling part of the semester 9

GitHub can also be used as a project management tool. It offers a similar feature to Trello called Projects, an adaptable spreadsheet that can also integrate with my issues and pull requests on GitHub to help me plan and track my work efficiently.

*Cela inclus les références à des documents annexes tels que le plan d'assurance qualité et/ou de test, etc.*

## 2. General description

### 2.1. Project environment

This project is part of a larger research project between CESR and LIFAT. It is currently being carried out as part of a programme for the regional valorisation of old books (mainly dating from the Renaissance), namely the *Humanist Virtual Libraries* controlled by the CESR.
Within this programme, projects such as TypoRef which aims to identify specimens of similar typical characters, and BaTyr which is a database of illustrations extracted, need software that meets the requirements of this project.

CESR does not have powerful computing machines capable of training deep neural network models, but it has internet, several machines and a large amount of remote and on-premises storage.

Agora, the software developed and published ten years ago by LIFAT to process images of historical documents, is undergoing a complete overhaul in this project. Its technologies need to be updated and, above all, its overhaul should meet the previously unattainable need for simplicity in scenario creation.

Therefore, no takeover of the existing system is planned, as it has to be completely redesigned.

### 2.2. User characteristics

End users of Deep-Agora are all historians of CESR.

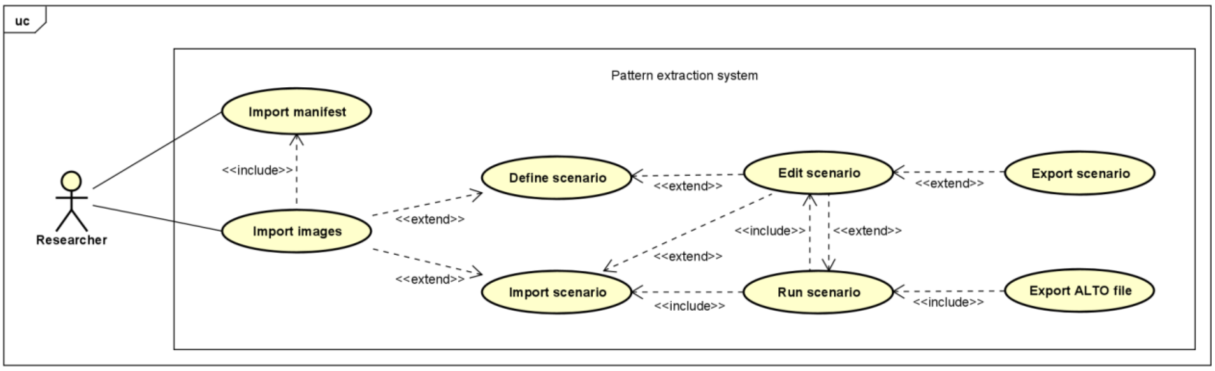
They have a sufficient but moderate command of computer tools. They often use them but need extensive training or solid documentation to use them in the case of advanced tools with complex functions. They did not have a satisfactory experience with Agora, as its interface was too complex. They do not need user access rights to use Agora.

### 2.3. System features

Users will use this software to extract patterns.
For this purpose, they should:

- import a manifest (redirecting to a collection of images)
- import images directly
- define a new scenario
- import an existing scenario from their file system
- edit the scenario by defining operations
- run the scenario to view the extracted content items
- export the scenario results to an ALTO file
- export the scenario to their file system, making it available for import.

In practice, from all the images in a collection, users select a typical one on which they build and test their scenarios to extract elements of content, label them, split them and merge them in an iterative way. They can then save their scenarios and run them on other collections.



### 2.4. General structure of the system

Training deep neural network models is not a task intended for Deep-Agora end users. This part of the project is to be carried out outside the software system, but within the environment, as the engineer's system. It includes data preparation of the datasets found on the internet and the deep learning laboratory where the neural network models are trained.

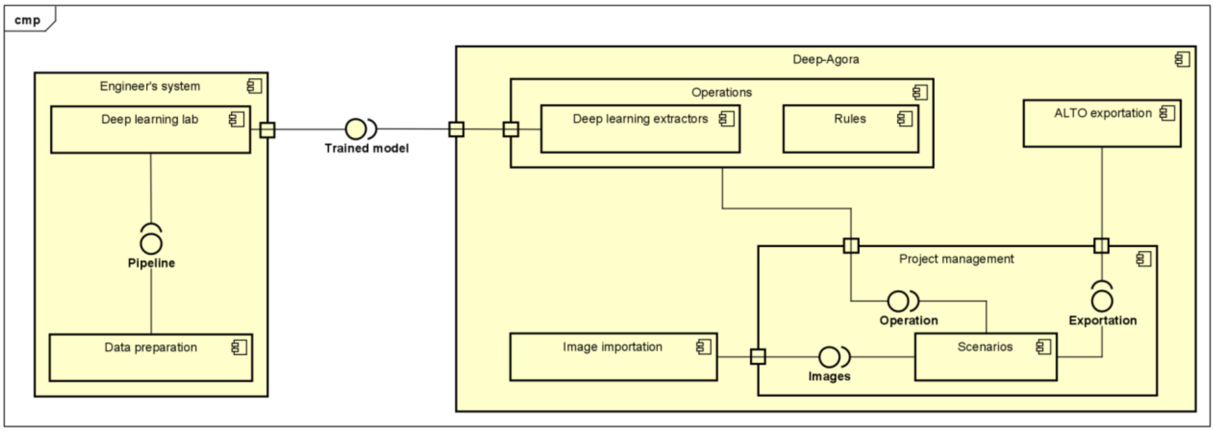
The software itself, Deep-Agora, simply receives trained neural network models and uses them as operations in the scenarios to simply extract elements of content.

Rules are another type of operation that can complete the scenarios with a more descriptive approach, to specifically label or merge elements. This type of operation exists and will be part of Deep-Agora but is not the subject of this project.

The scenarios are managed by projects that deliver the images, provide them with available operations and save their results.

Image importation will provide projects with usable images that are either directly provided or whose IIIF links allow them to be found from a manifest.

ALTO exportation processes to convert the results of the scenarios to an ALTO XML data structure, and save them to ALTO files.



## 3. Description of the external interfaces of the software

### 3.1. Hardware/software interfaces

IIIF links require an Internet connection to process HTTP request to online virtual libraries.

The machine on which the engineer's system deep learning lab will be run should have a GPU to process neural network training faster.

Data sets will be stored in the engineer's system.

### 3.2. Human/machine interfaces

The prototype should be made of computational documents combining scripts and good documentation, such as Jupyter Notebooks.

The HMI of the software should display at least 4 panels:

- Scenario: different operations in iterative order
- Tree of EOC: elements of content organised structurally in a tree
- Existing label: a list of extracted labels
- Current image: a picture of the image being analysed

To build scenarios, operations can be accessed through different dedicated tabs. A File tab is dedicated to the management of the user's project. A project tab is dedicated to configuring it. A scenario Tab is dedicated to clearing it or undoing the last operation performed.

The simplicity of the HIM to create scenarios, reuse them and adapt them to different documents is essential.

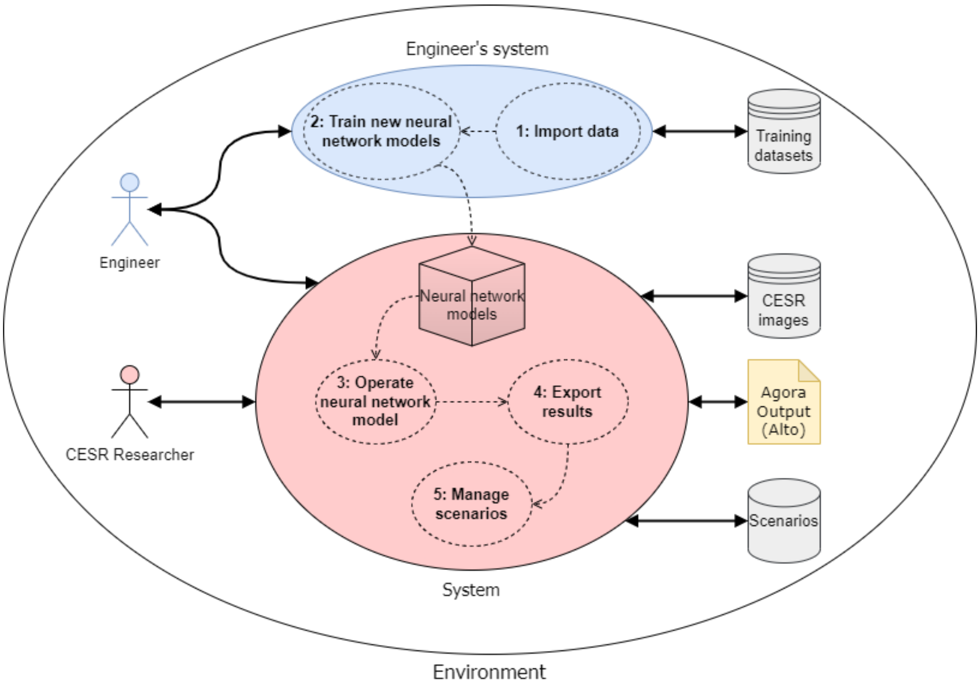
### 3.3. Software/software interfaces

To import images at the beginning of a project, databases are indirectly requested through the use of IIIF links. IIIF links are URLs that return images in response to a standard HTTP or HTTPS request. These links can redirect to internal or external networks.

Trained neural network models are implemented in Deep-Agora manually, by restoring their parameters from storage files.

In the engineer’s system, datasets are downloaded manually through websites and models are trained using a state-of-the-art generic framework for historical document processing.

## 4. Functional specifications



### 4.1. Definition of backlog 1

| | |
|---|---|
| Name | Import data |
| Summary | Implement a pipeline that acquires and prepares a dataset for each neural network model. |
| Priority (primary, secondary, optional) | Primary |
| Inputs | Datasets under different formats and containing different elements of content. **(Cf. …)** |
| Preconditions | Available datasets were downloaded on local storage. |
| Outputs | A class file and a training data folder. The training data folder contains an images folder and a labels folder. |
| Postconditions | The datasets were selected and merged to provide a single correctly structured one. It contains a sufficient amount of samples with the right elements of content for the model.<br>It is broken down into pairs (images, labels) with the same name.<br>The annotated images in the label folder are RGB images with the regions to segment annotated with a different colour for each class.<br>The file containing the classes has one row for each class (including the ‘negative’ or ‘background’ class) and each one has 3 values for the 3 RGB values. Each class must have a different code. |
| Interacting components | • Data preparation component<br>• Deep learning lab component<br>• File system<br>• Datasets from the file system<br>• Engineer’s system |
| Prioritized features list | 1. Acquire datasets from websites **(Cf. …)**<br>2. Select them accordingly to the input specifications of the targeted neural network model<br>3. Convert the images to the right format |

| | |
|---|---|
| | 4. Restructure their labels if they do not respect the postconditions<br>5. Merge the datasets selected<br>6. Reiterate from 2. for each model |
| Specific error handling and implementation | • Some datasets are not available for download<br>➔ Delete them from the list and evaluate again the feasibility of the targeted model |

## 4.2. Definition of backlog 2

| | |
|---|---|
| Name | Train new neural network models |
| Summary | Implement a generic framework for historical document processing to segment images into targeted elements of content. |
| Priority (primary, secondary, optional) | Primary |
| Inputs | A class file and a training data folder. The training data folder contains an images folder and a labels folder. |
| Preconditions | The dataset contains a sufficient amount of samples with the right elements of content for the model.<br>It is broken down into pairs (images, labels) with the same name.<br>The annotated images in the label folder are RGB images with the regions to segment annotated with a different colour for each class.<br>The file containing the classes has one row for each class (including the 'negative' or 'background' class) and each one has 3 values for the 3 RGB values. Each class must have a different code. |
| Outputs | A list of tuples of labels and coordinates. |
| Postconditions | The list of tuples contains only elements of the content with specified labels and their coordinates in the original image. |
| Interacting components | • Data preparation component<br>• Deep learning lab component<br>• File system<br>• Engineer's system |
| Prioritized features list | 1. Decide which model to train<br>2. Acquire the right prepared dataset from the data preparation component for the model<br>3. Split the data set<br>4. Declare the training parameters of the state-of-the-art framework model.<br>5. Train the model<br>6. Validate it or reiterate from 2. |
| Specific error handling and implementation | • The dataset does not dispose of enough samples or is not balanced enough.<br>➔ Communicate the error to the product owner<br>➔ Or pass to another model |

## 4.3. Definition of backlog 3

| | |
|---|---|
| Name | Operate neural network model |
| Summary | Implement a module to operate a trained neural network model as an operation in scenarios. |

| | |
|---|---|
| Priority (primary, secondary, optional) | Primary |
| Inputs | A single image<br>An image folder |
| Preconditions | Images are RGB and historical documents.<br>The neural network model has been validated by the product owner. |
| Outputs | In the file system:<br>• For a single image, the image with bounding boxes shows the elements of content.<br>• For an image folder, vignettes of each element of content are structured in a results folder.<br>In the software:<br>• a list of tuples of labels and coordinates per image. |
| Postconditions | The bounding boxes encapsulate the targeted elements of content on the image.<br>The names of the vignettes in the vignette folder are hierarchically structured and are the same images as the contents of the bounding boxes on the corresponding image. |
| Interacting components | • Operations component<br>• Scenarios<br>• Manifests<br>• File system<br>• Images from manifests and/or CESR |
| Prioritized features list | 1. Make the model extend the deep learning extractor class<br>2. Encapsulate deep learning extractor in the operations class<br>3. Build the image importation class to acquire images via manifest and/or from the file system<br>4. Create the scenario class<br>5. Append the operation to the scenario's list of operations<br>6. Choose the model to extract targeted elements of content<br>7. Restore the right model to operate<br>8. Segment the image and store the list of regions with their IDs in the scenario<br>9. For a single image, create an image with bounding boxes<br>10. For single or more many images, extract vignettes using the bounding boxes and export them to the file system<br>11. Link the output of the last operation to the output of the scenario |
| Specific error handling and implementation | • Segmentation is not efficient enough<br>➔ Retrain the model<br>➔ Revise the evaluation process of the model |

### 4.4. Definition of backlog 4

| | |
|---|---|
| Name | Export results |
| Summary | Edit the outputs of scenarios to convert them into ALTO files. |
| Priority (primary, secondary, optional) | Secondary |
| Inputs | A list of regions (ID, labels and coordinates) per image. |
| Preconditions | Each region has a corresponding vignette in the vignette folder |
| Outputs | An ALTO file |
| Postconditions | The ALTO lists all the elements of content from the input in a hierarchically structured way with their coordinates and their labels. |

| | |
|---|---|
| Interacting components | - Scenarios<br>- File system<br>- Deep-Agora system |
| Prioritized features list | 1. Define an ALTO template<br>2. Create the class ALTO exportation<br>3. Make routines to write regions hierarchically from input<br>4. Link the module to the scenarios class |
| Specific error handling and implementation | - |

### 4.5. Definition of backlog 5

| | |
|---|---|
| Name | Manage scenarios |
| Summary | Create an HIM to manage scenarios. |
| Priority (primary, secondary, optional) | Optional |
| Inputs | Manifests, images and parameters of neural network models |
| Preconditions | All the previous backlogs have been validated by the product owner |
| Outputs | ALTO files and serialised scenarios |
| Postconditions | Scenarios can be created, imported, edited, run and deleted by the end user. |
| Interacting components | - Project management<br>- File system<br>- Deep-Agora system |
| Prioritized features list | 1. Finish defining the scenario class<br>2. Create the project class<br>3. Create an adaptive project directory<br>4. Create an adaptive image subdirectory<br>5. Create an adaptive current image<br>6. Enable directory clearing<br>7. Enable saving of scenarios<br>8. Enable loading of scenarios<br>9. Allow scenarios to be run on all images<br>10. Allow to cancel the last operation on the current image.<br>11. Allow to delete the scenario<br>12. Create HIM<br>13. Enable image zooming |
| Specific error handling and implementation | - |

## 5. Non-functional specifications

### 5.1. Development constraints and design

The state-of-the-art framework for training the neural network model is dhSegment. The programming language is Python and the computational documents are made of Jupyter Notebooks. Jupyter Notebooks can be made of any IDE or Jupyter Lab, however, they use Conda environment. IIIF links are requested by HTTP or HTTPS protocols.

## 5.2. Functional and operational constraints

### 5.2.1. Performance

Segmentation on an image must last a few seconds, such as 2 or 3 maximum. The scenarios will be run frequently, about 3 times per 5 minutes. The software should not be unavailable for more than 5 seconds, except when processing a folder of images

### 5.2.2. Capabilities

The software runs on a single computer. It takes 3 different types of neural network models: text-lines, ornaments and figures. They are implemented manually and on demand. The software itself should be light and only the data from outside the system can consume significant storage. A model can process only one image at a time.

### 5.2.3. Operating modes

As a prototype, it can be started with a Jupyter Notebook file after starting a Jupyter server.
After implementing the HIM, it can be started with a python script. It remains on until the user closes the window.

### 5.2.4. Controllability

Data importation should display data samples before and after pre-processing.
Deep learning lab should display the training parameters, the evolution of the loss live, the number of live epochs and a graphic of the loss at the end of the training and evaluation.
During the prototype part, the results of the deep learning extractor should display the bounding boxes encapsulating the targeted elements of content on the image.
The ALTO exportation and the scenario management respectively display the ALTO file and the serialised scenario produced.

### 5.2.5. Security

The level of confidentiality of the system is non-existent: there is no user access control, no keywords or passwords.

### 5.2.6. Integrity

ALTO files and serialised scenarios are not protected. The end user can save them wherever they want.
The software only connects to the Internet when a manifest requires it. There is no protection.

## 5.3. Maintenance and development of the system

Maintenance of the HIM is palliative (fr. curative), which means it should only be done punctually on specific issues.

Maintenance of the operations and scenarios is curative, which means they should be restored if there is an issue. It should also be perfective to improve efficiency and evolutive since new needs can appear.

# GLOSSARY

Dans cette partie on doit trouver, classés par ordre alphabétique, les définitions des termes courants utilisés, des termes techniques, abréviation, sigles et symboles employés dans l’ensemble du document.

18

## BIBLIOGRAPHY

**There are no sources in the current document.**

Cette dernière partie recense les références techniques sur le projet sur :

- les documents relatifs à l’existant et à l’environnement ;
- les documents sur les méthodes et algorithmes cités ;
- les documents bibliographiques (internes et externes) ;
- les sources d’obtention des documents.

Maquette détaillée d’un cahier de spécif

# INDEX

Cette partie indique les pages où sont traités et mentionnés les sujets et les termes les plus importants du document.

**Aucune entrée d'index n'a été trouvée.**