EUROPEAN COMMISSION
Research Executive Agency (REA)
Inclusive, Innovative and Reflective Societies

Project acronym:	SIMPATICO
Project full title:	SIMplifying the interaction with Public Administration Through Information technology for Citizens and cOmpanies
Call identifier:	EURO-6-2015
Type of action:	RIA
Start date:	1 March 2016
End date:	28 February 2019
Grant agreement no:	692819

D2.1 – Basic Version of Text and Workflow Adaptation

WP2	Interaction Adaptation and Personalisation
Due Date:	28/02/2017
Submission Date:	28/02/2017
Responsible Partner:	University of Sheffield (USFD)
Version:	1.0
Status:	Final
Author(s):	Lucia Specia(USFD), Carolina Scarton (USFD), Gustavo Paetzold (USFD), Sara Tonelli (FBK), Alessio Palmero Aprosio (FBK), Michele Trainotti (FBK), Raman Kazhamiakin (FBK), Vincenzo Savarino (ENG)
Reviewer(s):	Unai Lopez Novoa (DEUSTO) and Matteo Gerosa (FBK)
Deliverable Type:	R: Report
Dissemination Level:	PU: Public

Simpatico

Version History

Version	Date	Author	Partner	Description
v0.1	29/11/2016	Lucia Specia	USFD	Initial TOC and brief description of the objectives of the deliverable.
v0.2	06/02/2017	Lucia Specia, Carolina Scarton, Gustavo Paetzold, Sara Tonelli, Alessio Palmero Aprosio, Michele Trainotti, Raman Kazhamiakin, Vincenzo Savarino	USFD, FBK, ENG	First draft for discussion.
v0.3	10/02/2017	Vincenzo Savarino	ENG	Citizen Data Vault (CDV) section added.
v0.4	17/02/2017	Lucia Specia, Carolina Scarton, Gustavo Paetzold	USFD	Version for internal review.
v0.5	27/02/2017	Lucia Specia, Carolina Scarton, Gustavo Paetzold	USFD	Version for final review.
v1.0	28/02/2017	Marco Pistore	FBK	Final quality check.

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Table of contents

List of figures

List of tables

Glossary

API	Application Programming Interface
CBOW	Continuous bag-of-words
CDV	Citizen Data Vault
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DSL	Domain-Specific Language
EN	English Language
ES	Spanish Language
EU	European Union
GL	Galician Language
HTML	HyperText Markup Language
ID	Identity
IT	Italian Language
JSON	JavaScript Object Notation
LS	Lexical Simplification
N/A	Not applicable
NLP	Natural Language Processing
PA	Public Administration
PDS	Personal Data Store
POS	part-of-speech
REST	REpresentational State Transfer
SCC	Sheffield City Council
SS	Syntactic Simplification
TAE	Text Adaptation Engine
TBD	To Be Defined
TF-IDF	term frequency - inverse document frequency
WAE	Workflow Adaptation Engine
XPath	XML Path Language

Executive summary

This document is the deliverable "**D2.1 – Basic version of text and workflow adaptation**" of the European project "SIMPATICO - SIMplifying the interaction with Public Administration Through Information technology for Citizens and cOmpanies" (hereinafter also referred to as "SIMPATICO", project reference: 692819).

The goal of SIMPATICO project is to improve the experience of citizens and companies in their daily interactions with the public administration by providing a personalized delivery of e-services based on advanced cognitive system technologies. This will be achieved through a solution based on the **interplay of language processing, machine learning and the wisdom of the crowd** to change for the better the way citizens interact with the Public Administration.

The main objective of WP2 - Interaction adaptation and personalization - is to devise a **self-evolving framework to adapt texts and workflows according to user needs**. This document covers work done for WP2 in the first 12 months of the project, with a brief report on the state of the art, key functionalities, architecture, interfaces and next steps for each of the following software components: **Text Adaptation Engine**, including **Text Profiling and Analytics** (in the languages of the project: English, Italian and Galician/Spanish), **Workflow Adaptation Engine** and **Citizen Data Vault**.

1 Introduction

An important part of the SIMPATICO solution is to adapt texts and workflows according to the user needs. This involves a personalised approach based on individual user profiles. Managing personal data and ensuring its privacy and security are thus also a critical requirement. These goals are covered under WP2 - Interaction adaptation and personalization. More specifically, the objectives of WP2 are to:

1. Analyse and annotate text profiles, i.e. select and prepare texts for adaptation.
2. Design operations to adapt text and workflows, based jointly on the requirements of users and characteristics of the texts and workflows, while making sure these operations can be generalised to various language constructions, texts and users.
3. Apply such operations seamlessly while ensuring continuous self-improvement of the framework with its use.
4. Manage user profile and personal data ensuring its privacy and security.

During the first 12 months of the project, WP2 focused on objectives 1, 2 and 4 to produce a basic version of Text and Workflow Adaptation components (objectives 1 and 2), and their connection to the Citizen Data Vault component (objective 4). This deliverable focuses on the description of software components (key functionalities, architecture and interfaces), but also presents, for each component, a brief survey of the state of the art and, where applicable, provides an initial evaluation of such components. For each component, we also provide a discussion about the next planned steps.

Current work towards **objective 1** is described in **Section 3**. This task involves analysing texts with the purpose of identifying various features that may be relevant for effective text adaptation. Such features include the readability level of the text, the type of reader it has been originally written for. At a later stage, the goal is to analyse metadata (date, source, domain, etc.), and linguistic information (number of clauses or depth of syntactic tree for sentence, concreteness level for word, etc.). Once the text is analysed, features judged relevant will be used to enrich the text .

Work done for **objective 2** is described in **Sections 2** and **4**. **Section 2** covers the initial versions of the Text Adaptation Engine for Lexical Simplification (Section 2.1) and for Syntactic Simplification (Section 2.2). These initial versions address the design and learning of operations for text adaptation. For lexical simplification, both machine learning and rule-based approaches were designed. For syntactic simplification, a basic set of adaptation rules was handcrafted to cover operations that apply to all project languages. Additional functionalities for non-native speakers, such as search in dictionaries and calls to machine translation systems, were also included. Most of the work covered so far was designed using general language resources, without any personalisation to specific users or groups of users. The evaluation of these components, where possible, uses data from our three use cases. These components were developed for the three languages covered by the project's use cases: English, Italian and Galician. However, the preliminary evaluation showed that the results for Galician were unsatisfactory due to the lack of language resources for this language. Therefore we have also initiated work on these components for Spanish, as our fallback strategy. Spanish is one of the two languages used by the Xunta PA, since all their online content is produced in two languages.

Section 4 covers current work on the Workflow Adaptation Engine. This component breaks the interaction process and the forms to be filled in blocks that codify information elements that the user

needs to provide. Basic rules were defined that present to the user only the elements that are relevant for a particular interaction.

Work done towards **objective 4** is covered in **Section 5**. This section describes the Citizen Data Vault component. This component allows citizens to manage and share their own personal data. It enables citizens to control how the data should be accessed and by whom. It represents a safety vault where the user's personal data, preferences and profile details are preserved and exploited only by authorized components (e.g. the adaptation engine).

2 Text Adaptation Engine

The Text Adaptation Engine of SIMPATICO is composed of two modules: the Lexical Adaptation and the Syntactic Adaptation. They have the goal of addressing the lexical (word-level) and syntactic (sentence structure) complexity of public administration service pages, respectively.

These two modules are independent, and can work simultaneously depending on the type of simplification that a given user requires. They can be used in conjunction in order for a complex sentence to have both of its lexical and syntactic complexity reduced. A user can, for example, request a syntactically simplified version of a complex sentence, then request lexical simplifications for words that they find complex. We describe these modules in more detail in what follows.

2.1 Lexical Adaptation

2.1.1 State of the art

The goal of a Lexical Simplification (LS) system is to replace the complex words in a text with simpler alternatives, without compromising its meaning or grammaticality. The LS task is often addressed as the series of steps in Figure 1.

The steps in Figure 1 can be described as:

- **Complex Word Identification:** Decide which words of a given sentence may not be understood by a given target audience and hence must be simplified.
- **Substitution Generation:** Find words or expressions that share at least one meaning with the target complex word.
- **Substitution Selection:** Decide which of the generated candidate substitutions can replace the complex word without compromising the sentences grammaticality or meaning.
- **Substitution Ranking:** Rank the remaining candidate substitutions of a given complex word by their simplicity.

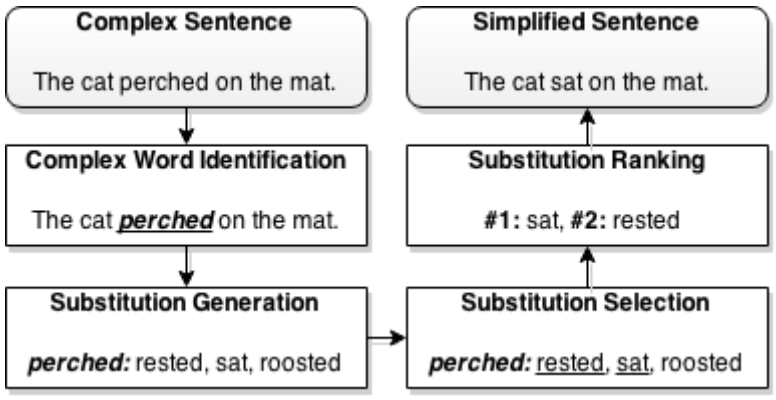

Figure 1 – Lexical Simplification Pipeline

Researchers in the field have been addressing each of those tasks in various different ways. The most popular approaches from literature have been surveyed and compared in performance by (Paetzold & Specia, 2016a). We take as state of the art solutions those which have performed best in their benchmarks. In the Sections that follow, we describe the most effective solutions in literature for each individual task. Please notice that this is the same state of the art summary provided by for the

D5.1 deliverable. It is important to highlight that the description of the state of the art refers only to English. For Italian and Galician there have been neither specific tools nor benchmarks to measure and compare the performance of different LS approaches. We assume that, when a state of the art approach is unsupervised and does not rely on language-specific resources, it may be applied also to these two languages.

Complex Word Identification

The current state of the art Complex Word Identification approaches are the ones introduced by (Paetzold & Specia, 2016b) and (Ronzano et. al., 2016), which employ ensembles: a method where various models are combined. The LEXenstein framework (Paetzold & Specia, 2015), gives access to these approaches.

Substitution Generation

The current state of the art Substitution Generation approaches are the ones introduced by (Glavas & Stanjer, 2015) and (Paetzold & Specia, 2016c), which extract candidates similar to complex words using different types of word embedding models. These strategies are also included in the LEXenstein framework.

Substitution Selection

The current state of the art Substitution Selection approach is the one introduced by (Paetzold & Specia, 2016), which employs a ranking technique called Unsupervised Boundary Ranking that learns a model from unannotated data. This strategy is made available by the LEXenstein framework.

Substitution Ranking

The current state of the art Substitution Ranking approaches are the ones introduced by (Horn et. al., 2014), (Paetzold & Specia, 2015) and (Paetzold & Specia, 2016), which employ supervised models that learn from annotated data. These strategies are included in the LEXenstein framework.

For more details on the aforementioned approaches, please refer to the deliverable D5.1, Section 5.1.1.

2.1.2 Short summary of key functionality

In SIMPATICO, the Lexical Adaptation technologies have the goal of removing the lexical complexity inherent to public administration service webpages by replacing complex words and expressions with simpler alternatives that suit the users' needs.

The lexical adapters of SIMPATICO work with the syntactic adapters in an effort to make texts easier to comprehend to the users of municipality web services in order to make them more streamlined and less expensive to offer.

Throughout the project development period pertaining to this deliverable, a number of new features and improvements were implemented for the lexical adapters of SIMPATICO:

- The lexical adapter for English replaced by a retrofitted context-aware simplification candidate generator, an unsupervised state of the art candidate selector and a novel supervised neural candidate ranker.
- The lexical adapter for Italian was replaced by a context-aware model for generation accompanied by an unsupervised selector. A second, rule-based version of the lexical adapter for Italian has also been implemented for comparison.

- A lexical adapter for Galician was created using newly created corpora.

In the Sections that follow, we describe in detail the Lexical Adaptation modules of SIMPATICO.

2.1.3 Architecture

The lexical simplifiers used in the SIMPATICO project will follow the same architecture illustrated in Figure 1, since it has been shown to be a successful model for the task. There will be one lexical simplifier for each language addressed in the project, which are English, Italian and Galician.

Because of the different degrees of data availability of each language, each simplifier will use a unique solution for the task. We describe the architecture of each simplifier in the Sections that follow.

2.1.3.1 The English Lexical Simplifier

English is the most widely studied target language in the topic of Lexical Simplification. Because of its popularity and large speaker demographic traits, it is also the one that has the largest array of resources available for the creation of lexical simplifiers. Consequently, the English lexical simplifier created for the SIMPATICO project is the most sophisticated among the three. We describe it in what follows.

Substitution Generation

The Substitution Generation approach used by the English lexical simplifier extracts candidate substitutions for complex words by using the Newsela corpus (https://newsela.com/data) and a retrofitted context-aware word embeddings model with 1300 dimensions that employs the CBOW (bag-of-words) model introduced by (Mikolov et. al., 2013).

The Newsela corpus (version 2016-01-29.1) contains 1,911 original news articles in their original form, as well as 4 or 5 versions simplified by trained professionals to different reading levels. In has a total of 10,787 documents, each with a unique article identifier and a version indicator between 0 and 5, where 0 refers to the article's in its original form, and 5 to its simplest version.

To employ the Newsela corpus in Substitution Generation, we first produce sentence alignments for all pairs of versions of a given article. To do so, we use paragraph and sentence alignment algorithms of (Paetzold & Specia, 2016d). They align paragraphs with sentences that have high TF-IDF similarity, concatenate aligned paragraphs, and finally align concatenated paragraphs at sentence-level using the TF-IDF similarity between them. Using this algorithm, we produce 550,644 sentence alignments.

We then tag sentences using the Stanford Tagger (http://nlp.stanford.edu/software/tagger.shtml), produce word alignments using Meteor (http://www.cs.cmu.edu/~alavie/METEOR). To extract candidates, we first consider all aligned complex-to-simple word pairs as candidates. Then we filter them by discarding pairs which: do not share the same POS tag, have at least one non-content word, have at least one proper noun, or share the same stem. After filtering, we inflect all nouns, verbs, adjectives and adverbs to all possible variants.

We produce complementary candidates with the aforementioned embeddings model. First we train the model by tagging a corpus of 7 billion words taken from various sources with universal POS tags, which cluster all inflections of verbs, nouns, adjectives and adverbs into one POS tag each. Then we employ the word2vec toolkit to train a typical CBOW model over the corpus. Finally, we retrofit the model over the synonymy relations in WordNet (https://wordnet.princeton.edu) using the approach of (Faruqui et. al., 2015), which approximates in the embedding model's feature space words that share some sort of relation.

With the model at hand, and given a complex word in context, the generator considers as candidate substitutions the 10 words which have the shortest cosine distances to the complex word in the embeddings model. In our approach, we discard any candidates that do not have the same universal POS tag of the complex word, or that are morphological variants of the complex word.

Substitution Selection

For Substitution Selection, the English lexical simplifier employs a technique called Unsupervised Boundary Ranking, which was previously described in Section 2.1.3. First, it acquires a set of simplification problems composed each by a sentence with a target complex word in it. Then it employs the aforementioned Substitution Generation approach to produce 10 candidate substitutions for the target complex word in each problem. Finally, it extracts features of the complex word and candidates and trains a linear binary classification model over the data by assigning label 1 to the complex word of each instance, and 0 to all candidate substitutions. This process learns a boundary between positive and negative examples.

Once the model is trained, it can perform selection. When the selector receives a simplification problem along with a set of candidates, it ranks the candidates according to how far they are from the negative examples of the model, then discards the 50% of candidates with the worst rankings. Figure 2 illustrates this approach.

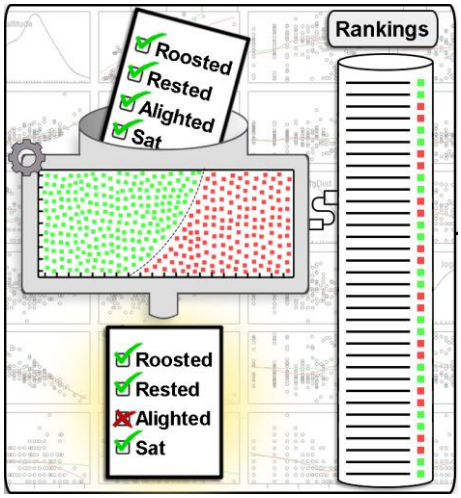

Figure 2 – Unsupervised Boundary Ranking model

Substitution Ranking

For the final step of the pipeline, the English lexical simplifier ranks candidates using a novel supervised Neural Ranking model (Paetzold and Specia, 2017). In order to be trained, the model requires manually annotated data, where each instance is composed by a sentence with a target complex word, and a series of equivalent candidate substitutions ranked by their simplicity. We use the BenchLS dataset for this purpose (Paetzold and Specia, 2016a). Notice that this resource is only available for English, which is why the simplifiers for Italian and Galician use a different Substitution Ranking approach.

Figure 3 illustrates the architecture of the Neural Ranking model employed by the English lexical simplifier. It is composed of three hidden layers with eight nodes each. In order to train the model, we first create one training instance for each distinct pair of candidates in each problem of the dataset used. The instances are composed by a set of features calculated for each candidate, and the label is the difference between the candidates simplicity ranks. Once the instances are produced,

they are passed to the Neural Network, and the error between the label and the predicted value is backpropagated through the network in order for the model to achieve learning. This process is repeated for 500 iterations. At the end of the training procedure, the Neural Ranking model is able to estimate the difference in simplicity between two candidate substitutions.

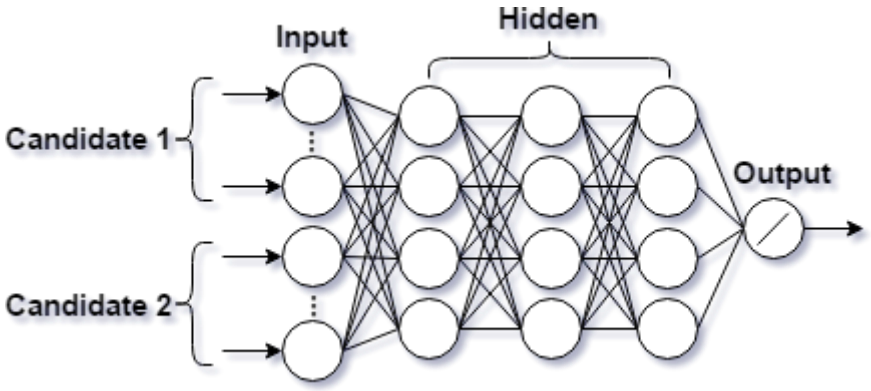

Figure 3 – Neural Ranking regression model for Substitution Ranking

Once the model is trained, then the ranker can solve unseen ranking problems. When presented with a set of candidate substitutions to rank, it first estimates the simplicity difference between all possible pairs of candidates. It then calculates a score for each candidate by summing the differences estimated between itself and the remaining candidates. Finally, it ranks the candidates by simplicity using the score calculated for each one of them: the lower the score, the simpler the candidate.

2.1.3.2 The Italian Lexical Simplifier (v.1)

The Lexical Simplification approach used for Italian greatly resembles the one previously presented for English. However, unlike for English, there are not as many linguistic resources for the Italian language that could be incorporated into a lexical simplifier. Because of that, we opt for a slightly modified array of solutions for the task, which we describe in what follows.

Substitution Generation

While the English lexical simplifier extracts candidates from both parallel complex-to-simple corpora and retrofitted context-aware word embedding models, the Italian simplifier uses as source of candidates only a standard context-aware word embeddings model. This is due to the fact that there are no examples of professionally produced simplification databases as large as the Newsela corpus.

The context-aware embeddings model used for Italian is trained over a corpus of 1.62 billion words tagged with universal POS tags by the Stanford Tagger. It employs the Skip-Gram model of (Mikolov et. al., 2013) and uses 300 word vector dimensions. The corpus contains data from news, Italian Wikipedia, subtitles, but also domain-specific data such as several administrative documents and the official documents issued by the Italian Parliament. Like the Substitution Generation approach for English, given a complex word in a sentence, it selects as candidate substitutions the 10 closest words in the model that share the same POS tag and are not a morphological variant.

Substitution Selection

Since the Substitution Selection approach used by the English lexical simplifier is unsupervised, it can be easily adapted to the Italian language. We train an Unsupervised Boundary Ranking model using the same parameters described in Section 2.2.1.2, except the training dataset used is composed of

100 lexical simplification problems automatically extracted from use case webpages written in Italian.

Substitution Ranking

Since the datasets available for Italian with manually annotated Lexical Simplifications are very small and meant mainly for evaluation purposes (Brunato et al., 2015; Tonelli et al., 2016), we resort to an alternative unsupervised ranking approach.

The approach used is called rank averaging. When presented with a complex word in a sentence along with a set of candidate substitutions, the Italian ranker will first calculate a series of features for each candidate. After that, it will produce one ranking for each feature calculated. Finally, it will average the ranks obtained by each candidate, then use this average to re-rank them. This approach has shown to be a reliable alternative to supervised models (Glavas and Sanjer, 2015).

The Italian Lexical Simplifier (v.2)

For comparison, we also implemented a second version of the Italian Lexical Simplifier based on hand-crafted rules. Specifically, the **Substitution Generation** step is performed by matching a given Italian word with a lexicon created by merging Italian WordNet (Bentivogli and Pianta, 2005) and other custom dictionaries. This returns, for the given word, a list of synonyms. These are further **ranked by frequency** based on the freely available "Corpus e Lessico di Frequenza dell'Italiano Scritto"[1]. The assumption behind this choice is that word frequency is a proxy for simplicity. The motivation for using this simpler approach is that it was not possible to find resources for Italian (corpora, training data, etc.) that would allow us to build the same approach used for the English lexical simplifier.

2.1.3.3 The Galician Lexical Simplifier

The Galician language suffers from an even greater scarcity of resources for Lexical Simplification. Because of that, we had to resort to an even simpler approach than that used for Italian.

Substitution Generation

Since there are no examples of reliable POS taggers that produce universal POS tags for the Galician language, it employs a traditional word embeddings model for Substitution Generation. It is trained with the help of word2vec, using a CBOW model and 300 dimensions over a corpus of 1 million words compiled from numerous sources. Given a target complex word in a sentence, it selects as candidate substitutions the 10 closest words in the model that are not morphological variants.

Substitution Selection

For Substitution Selection, it also employs the Unsupervised Boundary Ranking approach using the same parameters described in Section 2.2.1.2. The model is trained over a set of 100 simplification problems automatically extracted from use case webpages written in Galician.

Substitution Ranking

Since there are no examples of manually annotated lexical simplification problems for Galician, we employ the same Substitution Ranking used for the Italian lexical simplifier: rank averaging. The approach is the same described in Section 2.2.2.3, except the features use resources for the Galician Language.

[1] http://linguistica.sns.it/CoLFIS/Home.htm

2.1.3.4 The Spanish Lexical Simplifier

As a fallback strategy, we developed Lexical Simplifier for Spanish, for which more resources are available. We employ the same **Substitution Generation**, **Selection** and **Ranking** strategies used for the Galician version. Additionally, it also uses a CBOW embeddings model of 300 dimensions trained with word2vec, an Unsupervised Boundary Ranking model trained over 100 automatically extracted simplification problems, and rank averaging.

2.1.4 Summary of Results

We compared our English lexical simplifier, which we name NNLS for simplicity, to previous work. We compare them against all LS approaches featured in the benchmarks of (Paetzold & Specia, 2016a), which are the Devlin, Biran, Yamamoto, Horn, Glavas and Paetzold simplifiers. These simplifiers extract candidates from WordNet, Wikipedia and Simple Wikipedia articles, Merriam dictionary, sentence-aligned Wikipedia and Simple Wikipedia articles, typical word embeddings and context-aware word embeddings, respectively. They rank candidates using word frequencies, hand-crafted simplicity metrics, Support Vector Machines, rank averaging and Supervised Boundary Ranking, respectively. They are all available in the LEXenstein framework.

We use two common evaluation datasets for LS: BenchLS, which contains 929 instances and is annotated by English speakers from the U.S, and NNSEval (Paetzold and Specia, 2016c), which contains 239 instances and is annotated by non-native English speakers. Each instance is composed of a sentence, a target complex word, and a set of gold candidates ranked by simplicity. The evaluation metrics used are Accuracy (A), which is the proportion of instances in which the target word was replaced by a gold candidate, and Precision (P), which is the proportion of instances in which the target word was either replaced by a gold candidate or not replaced at all.

Table 1 shows the results of our evaluation. As it can be noticed, NNLS, the English lexical simplifier developed for the SIMPATICO project, outperforms all other simplifiers by a considerable margin in all datasets and metrics used (Precision and Accuracy).

Simplifiers	**BenchLS**		**NNSeval**	
	Precision	**Accuracy**	**Precision**	**Accuracy**
Devlin	0.309	0.307	0.335	0.117
Biran	0.124	0.123	0.121	0.121
Yamamoto	0.044	0.041	0.444	0.025
Horn	0.546	0.341	0.364	0.172
Glavas	0.480	0.252	0.456	0.197
Paetzold	0.423	0.423	0.297	0.297
NNLS	0.642	0.434	0.544	0.335

Table 1 – Lexical Simplification performance comparison results for English

As for Italian, we evaluate the two versions of the simplifier (the unsupervised one based on the LEXenstein framework and the rule-based version) using the sentence pairs from the SIMPITIKI corpus (Tonelli et al., 2016) that contain lexical simplifications of single tokens (Table 2). This small test set includes 219 simplifications. In this evaluation, we consider the simplification correct if, among the top-5 options suggested by the system, one matches the "gold" (i.e. manual) simplification. We evaluate our approach using the same metrics considered for English: precision and accuracy. However, this evaluation is different w.r.t. the English one because in this case we have only one "gold" simplification provided by a human annotator, while for English the gold standard contains 5 ranked simplifications.

	Precision	**Accuracy**
LEXenstein framework	0.20	0.19
Rule-based tool	0.30	0.11

Table 2 – Evaluation of Italian lexical simplification

The evaluation confirms that rule-based systems tend to outperform machine learning-based approaches in terms of precision, but suffer a recall drop because they rely on external vocabularies and resources that may not cover the domain of interest. The unsupervised approach is less precise but has a better coverage. In our case, the unsupervised tool suggests a simplification for almost any word, provided that it is present in the corpus used to build the word embeddings. For the next project steps, it will be necessary to take a decision on which approach suits best SIMPATICO requirements. A possible solution could be to combine the two approaches in a cascade fashion.

Since there is no evaluation datasets available for Galician, we resorted to a small human evaluation of these simplifiers. Table 3 illustrates some examples of the judgments made by three native speakers of Galician for a series of simplifications made by the previously described simplifiers. Annotators were instructed to judge a simplification as "Bad" if it is not simpler than the complex word, or if it compromises the grammaticality and/or meaning of the sentence, and "Good" otherwise.

Sentence	**Complex**	**Simple**	**Verdict**
Os líquens que medran no residuo asfáltico suxiren a sua baixa toxicidade, que cabería comparar á dunha autoestrada.	medran	crecen	Good
Os líquens que medran no residuo asfáltico suxiren a sua baixa toxicidade, que cabería comparar á dunha autoestrada.	residuo	disolvente	Bad

Os líquens que medran no residuo asfáltico suxiren a sua baixa toxicidade, que cabería comparar á dunha autoestrada.	suxiren	indican	Good
Os líquens que medran no residuo asfáltico suxiren a sua baixa toxicidade, que cabería comparar á dunha autoestrada.	toxicidade	urina	Bad
Os líquens que medran no residuo asfáltico suxiren a sua baixa toxicidade, que cabería comparar á dunha autoestrada.	comparar	compara	Bad
Os líquens que medran no residuo asfáltico suxiren a sua baixa toxicidade, que cabería comparar á dunha autoestrada.	autoestrada	ap-9	Bad

Table 3 – Manual evaluation of Galician lexical simplifications

Our small scale evaluation showed that most of the simplifications suggested are not acceptable. This was part of the reason behind the decision to switch to Spanish as a language for the Xunta de Galicia use case. Further developments and evaluation for Spanish will be performed in the next months, before the pilot evaluations.

2.1.5 Interfaces

In this Section we describe the part of the Text Adaptation Engine API pertaining to the Lexical Adaptation module. Table 4 describes a summary of the Lexical Adaptation function of the TAE API.

Method	/lexical
Type	POST
Description	Receives as input a target complex word, the sentence in which it was found, the index of the complex word in the sentence, and the language in which the sentence is written. As output, it returns a simplified sentence, the replacement found for the complex word, and its index in the simplified sentence.

Table 4 – Lexical Adaptation API

2.1.6 Links

For more details about the input and output parameters pertaining to this API function, one can resort to its Swagger specification at:

https://app.swaggerhub.com/api/ghpaetzold/SIMPATICO_TAE_Lexical_API/0.0.1

One can find the code for the Lexical Adaptation API at the following github project page:

https://github.com/SIMPATICOProject/SimpaticoTAEServer

2.1.7 Next Steps

A number of new models and features are planned to be developed in the upcoming months of the SIMPATICO project. They aim at both improving the quality of simplifications produced (better models), minimizing the number of incorrect simplifications (filtering), and making the models more personalised (based on user profile information, dedicated data, and user feedback).

For the Spanish simplifier, we plan to employ the Stanford Tagger[2] in order to annotate a corpus of raw text so that a context-aware candidate generator can be implemented.

The model improvements will come from more advanced methods based on machine learning. The higher confidence of the models will come from flexible simplification filtering rules that act as complementary Substitution Selection approaches capable of impeding a word to be replaced by a candidate simplification judged complex by a given user.

The personalisation will happen by devising adaptations that allow for the users' personal preferences to be more effectively captured by the simplification models. In the case of models which are trained over annotated data, such as the supervised ranker of the English lexical simplifier and the Substitution Selection approach employed by all simplifiers, we will increment their input with features that describe the users' profile, such as age band, proficiency level, native language and education level. We will also deal with incorporating user feedback once the TAE starts to be tested by our users.

2.2 Syntactic Adaptation

Syntactic simplification aim to simplify syntactic constructions that are considered complex to a given audience. For example, low literacy readers may have problems to read long sentences with several clauses or with several words before the main verb. This happens because low literacy readers can have an overload of the working memory while reading a long passage. Breaking long sentences or re-arranging its words through syntactic simplification can make the sentence more understandable.

In this section we present the SIMPATICO solution for syntactic adaptation. Section 2.2.1 summarises the state-of-the-art work for syntactic simplification. In Section 2.2.2, we present a short summary of SIMPATICO technologies for syntactic simplification. In Section 2.2.3, we present the detailed architecture of the tools for each language addressed by the SIMPATICO project. Section 2.2.4 contains the description of the interface of the syntactic adaptation TAE module. Finally, sections 2.2.5 and 2.2.6, respectively, presents links for our tools and summarises the next steps.

2.2.1 State of the art

In Deliverable 5.1, sections 6.1.2 and 6.1.4, we presented a detailed list of the state-of-the-art for syntactic simplification.

In summary, there are two approaches that are employed for syntactic simplification: rule-based and statistical models.

Rule-based systems are the most popular approach and apply a mix of handcrafted and automatically inferred syntactic modifications to a sentence's structure (Siddharthan, 2004). The architecture of such systems is usually composed by three phases: *Analysis*, *Transformation* and *Regeneration*.

The *Analysis* phase encompasses a set of pre-processing steps (e.g. such as the identification of sentences with discourse markers that can indicate conjoint clauses). The *Transformation* phase applies the simplification rules and the *Generation* phase reconstruct the simplified sentence(s) with the correct verbs and correct casing. A detailed explanation of this approach can be found in Section 6.1.2 of Deliverable 5.1.

[2] http://nlp.stanford.edu/software/tagger.shtml

Text simplification can also be viewed as a "translation" task where the original text is "translated" into the simplified text (Shardlow, 2014). Applications of this type use SMT frameworks (e.g. MOSES (Koehn et al., 2010)), in order to build the simplification systems from parallel original-simplified data. Other statistical approaches have explored (quasi-)synchronous grammars for text simplification that use the tree-to-tree transformations between original and simplified texts (Siddharthan, 2014). It is worth mentioning that approaches using statistical models can cover all level of simplification, not just syntactic. However, such systems usually need large parallel corpora to be trained with reliable confidence. More details about this approach can be found in Section 6.1.4 of Deliverable 5.1.

Publicly available software tools

Although significant research has been done in syntactic simplification, there are no freely available software for that. Some online tools either only address parts of the simplification process (focusing mainly on the analysis phase of Siddharthan's framework). Others are not freely available or do not meet the requirements of SIMPATICO solutions.

For English

The only tool for syntactic simplification available is a demo developed by Advaith Siddharthan that implements his framework for syntactic simplification.[3] However, such demo tool only allow one to copy and paste small portions of text on the actual demo website, it does not have an API for external access.

Therefore, for SIMPATICO we implemented our own syntactic simplification tool. Our solution is rule-based due to the lack of in domain data for training a statistical system for text simplification. In addition, it focuses on rules that apply to all languages of the project.

For Italian

The only system developed so far for Italian is ERNESTA (Enhanced Readability through a Novel Event-based Simplification Tool) (Barlacchi and Tonelli, 2013), which focuses on syntactic simplification of children's stories. Since no large training data is available in this language, this system applies a rule-based approach to simplify a story into a set of simple statements, containing only relevant information to the story. This simplification strategy is very radical in that all non-mandatory parts of a sentence are discarded, only main clauses are retained, and verbs are transformed in the present tense. This approach was chosen because the tool was meant for children, but in our case it would remove parts of the sentences that are important from a semantic point of view. Therefore, we re-use some modules of ERNESTA to build the Italian syntactic simplifier, but we significantly revise the global simplification strategy.

For Spanish and Galician

There was no available research addressing syntactic simplification for Galician and, therefore, our solutions are the new state of the art.

For Spanish, although previous work have addressed the topic (e.g. Saggion et al., 2015; Stajner et al. 2015), there is no tool freely available to be used.[4] Therefore, for the SIMPATICO project purposes, we decided to implement our own technologies, that will be publicly available.

[3] http://homepages.abdn.ac.uk/cgi-bin/cgiwrap/csc323/RegenT/demo.cgi

[4] Saggion et al. (2015) mention a demo system that is only available on request.

2.2.2 Short summary of key functionality

The Syntactic Adaptation solution within the SIMPATICO project aims to make sentence structures simpler. Together with the Lexical Adaptation module, they are the core of the Text Adaptation engine that will generate simplified versions of the public administration's webpages.

Syntactic simplification is a much less explored topic than lexical simplification, therefore, the syntactic adaptation solutions for all languages (except Italian) in the SIMPATICO project were developed from scratch:

- **English:** we made use of existing resources and tools for NLP in order to implement the syntactic rules. Since such resources and tools were already previously validated, the performance of our simplifier for English is higher than for the other languages. The English syntactic simplifier is composed of *Analysis*, *Transformation* and *Generation* modules and it receives a sentence as input and returns either the same sentence or a simplified version. For English, we have implemented simplification rules for conjoint clauses, relative clauses, appositive phrases and passive to active voice.
- **Italian:** we modified the ERNESTA tool (Barlacchi and Tonelli, 2013) in order to adapt it to the syntactic simplification strategy needed in the project. We also had to develop specific NLP tools to support the simplification task (Aprosio and Moretti, 2016). The Italian syntactic simplifier includes *Analysis*, *Transformation* and *Generation* modules, like the English one. It receives a sentence as input and returns either the same sentence or a simplified version. The rules we have implemented so far include splitting of conjoint clauses, relative clauses and appositive phrases.
- **Galician:** we needed to create basic NLP tools (e.g. a parser) before implementing syntactic simplification rules for Galician. However, due to lack of large enough resources to create these tools, only rules for simplifying appositive phrases and passive voice were feasible for this language. Therefore, we also experiment with a machine translation approach in order to use all the English rules for Galician.
- **Spanish:** we made use of existing resources and tools for NLP in order to implement a toy set of syntactic rules. Similarly to the English case, resources and tools were already previously validated. Therefore, the first version of the Spanish simplifier can adapt examples containing conjoint and relative clauses, appositive phrases and passive voice.

2.2.3 Architecture

Our approach for syntactic simplification is rule-based and follows the framework proposed by Siddharthan's work. Therefore, our systems are composed by three main modules: analysis, transformation and generation.

In Section 2.2.3.1 we present the architecture of the English simplifier and its evaluation results. Section 2.2.3.2. presents the Italian simplifier and the set of syntactic rules that have been implemented so far. Section 2.2.3.3 contains the description of the Galician simplifier, including a discussion of the limitations faced for this language, the evaluation of the results, and an alternative approaches that uses machine translation from Galician to English in order to use the English simplifier. Section 2.2.3.4 presents the description of the first version of the Spanish simplification tool with some toy examples.

2.2.3.1 The English Syntactic Simplifier

For English, several resources and tools for NLP were already freely available and with considerable work done in evaluating and improve such resources. We used the Stanford Dependency Parser (Chen and Manning, 2014), available at CoreNLP toolkit,[5] the NodeBox English Linguistic library for python (De Bleser et al., 2002)[6] and a Language Independent Truecaser for Python.[7] For this language we implemented rules for the simplification of conjoint clauses, relative clauses, appositive phrases and passive voice.

Analysis

The English *Analysis* module is responsible for pre-analysing the sentences searching for clues for the simplification of conjoint clauses and relative clauses. Such clues are discourse markers for conjoint clauses and relative pronouns for relative clause. An exhaustive list of the discourse markers and relative pronouns we consider are shown Table 5.

Discourse markers of time	Discourse markers of concession	Discourse markers of justification	Discourse markers of condition	Discourse markers of addition	Relative pronouns
when	although	because	if	And	whom
after	though	so	or		whose
since	but	while			which
before	however				who
once	whereas				

Table 5 – Discourse markers and relative pronouns considered by the analysis module

At this stage, sentences are marked whether or not they present a clue that may indicate a conjoint or relative clause. If this module returns True, the sentence is them sent to the part of the transformation module responsible for the simplification of such clauses.

It is worth mentioning that the classification of discourse markers into different classes is needed to perform the correct simplification for each marker. For example, a conjoint clause with "although" as discourse marker should be processed differently than a conjoint clause with "when" as a discourse marker. This stage of re-building the sentences with the correct simplifications is done at the *Generation* module.

[5] http://stanfordnlp.github.io/CoreNLP/
[6] https://www.nodebox.net/code/index.php/Linguistics
[7] https://github.com/nreimers/truecaser

Transformation

In the *Transformation* module, the simplification rules are applied. The rules developed up to this deliverable (M12) account to the simplification of conjoint clauses, relative clauses, appositive phrases and passive voice. Examples of each type of simplification are found below:

- Conjoint clauses:
  - **Original:** If you're in a nursing home you may be eligible for funded nursing care and continence aids. (SCC website)
  - **Simplified:** Suppose you're in a nursing home. Then you may be eligible for funded nursing care and continence aids.
- Relative Clauses:
  - **Original:** 'The pace of life was slower in those days', says Cathy Tinsall, who had five children. (Siddharthan, 2004)
  - **Simplified:** 'The pace of life was slower in those days', says Cathy Tinsall. Cathy Tinsall had five children.
- Appositive phrases:
  - **Original:** "There's no question that some of those workers and managers contracted asbestos-related diseases," said Darrell Phillips, vice president of human resources for Hollingsworth & Vose. (Siddharthan, 2004).
  - **Simplified:** "There's no question that some of those workers and managers contracted diseases,'' said Darrell Phillips. Darrell Phillips was vice president of human resources for Hollingsworth & Vose.
- Passive Voice:
  - **Original:** These organisations have been checked by us and should provide you with a quality service. (SCC website)
  - **Simplified:** We have checked these organisations. And these organisations should provide you with a quality service.
- Multiple simplification:
  - **Original:** The meeting, which is expected to draw 20,000 to Bangkok, was going to be held at the Central Plaza Hotel, but the government balked at the hotel's conditions for undertaking necessary expansion. (Siddharthan, 2004)
  - **Simplified:** The meeting is expected to draw 20,000 to Bangkok. The meeting was going to be held at the central plaza hotel. But the government balked at the hotel's conditions for undertaking necessary expansion.

This module is the core of our syntactic simplification engine and is responsible for calling the other two modules. The main method is called *simplify* which receives a sentence as input and return a simplified sentence. The simplification is implemented as a recursive process, that will keep

simplifying the sentence until there is no more simplification available. The order of simplification is: appositive phrases, conjoint clauses, relative clauses and passive voice.[8]

For conjoint and relative clauses, the *Analysis* module returns a flag indicating a potential marker that can initiate a conjoint or relative clause. For appositive phrases and passive voice, the detection is done at the *Transformation* module (this may change in the future, when more analysis will be included in the *Analysis* module).

The *Transformation* module is also responsible for sending the *Generation* module all the information needed for generating the simplified sentences. Such information include: discourse marker, relative pronoun, part-of-speech of main and modal verbs and part-of-speech of subject.

All the simplifications are done based on the output of the dependency parser. Table 6 shows the parser output for the sentence "These organisations have been checked by us and should provide you with a quality service." as an example. In order to perform the simplification, the *Analysis* model first searches for discourse markers. In this case, "and" is found and the sentence is marked to be send to the conjoint clauses rule. Once this rule is activated, it will search for two tags in the root dependencies ADVL (adverbial clause modifier) or CC (coordinating conjugation). In the particular case shown in Table 6, there is a CC relation between "checked" (the root) and "and". Since "and" is on the list of markers chosen for simplification in the *Analysis* module, the next step is to search for a CONJ (conjunction) tag. In our example, there is a CONJ relation between "checked" and "provide". The conjoint clause rule is then applied and the sentence is split into two. Each sentence is then sent to the *Generation* module. The simplified sentence after the application of the conjoint clause rule would be "These organisations have been checked by us. And these organisations should provide you with a quality service." Then, the simplified sentences are sent again to the simplifier in a recursive step. The simplifier will search for any other operation that can be performed. In this example, the first simplified sentence "These organisations have been checked by us." is at the passive voice, therefore, the simplification rule for transforming from passive into active voice is applied.

det(organisations-2, These-1) nsubjpass(checked-5, organisations-2) aux(checked-5, have-3) auxpass(checked-5, been-4) root(ROOT-0, checked-5) case(us-7, by-6) nmod(checked-5, us-7) cc(checked-5, and-8) aux(provide-10, should-9) conj(checked-5, provide-10) dobj(provide-10, you-11) case(service-15, with-12) det(service-15, a-13) compound(service-15, quality-14) nmod(provide-10, service-15)

Table 6 – Dependency parser output

[8] This order was selected empirically, after a set of experiments was performed.

Generation

The *Generation* module generates the simplified sentence(s). For the cases of conjoint and relative clauses and appositive phrases, the simplification consists in splitting the sentence into two. For passive voice changes, the simplification is done inside the sentence. Therefore, this module needs to account for the different types of simplification and perform the correct generation approach. We created three methods for generation: one shared by conjoint and relative clauses, one for appositive phrases and one for passive voice.

Apart from specific methods for each simplification type, general methods are also implemented in this class. Such methods deal with truecasing and removal of extra punctuation. For truecasing, we used the model and tool available at https://github.com/nreimers/truecaser. The method for punctuation removal is composed by several rules that identify punctuation repetition.

In the case of conjoint clauses, the simplified sentences need to follow the discourse marker being simplified. For example, if the complex discourse marker is "although" the second simplified sentence will start with "but". On the other hand, if the complex discourse marker is "if" the first simplified sentence will start with "suppose" and the second simplified sentence will start with "then". For some cases, the order of the simplified sentences will not follow the order in which the same information appears in the original sentence. For example:

- **Original:** When your Direct Payment ends we'll contact you to reclaim any money you've not spent. (SCC website)
- **Simplified:** We'll contact you to reclaim any money you've not spent. This'll happen when your direct payment ends.

Table 7 summarises the operations performed by the *Generation* module for conjoint clauses.

Discourse marker class	Operations
Time	S → S1. This <<modal verb>> <<to happen conjugated>> S2. (if marker is in the middle of S) S → S1. This <<to be conjugated>> S2. (if marker is the first word of S)
addition	S → S1. And S2.
condition (if)	S → Suppose S1. Then S2.
condition (or)	S → S1. Alternatively S2.
Justify	S → S1. So S2.
concession	S → S1. But S2.

Table 7 – Simplification operation per class of discourse markers

The rules for relative clauses are all applied at the *Transformation* module. Some operations such as the simplification of "My uncle, whose child you just met, is a pediatrician." into "My uncle is a pediatrician. My uncle whose child you just met." should probably be done at the *Generation* module. However, for simplicity, this was done at the *Transformation* module. At the *Generation* phase the simplified sentences are put together and the general methods are applied.

For appositive phrases simplification, the verb that connects the subject to the apposition is defined according to the number of the subject and the tense of the main verb. For example:

- **Original:** Truffles, a luxury food, are delicious.
- **Simplified:** Truffles are delicious. Truffles are a luxury food.
- **Original:** The insect, a large cockroach with hairy legs, crawls across the kitchen table. (http://www.chompchomp.com/terms/appositive.htm)
- **Simplified:** The insect crawls across the kitchen table. The insect is a large cockroach with hairy legs.

Finally, the changes in passive voice also required verb changes. Such changes need to respect the tense of the modal verb and the number of the subject. Moreover, changes in the pronoun realisation should also be considered. When pronouns are the subject of the passive voice, they appear in the objective form ("me", "you", "him/her/it", "us", "them"). However, in order to change the voice of the sentence, the pronouns also need to change from the objective to the subjective form ("I", "you", "he/she/it", "we", "they"). Examples:

- **Original:** A letter was written to me by Rita.
- **Simplified:** Rita wrote a letter to me.

- **Original:** The window is being broken by the men today.
- **Simplified:** The men are breaking the window today.

- **Original:** The window must be broken by us.
- **Simplified:** We must break the window.

- **Original:** The windows have been broken by the hammer.
- **Simplified:** The hammer has broken the window.

In order to change the conjugation of the verbs from the passive to the active voice, we used the NodeBox toolkit that provides methods for conjugating verbs.

Evaluation

We conduct an evaluation of our simplifier for English using 1100 sentences extracted from Sheffield City Council webpage, that were previously simplified by humans. This manual simplification was done by two fluent non-native speakers of English (including ourselves). From the total sentences, 242 were manually simplified, while 292 were simplified by our system. From the 292 sentences, 280 were in some extent different from the manual evaluation.

We then performed a manual evaluation of the 280 sentences, annotating them in five class:

- Equal to the reference (36 sentences) → differences were due extra spaces in the simplification version or error in the analysis;
- Alternative simplification (15 sentences) → the simplification is valid and correct, but diverge from the manual simplification;

- Over-simplification (42 sentences) → the rules were applied correctly and the simplifications are valid, although the manual version is not simplified;
- Non-simplified (117 sentences) → sentences that were manually simplified but not automatically simplified;
- Wrong simplification (70 sentences) → automatic simplifications are not valid or not correct.

Considering that 117 sentences from the set of sentences diverging between automatic and manual simplifications were not simplified by the automatic systems, such sentences are removed from the evaluation of the applied rules. Although we probably need to study them deeply in order to understand why the simplifier is not able to perform the simplification, they should not be taken into account in the evaluation of the simplified sentences. Also, if we consider that sentences in classes "equal to reference", "alternative simplification" and "over-simplification" are not problematic, since the rules were applied correctly and the simplified sentence is also valid, then only 70 sentence are the bottleneck of our systems. This corresponds to 23% of the 292 sentences simplified by our system.

2.2.3.2 The Italian Syntactic Simplifier

The Italian syntactic simplifier is a modified version of the ERNESTA tool (Barlacchi and Tonelli, 2013): it simplifies complex sentences made of different clauses by splitting them into simple sentences, but does not delete any optional part of the syntactic tree (i.e. adjuncts), differently from the original tool. Also verb tenses remain the same.

Analysis

The analysis is performed using the NLP components available in the TINT system (Palmero Aprosio and Moretti, 2016)[9], which is the first open-source suite for Italian NLP and was developed and released within the framework of the SIMPATICO project. The tool was necessary to provide the information layers related to sentence structure, verb tense, etc., that were needed to automatically decide how a sentence should be simplified. Similar to English, the main features considered for syntactic simplification are the presence of relative clauses and of coordinating discourse connectives.

Transformation

In this stage, coordinated clauses and relative clauses are split into two sentences.

Generation

In this step, punctuation is added and the subject of the first sentence is repeated also in the second sentence. For example:

- **Original:** Il tecnico verifica la correttezza delle pratiche e comunica i risultati all'ufficio competente.
- **Simplified:** Il tecnico verifica la correttezza delle pratiche. Il tecnico comunica i risultati all'ufficio competente.

- **Original:** Il tecnico comunica i risultati del controllo all'ufficio competente che li trasmette al catasto.

[9] Demo, source code and detailed information available at http://tint.fbk.eu/

- **Simplified:** Il tecnico comunica i risultati del controllo all'ufficio competente. L'ufficio competente li trasmette al catasto.

Since no tool for anaphora resolution is currently available in Italian, we cannot perform more complex substitutions (for example replacing a pronoun with the antecedent). We plan to implement such tool in the second year of the project.

2.2.3.3 The Galician Syntactic Simplifier

For Galician, the amount of resources and tools developed is much smaller than for English. Although there are some recent initiatives such as Linguakit[10], there is yet a lot to be done. In order to build the syntactic simplifier, we need to train a dependency parser,[11] a part-of-speech tagger and a truecasing model. The dependency parser and part-of-speech tagger were trained using the CoreNLP framework and the Universal Dependencies dataset.[12] The truecasing model was trained using the same tool the performs the truecasing for English.

In order to correctly conjugate verbs in the transformation from the passive voice into active voice, we implemented a module that reads the verb lexicon available at Linguakit and performs a task similar to what NodeBox does for English.

Due to limitations in the Universal Dependencies for Galician, we could not implement any rule for solving conjoint clauses or relative clauses. The problem was that some dependency tags used for the identification of such rules were not annotated into the dataset. Instead, sentences with such clauses were annotated with more general tags. Therefore, we could not generalise and generate specific rules for these clauses and no *Analysis* module was implemented.[13]

Transformation

The *Transformation* module only implemented rules for simplifying passive voice and appositive phrases. In both cases, however, the ideal parser tags were also not available and we needed to extract the rules using more general tags. This is more susceptive to errors.

We formulated the rules for Galician starting from the English rules and adapting them for Galician language. Examples of simplification are shown below:

- Appositive phrases:
  - **Original:** O delegado territorial da Xunta en Lugo, José Manuel Balseiro, explicou esta mañá que a Consellería de Política Social inxecta anualmente en Viveiro máis de 2,6 millóns de euros para o sostemento das 238 prazas existentes nesta localidade nos distintos centros que prestan servizo e atenden a persoas dependentes ou con necesidades especiais (Xunta de Galicia wepage)
  - **Simplified:** "José Manuel Balseiro explicou esta mana que a Consellería de Política Social inxecta anualmente en Viveiro mais de 2,6 millóns de euros para o sostemento das 238 prazas existentes nesta localidade nos distintos centros que prestan servizo

[10] https://github.com/citiususc/Linguakit

[11] A dependency parser was already available for Galician into the Freeling toolkit (http://nlp.cs.upc.edu/freeling/). However, our framework interacts better with the Stanford CoreNLP platform and, therefore, we decided to train a new parser for our purposes.

[12] http://universaldependencies.org

[13] The tags needed to the simplification of conjoint and relative clauses are ADVCL, NMOD and ACL:RELC.

e atenden a persoas dependentes ou con necesidades especiais . José Manuel Balseiro é o delegado territorial da Xunta en Lugo.

- Passive Voice:
  - **Original:** O zoco foi metido por Ana.
  - **Simplified:** Ana meteu o zoco.

Generation

This module is responsible for calling the verb conjugation module and correctly inflect the verb for passive voice simplification, applying the truecasing, correcting punctuation marks and include the correct verb in the second sentence of the appositive phrases simplification.

Evaluation

We applied the simplification rules at 790 sentences extracted from the selected use-cases for the Galician pilot. From these sentences only 22 were simplified by our systems. After an evaluation with native speakers of Galician, only 2 sentences were identified as showing correct grammar and meaning preservation (less than 1%).

After this rather negative result, we also experimented with a machine translation approach where we first translated the sentences from Galician into English, then we applied the English syntactic simplifier and, finally, translated the simplified sentences back into Galician. In this scenario, 58 sentences were simplified by the systems and 14 were considered correct (slightly over 2%). Although in this case the recall is higher because the English simplifier considers more types of simplification, the precision is still very low.

Given the poor performance of the simplifier, the consortium made a strategic decision of switching from Galician to Spanish. Although Galician is the prefered language in Galicia, the majority of the citizens of the country are bilingual. Spanish language has much more NLP resources available and with higher quality. Therefore, the simplifications for this language are expected to be more reliable.

2.2.3.4 The Spanish Syntactic Simplifier

We used the Stanford Dependency Parser, with the Spanish models version 3.7.0 (with support to dependency parser). The same tool used for English was employed for truecasing. We still need to study resources for solving verb conjugation. At the moment, we plan to use the lexicon available at Linguakit for Spanish (similar to the lexicon used for Galician).

For Spanish, since all the necessary tags are available in the parser, all rules have potential to be implemented. For this deliverable, we implement a toy version of the possible rules also having the English rules as examples. However, specific rules for Spanish language may be studied and developed in order to generate a natural simplifier for this language.

Analysis

This modules follow the English implementation and pre-select sentences with selected discourse markers and relative pronouns to be simplified. Table 8 shows the discourse markers and relative pronoun that we addressed in the first experiments with the Spanish syntactic simplifier.

Discourse markers of time	Discourse markers of concession	Discourse markers of justification	Discourse markers of condition	Discourse markers of addition	Relative pronouns
(not implemented yet)	pero	(not implemented yet)	si	y	que

Table 8 – Discourse markers and relative pronouns considered by the Analysis module for Spanish

Transformation

The *Transformation* module for Spanish also implements the rules in a similar way to the same module for English. The tags used for the identification of the tags are the same or similar variations. Examples of simplified sentences by our toy simplifier are:

- Conjoint clauses:
  - **Original:** Si la solicitud no reúne alguno de los requisitos exigidos en la presente orden, requerirán a las personas interesadas para que, en un plazo de diez días hábiles, subsanen la falta o adjunten los documentos preceptivos. (Xunta de Galicia webpage)
  - **Simplified:** Supongamos la solicitud no reúne alguno de los requisitos exigidos en la presente orden. Requerirán a las personas interesadas para que , en un plazo de diez días hábiles , subsanen la falta o adjunten los documentos preceptivos.
- Relative Clauses:
  - **Original:** Los datos de ejecución y los indicadores de resultado inmediato, que se deberán de cumplimentar según el modelo del anexo V de esta orden, se facilitarán en el período de justificación de la correspondiente subvención. (Xunta de Galicia webpage)
  - **Simplified:** Los datos de ejecución y los indicadores de resultado inmediato se facilitarán en el período de justificación de la correspondiente subvención. Los datos de ejecución y los indicadores de resultado inmediato se deberán de cumplimentar según el modelo del Anexo V de esta orden.
- Appositive phrases:
  - **Original:** Juan Carlos I, rey de España, presidirá el acto de homenaje a Cervantes. (http://www.gramaticas.net/2012/07/ejemplos-de-aposicion.html)
  - **Simplified:** Juan Carlos I presidirá el acto de homenaje a Cervantes. Juan Carlos I és rey de España.
- Passive Voice:
  - **Original:** El coche fue reparado por el mecánico. (http://roble.pntic.mec.es/acid0002/index_archivos/Gramatica/voz_pasiva.htm)
  - **Simplified:** El mecánico reparó el coche .

Generation

This module is intended to perform very similar to the English *Generation* module. As mentioned before, we are still investigating the best resources and tools for this module.

2.2.4 Interfaces

In this Section we describe the part of the Text Adaptation Engine API pertaining to the Syntactic Adaptation module. Table 9 summarises the syntactic simplification interface.

Method	/syntactic/{sentence}/{lang}
Type	POST
Description	Returns a sentence that can be equal to the input sentence received as a parameter or can be a simplified version, after applying the syntactic simplification rules. The language dependent simplifier is called according to the language also sent as a parameter. The call and the response are processed by the Interactive Front-End.

Table 9 – Syntactic Adaptation API

In the Text Adaptation Engine API, one can request a syntactic simplification through the /syntactic POST function. It receives as input a simplification problem composed by a sentence. In case the simplification problem is successfully addressed, it returns a response containing the simplified sentence. In case of an unsuccess, it either returns the original sentence or an error message describing the problem.

2.2.5 Links

For more details about the input and output parameters pertaining to this API function, one can resort to its Swagger specification at:

https://app.swaggerhub.com/api/carolscarton/SIMPATICO_TAE_Syntactic_API/0.0.1

One can find the code for the Syntactic Adaptation API at the following github project page:

https://github.com/SIMPATICOProject/SimpaticoTAEServer

2.2.6 Next Steps

For the upcoming months we plan to improve the syntactic simplification for **English** by first addressing the 70 sentences classified as "wrong simplification". We will check whether or not it is possible to refine the rules in order to avoid such problems. Moreover, we intend to evaluate the output of our system with a grammar checker. In this scenario, if a sentence is judged to have grammar problems, we can consider it as a wrong simplification and should not show it to the user. We are evaluating the tool developed by Napoles et al. (2016)[14] in order to identify grammar errors in our simplified sentences. In addition, we plan to include a pre-processing step that will select sentences to be simplified based on its complexity. In this case, we want to avoid over-simplification that can complicate the information instead of simplifying it.

[14] https://github.com/cnap/grammaticality-metrics

Due the lack of resources and lack of NLP personnel in the project to work on the development and evaluation of new rules, we will not address **Galician** in this module anymore. Instead, we will focus on building a high quality syntactic simplifier for Spanish.

For **Spanish**, the syntactic simplification solution still need to be revised and extended. Up to now, we only developed a toy set of rules, where some rules derived from the English are applied. New rules for Spanish will be studied and implemented in order to deal with simplification types that are exclusive of this language. Ways to check the grammar of the simplified sentences and a pre-processing phase for sentence selection should also be included.

For **Italian**, we will continue the integration of simplification rules following Municipality of Trento's suggestions, and we will use the SIMPITIKI corpus to evaluate different simplification strategies. We will also implement and release a tool for basic anaphora resolution in Italian, which would greatly improve the identification and explicitation of pronoun referents, a crucial step towards simplification.

Personalisation of the simplification process is an aim of the SIMPATICO project. In this case, the simplification process will be dependent on the user's profile and preferences. Rule-based systems are not completely flexible for this type of personalisation, therefore, we intend to explore statistical and neural-based methods for simplification at least for English, due to resources constraints.

3 Text Analytics for Text Profiling

In order to extract information about a given document, in particular about its readability, a set of analytics is provided, combining state-of-the art tools for lexical and syntactic processing, domain-specific information and hints about grammaticality and style. This information will mainly serve to inform the civil servants and others responsible for creating the content of the PA's websites, on the level of complexity of a given text. This information will support them in the process of creating and or/editing content such as to make it more readable and understandable by the end-users.

This section includes a brief description of the state of the art for text profiling (Section 3.1.), a summary of the key functionalities of SIMPATICO text analytics (Section 3.2.), information on the interfaces (3.3.) and the links to the code (3.4.). Finally, we present our plans for 2nd-year extensions (Section 3.5.).

3.1 State of the art

Although no available tools provide as many information layers as those extracted with the SIMPATICO analytics module in a multilingual setting, past work has discussed language-specific readability and suggested different metrics. For English, readability was traditionally computed combining factors such as word and sentence length (Flesch, 1946; McLaughlin, 1969; Fleisch-Kincaid, 1975), but more recent studies have taken into account also other aspects related to syntactic complexity, phrase density, etc. (Crossley et al., 2011). Along this line, some attempts have been made to provide a suite of tools for extracting a number of language-specific readability indices from text (see for example McNamara et al. 1996, among others), usually focusing on English. For Italian, little work has dealt with text analytics for readability assessment (Dell'Orletta et al.,2011; Tonelli et al., 2012). Dell'Orletta et al. (2014) present a study to assess the readability of single sentences instead of documents, and analyse which indices work best at sentence level. Some of these have been integrated in our Text Analytics software. As for Spanish, work on readability and simplification has been carried out within the "Simplext" National project and then continued in the "Able to include"[15] European project. Some of the readability metrics considered in these projects have been included in our Analytics tool, with the exception of those developed for people with cognitive disabilities, since this is not the target of SIMPATICO. Finally, we are not aware of any specific work addressing readability and text complexity in Galician.

The advancement of our analytics w.r.t. the state of the art during this first project year does not lie in novel metrics, but rather in the availability of a number of indices and suggestions, comparable across languages. Indeed, the indices have been selected so that they are valid in all project languages.

3.2 Short summary of key functionalities

Given a plain text in input, the tool processes it with a system for linguistic analysis in different languages, all based on the Stanford CoreNLP suite (using models for English, Italian, Spanish), and outputs the information presented in Table 10.

[15] http://able-to-include.com/

Document statistics	
Language	
Number of sentences	
Number of tokens	
Number of words	
Number of content words (i.e. verbs, nouns, adjectives and adverbs)	
Difficulty level	
Standard readability index for the language	Flesch (EN) / Gulpease (IT) / Flesch-Szigriszt (ES)
Text difficulty for users with a very basic vocabulary (500 most frequent in the language)	
Text difficulty for users with a basic vocabulary (2500 most frequent words)	
Text difficulty for users with a limited vocabulary (5000 most frequent words)	
Part of Speech distribution	
Proportion of nouns, adjectives, nouns, etc.	
Information on parse tree	
Depth of the parse tree, n. of coordinated and subordinated clauses	

Table 10 – Output of the Text Analytics Tool

Part of speech distribution can be an important indicator of text complexity. For example, the average number of noun phrases in the PA domain is higher than in standard language, and it is known to negatively affect texts readability. If noun phrase density is too high, it should be reduced, for example by increasing verbal phrases.

The analytics include also suggestions from LanguageTool[16], an open source software (LGPL) that checks the grammaticality and the style of sentences and suggests possible changes. It is available for Italian, English, Spanish, Galician, and many other languages. For the moment, it is integrated in the standard version, but in the future we plan to customise it to the domain, keeping only the suggestions that are appropriate for the PA domain.

All modules needed to process Italian texts that were not available in the Stanford CoreNLP framework have been developed during the project and made available to the research community within the TINT suite as stand-alone tool, Java library or REST API service[17].

[16] https://www.languagetool.org/
[17] http://tint.fbk.eu/

3.3 Interfaces

The Text Analytics Engine provides RESTful API for getting information about a piece of text:

Method	/simp
Type	GET/POST
Description	Receives as input a text and the language in which it is written. As output, it returns an enriched version of the text, with definitions, statistics, links to Wikipedia and proposed simplifications.

Table 11 – Text Analytics Engine API

3.4 Links

The detailed documentation of the Text Analytics Model Repository API is available here (Swagger 2.0 documentation):

https://dev.smartcommunitylab.it/simp-engines/swagger-ui.html#!/tae-controller/

The Text Analytics software is available as standalone Java project based on the Spring Boot project. Its source code and an example is available here:

https://github.com/SIMPATICOProject/simpatico-adaptation-engines

3.5 Next Steps

In the next steps, we plan to extend the available analytics with additional metrics, for example those pertaining to discourse level (e.g. number of subordinate clauses, position of main clause, etc.). We will also extend the analytics so that information about users can be taken into account. For example, selecting a user profile with “high language proficiency”, the metrics related to basic lexical knowledge will be greyed out.

4 Workflow Adaptation

In many cases the structure of the e-service forms is very complex mainly because forms are composed by many atomic information units (e.g. user registry, services specific information and data, declaration, ...). Usually, the design of the e-service is a replica of the paper forms. Since the pre-existing paper modules have been implemented considering the possibility to have a civil servant helping and following the citizens or professionals in the form filling the digital e-services present two major issues: (1) the users suffer for lack of domain knowledge; (2) the tools to support filling out forms (e.g. how-to blog, information field prefilled, help pages) do not fulfil the user needs.

This situation puts the citizen in the unfortunate position of having to manage the complex interaction without the help of the Civil Servant. This is why it is necessary to introduce interface adaptation techniques for aligning the interaction complexity and the specific user competences and capabilities. In the SIMPATICO project, the Workflow Adaptation Engine (WAE) is the component responsible to fulfill this expectation. In the following subsections we introduce the state of the art in interface and process adaptation techniques, and present the functionality of the WAE component.

4.1 State of the art

Interface adaptation is a well know issue and the literature has many references to the adaptation frameworks exploring different techniques. More specifically, adaptive web-based systems tackle this problem by displaying only the information that is meaningful to the user. In De Bra et al. (2005) and Lohmann et al. (2006), the authors address the problem exploiting semantic models for the interface adaptation combined with a corresponding adaptation execution engine. Henricksen and Indulska (2001) address the problem introducing a specific web server that proposes context-specific web pages. The context knowledge of the page is acquired from the metadata sent by a specific adaptive browser. Brusilovsky and Schwarz (1997) propose to use the “User as a student” paradigm and the “incremental interface” concept. The idea is based on a set of interface stereotypes: the more the user experience grows, the more the complex parts of the interface will be accessible. Finally, Montes García et al. (2014) propose an extended CSS in order to implement a Domain-Specific Language (DSL) to express adaptation rules and to decouple them from the business logic. However, the assumption underlying all those papers is that the interfaces are implemented with the adaptation requirements in mind, which is not the case of the SIMPATICO approach.

Business process adaptation is widely addressed in the literature. For the sake of simplicity, we assume that a business process defines a set of actions that must be executed in order to fulfil a goal. This goal can be represented using imperative or declarative languages. Imperative process modeling languages (e.g., OASIS (2007)) describe the way to achieve the goal, and assume that the environment is stable. Declarative process model languages (e.g., van der Aalst et al. (2009)) specify the goal throughout the constraints on the task's execution in order to approximate the desired behavior.

The problem of supporting the evolution of business processes models has been addressed by several previous work. Some of them are able to define various context settings and to associate them with specific process variants (Hallerbach et al., 2010; Rosa et al., 2009). Some other works address the process evolution by analysing previous executions and adaptations (Rinderle et al., 2005, Li et al., 2009, Plosser et al., 2009, Guenther et al., 2007, Lu and Sadiq, 2006).

Many approaches focus on the problem of extracting useful information from the execution logs. These approaches differ both in what they log and in the techniques that they use to analyse those

logs (e.g., Rinderle et al. (2005), Li et al. (2009), Guenther et al. (2007), Lu and Sadiq (2006), Schonenberg et al. (2008)).

One direction here is to use process mining techniques, as in Li et al. (2009), considering large collections of structurally different process variants created from the same process model. The authors use a heuristic search to find a new process model such that the weighted average distance between the new model and the variants is minimal.

Similarly, mining techniques are used to analyse the changes in logs in Guenther et al. (2007). The outcome of this approach is an abstract change process consisting of change operations and causal relations between them. These change processes can be used as an analysis tool to understand when and why changes were necessary.

A different technique is adopted in Rinderle et al. (2005) where concepts and methods from case-based reasoning (CBR) are used in order to also log, together with the change operations, the reasons and context of each change. Change information is stored as cases in a case-base specific to the process model. The case-bases are used to support process actors in reusing information about similar ad-hoc changes, and are also continuously monitored to automatically derive suggestions for process model changes.

Change analysis and reuse is also relevant for loosely specified process models. Lu and Sadiq (2006) facilitate reuse of changes by providing a search interface for the repository of process variants. For declarative processes, Schonenberg et al. (2008) supports users through recommendations, which are generated based on similar past executions and considering certain optimization goals.

4.2 Short summary of key functionalities

A way to deal with the e-service form adaptation problem is to break a complex form into small pieces, giving the user the possibility to focus himself in the compilation of specific information (Figure 4). SIMPATICO implements this feature through the Workflow Adaptation Engine (WAE) that changes the form interaction based on the user profiles. The Workflow Adaptation Engine will be used in a context where the electronic forms are already present. This means the proposed solution should impact on the actual system as less as possible. To fulfill these constraints we came up with a solution combining interface adaptation and process adaptation techniques.

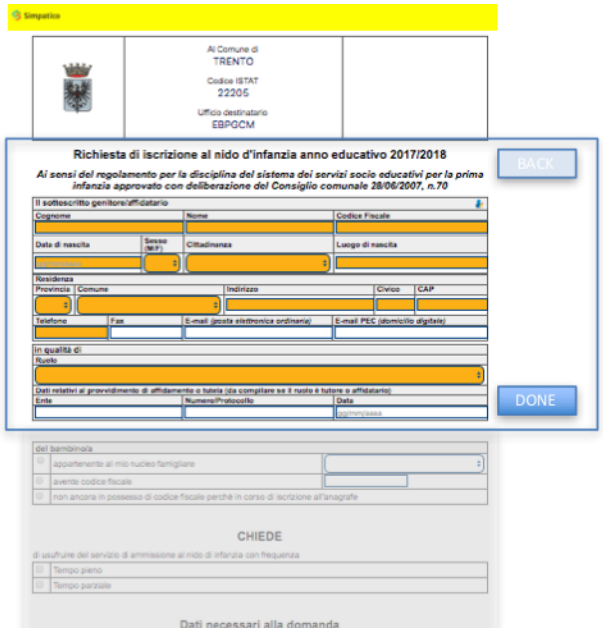

Figure 4 – Workflow Adaptation Example

The proposed solution relies on the Interaction Model, the WAE and the User Profile. The Interaction Model defines the information blocks and the interaction dependencies between those blocks (Figure 5). WAE is responsible for the e-service interface adaptation based on the Interaction Model and on the user specific interaction capabilities present in the User Profile. At the implementation level, this is a client-side library (i.e., a JavaScript library in case of e-service HTML forms) which should be injected in the adapted digital module (Figure 6).

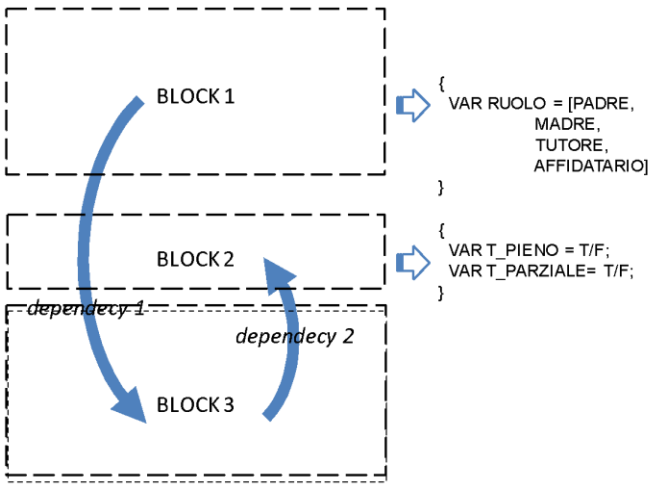

Figure 5 – Interaction model

The definition of the model takes place at design time, while the adaptation of the form takes place at runtime. More specifically, the adaptation manager defines the relevant information blocks of the form and the order in which the blocks should be processed by the user. This defines the form

interaction workflow. This workflow is defined with the help of the block dependencies and information constraints. Please note that this activity may be done even after the corresponding e-service has been implemented and deployed. To associate different form blocks to the existing implementation (e.g., to the existing HTML page), explicit association links are provided (e.g., in the form of XPath).

The e-service form models are uploaded to the Workflow Adaptation Engine Repository. It is possible to have more than one form model for the same e-service form, associating them to different user profiles. The repository provides the necessary operations for storing, updating, deleting, and querying the models.

At run time, the client-side engine implementation interacts and retrieves a profile associated to the e-service form and to the profile of the current user. The model is then used to modify the form and to guide the user through the data insertion workflow.

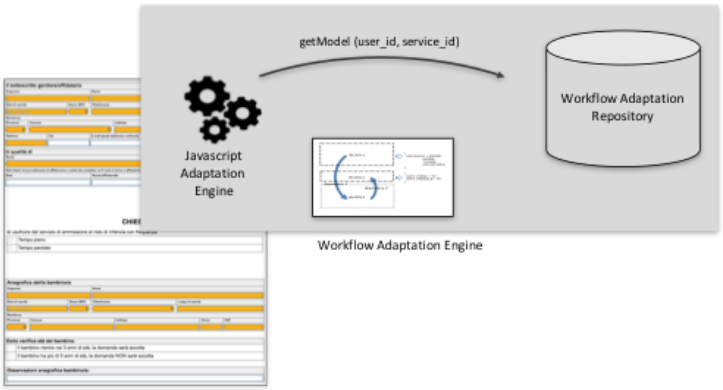

Figure 6 – Workflow Adaptation Engine logic overview

4.3 Architecture

The workflow model of the e-service forms is organised in a hierarchy of **blocks**, where each block represents a logically grouped elements. More specifically, the model distinguishes between the simple blocks that group elements like fields, text, etc. and the containers that group together other blocks in compilation units. Each block is therefore defined with:

- block identity elements (block id, type, information tags, to be used for relation with other SIMPATICO components, e.g., with Citizenpedia).
- list of underlying fields (in case of simple block)
- list of underlying blocks (in case of container blocks)
- XPath expression to associate the model element to the corresponding fragment of the HTML page
- dependencies, i.e., other blocks that should be compiled before the current one
- a precondition to evaluate whether or not this block is eligible as next step of the workflow (is defined on top of the workflow context properties)
- a postcondition to evaluate whether or not this block is complete (this is defined on top of the workflow context properties)

The block is available for compilation by the user if the blocks in the dependency list have been completed and the precondition is satisfied. The block is considered complete if its postcondition holds.

During the execution of the e-service form workflow, the user populates the form **fields**. Some of the fields are associated to the workflow **context** properties, i.e., variables that evolve during the execution. Each variable is represented with the key-value pair and may be tagged in order to link its value to the SIMPATICO ontology elements (e.g., to the user language or citizenship). The fields are represented with the following attributes:

- field identifier
- XPath expression to link to the underlying HTML page
- bound context property represented with the property key and binding direction (IN/OUT/INOUT), which characterized how the form data is propagated to/from the context.

By filling in the form data, the user populates the context properties and makes the workflow execution evolve.

4.4 Interfaces

The Workflow Engine uses a model repository that provides a RESTful API for creating, updating, deleting and querying the workflow models. More specifically, the following operations described in Table 12 are allowed.

Method	/wae/model?uri={uri}
Type	GET
Description	Return a list workflow models corresponding to the specific e-service form (identified with URI).

Method	/wae/model/page?uri={uri}&profileId={profileId}
Type	GET
Description	Return a workflow model corresponding to the specific e-service form (identified with URI) and a profile (identified with the profileId).

Method	/wae/model
Type	POST
Description	Create and save in repository a new model. The input model defines the workflow representation together and the associated user profiles.

Method	/wae/model/{modelId}
Type	PUT
Description	Update a specific workflow model

Method	/wae/model/{modelId}
Type	DELETE
Description	Delete a specific workflow model

Table 12 – Workflow Adaptation API

4.5 Links

The detailed documentation of the Workflow Model Repository API is available here (Swagger 2.0 documentation):

https://dev.smartcommunitylab.it/simp-engines/swagger-ui.html#!/wae-controller/

The source code for the Workflow Engine Model Repository and the Workflow Adaptation Engine is available as a standalone Java application based on the Spring Boot project. The source code and a demo are available here:

- Source code: https://github.com/SIMPATICOProject/simpatico-adaptation-engines
- Demo: https://dev.smartcommunitylab.it/simp-engines/wae/webdemo

4.6 Next Steps

As a first next step, it will be necessary to validate the proposed model against various use cases. Although the defined model is general, some of the use cases might require specific extensions. Second, since in this preliminary version the workflow adaptation techniques are rather simple, we will also target the extension of the Workflow Adaptation considering the user profile and user capabilities in the loop. Finally, we plan to address the usability issues and start identifying the requirements for the adaptation modelling support tools.

5 Citizen Data Vault

In the context of SIMPATICO project, the Citizen Data Vault is in charge of providing a secure repository of Personal Data of users (citizen, company) used during their interactions with the personalized services provided by the PAs by means of the SIMPATICO Platform.

5.1 State of the art

In the definition and implementation of the Citizen Data Vault several technologies have been taken into account, mainly belonging to the state of art of Personal Data Store (PDS) technologies and solutions. This state of the art of technologies on Personal Data Store was already described in the D5.1. This section provides a summary of the key aspects of this state of art.

This section provides some definitions and key solutions have been taken into account.

According to World Economic Forum (June 2010) definition, Personal Data is defined as data (and metadata) created by and about people. Personal data is also very broadly defined in Article 2 of the European Data Protection Directive as: "*... any information relating to an identified or identifiable natural person ("data subject")*...". This definition is, for the most part, unchanged under the new General Data Protection Regulation (GDPR).

According to the European Commission report (Report on Personal Data Stores[18]) a Personal Data Store (PDS) is a technology that enables individuals to gather, store, update, correct, analyse, and/or share personal data. Of particular importance is the ability to grant and withdraw consent to third parties for access to data about oneself. In other definitions the term "store" is changed with the terms "Locker" or "vault".

According to this definition it is possible to identify two main parties, that with the PDS itself compose the PDS ecosystem: a “Data Source”, that holds and provides data about each individual, and a “*Data Requester”, a* company and/or service that want to access data about individuals by means of the PDS.

As concern a general list of PDS features, starting from different definition and requirements of PDS, the range varies:

- *Data Storage*– an access point for personal information
- *Data Management*– a toolset for analyzing and understanding what data means
- *Data Sharing* – the ability to choose how to share personal information and with whom
- *Data Collection* – the ability to track and collect data
- *Verifications* – the ability to authenticate sensitive information generated by 3rd parties
- *Identity Assurance*– the ability to prove I am who I say I am
- *Privacy Management* – my info has a privacy setting determined by me, not organizations

[18] Study on Personal Data Stores conducted at the Cambridge University Judge Business School: https://ec.europa.eu/digital-single-market/en/news/study-personal-data-stores-conducted-cambridge-university-judge-business-school

- *Manage Permissions*– deciding the communication channels between me & my contacts
- *Express Interests & Intentions*– the ability to announce what I want to buy, do or access
- *Plan & Implement Projects* – a life management system for how I will use my info over time

In SIMPATICO context the Citizen Data Vault will not cover all the above features, but a selection according to the main scenarios and according to the requirements belonging from the three pilot use cases. Moreover these three use cases will specify deeper which type of data can be managed by the Citizen Data Vault according to the identified e-services.

Starting from the above mentioned characteristics of a Personal Data Store and related technologies, a list of potential solutions have been studied and analyzed. These solution have influenced the SIMPATICO Citizen Data Vault, or used as means of comparison:

Cloud based

- Mydex[19] – A Personal Data Store for individuals that helps them collect, store, manage, use and share their own personal data for their own purposes.
- Personal.com [20]– protect online personal data exhaust and sell it back to advertisers

Open Source Project

- Project Higgins[21] – open source identity framework designed to integrate identity, profile and social relationship information across multiple sites, applications and devices using an extensible set of components.
- Project Danube[22] – open source project to develop an XDI-based Personal Data Store – a semantic database for personal data which is controlled by the user. Applications on top of this database include the Federated Social Web and the selective sharing of personal data with organizations.
- OpenPDS[23] - a personal metadata management framework that allows individuals to collect, store, and give fine-grained access to their metadata to third parties
- Enigma[24] - a decentralized cloud platform with guaranteed privacy. Private data is stored, shared and analyzed without ever being fully revealed to any party. Secure multi-party computation, empowered by the blockchain technologies.
- MyData[25] -is an effort by the Open Knowledge Finland community to define a human-centric way to manage and process personal information. The core is that individuals should be in control of their own personal data.

Particular attention has been paid to MyData Solution, especially on consent management and service registration, and to Personal.com for the approach used on metadata management and web-form auto filling.

[19] http://mydex.org/
[20] http://www.personal.com/
[21] http://eclipse.org/higgins/
[22] http://projectdanube.org/
[23] http://openpds.media.mit.edu/
[24] http://enigma.media.mit.edu/
[25] https://github.com/HIIT/mydata-stack

5.2 Short summary of key functionality

The Citizen Data Vault (CDV) makes the citizens able to control their personal data profile in an easy way.

Through the Citizen Data Vault, a citizen will be able to manage and share their own personal data and profile. It enables them to control how the data should be accessed and by whom, within the legal framework in which they are involved. This way, the citizen will represent the provider and the owner of their personal information.

Main typical use cases of CDV (see Figure 7) are the following:

- Manage Personal Data and Metadata (Ontology): users can have the **whole control** of their personal data stored in the CDV
- Register Services: the e-service has to be registered in order to provide a **description**.
- Manage Data consent: users have to provide a **consent** in order that the e-service can access to and use the personal data of the user.
- Data service and Personal Data Ontology Mapping: the e-service owner has to perform the **mapping** between the data set of the service and the Personal Data taxonomy.
- Store Personal Data: the CDV allows users to **store** their personal data
- Retrieve Personal Data: the CDV allows users to **retrieve** their personal data.

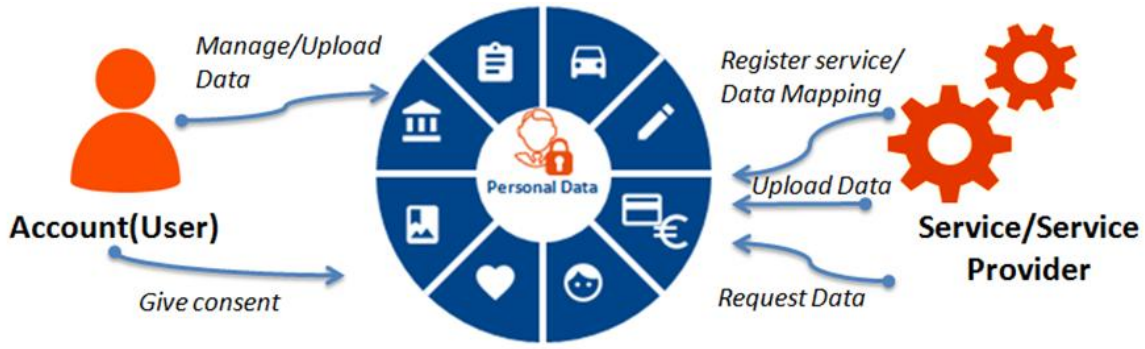

Figure 7 – CDV main features

The CDV has been implemented according to the previous definitions and objective and taking into account the following core concepts:

CDV Account: A registered data owner that can manage personal data by means of CDV. User has to be registered in CDV and an account needs to be associated to him/her.

Service Link and Consent. A Service Link certifies that there is a bond/link between an account(user) and a service. A consent is and authorization of a service to access a specific dataset under certain conditions. Both Service Link and Consent have a lifecycle status set by the user.

Data Source and Data Requester. According to the general definition of a Personal Data Store, a service can be a "Data Source", or an entity requesting data ("Requester"), or both. In the particular case, the CDV as well as providing the services of "Personal Data Service" behaves as "Datasource“ by

default. In order to enable multiple data sources, enabling a distributed architecture of the CDV, each source will managed by a data connector.

Each service to be able to interact with the CDV has to:

- Be registered on CDV(*Service Registration*);
- Be associated with an Account (*Service Linking*);
- Get a "consent" to access data (*Data Consent*).

Service Registration. This phase produces a description of the service and in particular the mapping of service dataset with categories and data fields from a taxonomy of Personal Data. A service is first registered providing a set of metadata describing the service itself. Once registered, a description of the service data is provided on which there is a mapping with the concepts of the Personal Data Taxonomy. For “data” we refer to the data that it consumes by interacting with CDV.

Service Linking. This step creates a link between Service and Account. In particular, the link service contains the service identifier (service ID) and the account (Account ID). The latter is represented by a surrogateID which is the user identifier within the service. The Service Linking process may include a registration / authentication process at the service (eg. Oauth). The Service Link will be saved in the CDV and in the service as it will be necessary for the interactions of the service with the CDV (data upload) and for the generation of a "consent".

Data Consent. This step provides a “consent” for the access of data that is available and/or mediated through the CDV. “Consent“ cannot exist without a Service Link. Once generated the consent this will be saved by both the CDV and the Service and used for data request. In particular the stored consent contains an "accountID", a "resourceSetID" and a "serviceLinkID". The "resultSetID identifies the set of data that the user will select during the generation of the consent. This ResultSet will be created and saved, returning its ID that will be inserted into the consent. During the consent verification the CDV verifies the validity of the consent, the service link, the resultset and finally, the state of the consent. In fact, any time the user has the option (like service linking) to change the status of consent (inactive, active, suspended ...).

Secure Personal Data. The CDV provides an infrastructure to securely protect personal data and avoiding a fraudulent access by 1) adopting the HTTPS protocol to transmit securely the user data; 2)by exposing a set of OAUTH2.0 protected APIs, to ensure the interaction only with the authorized data consumer and sources, 3) by adopting a multiple key based data encryption.

Address legal requirements. The CDV provides the functionality to have a copy of personal data in different formats (i.e. JSON, CSV..) and addressing the right to be forgotten enabling the functionality to remove the user Account and all its stored data.

5.3 Architecture

All the above features are provided by a set of components. The CDV architecture, multilayered and service based is depicted in Figure 8. A set of internal and external APIs are provided by the *Service Data Service* enabling the interaction with the data owner (user) via the CDV Dashboard or with external modules/services via a set of secure REST API.

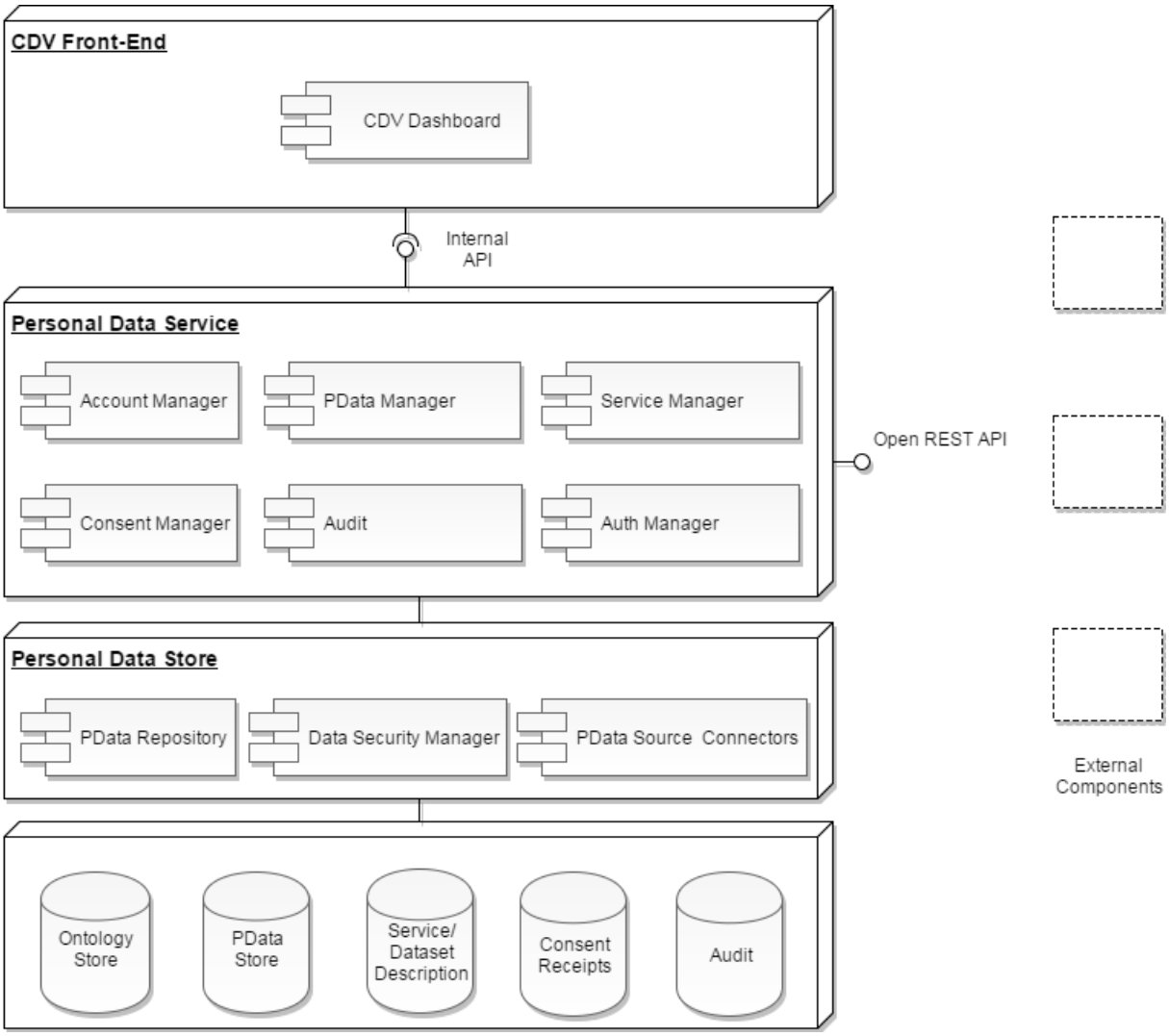

Figure 8 – Citizen Data Vault architecture

Service Manager

The Service Manager component is responsible for the registration of the services and the mapping of their data structure with Personal Data taxonomy. Each data field is mapped with a personal data field (pdata) belonging to a set of Personal Data Category. It is possible to modify the structure of Personal Taxonomy by adding or removing category or related data fields. The module provides a set of REST API for the registration of a services and the retrieve of its mapping.

Figure 9 – Service registration and data mapping by interacting with the Service Manager component

Account Manager

The Account manager is the main component of CDV architecture. It is responsible of the management of user account and of the service link and consent lifecycle by interaction with the Service Manager and Consent Manager (Figure 10 - Figure 11). The account manage provides a set of secure API for the interaction with the dashboard (user interface) and the interaction with external components.

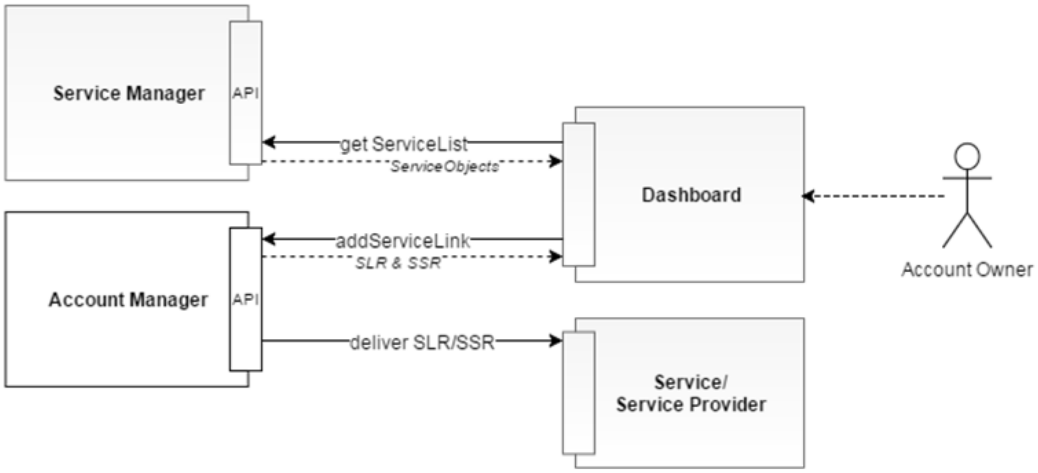

Figure 10 – Account manager interactions with other components.

Consent Manager. This component is responsible of the management of Data consent process and its lifecycle. This component does not interact directly with the dashboard or the external components but it provides its functionalities to the Account Manager.

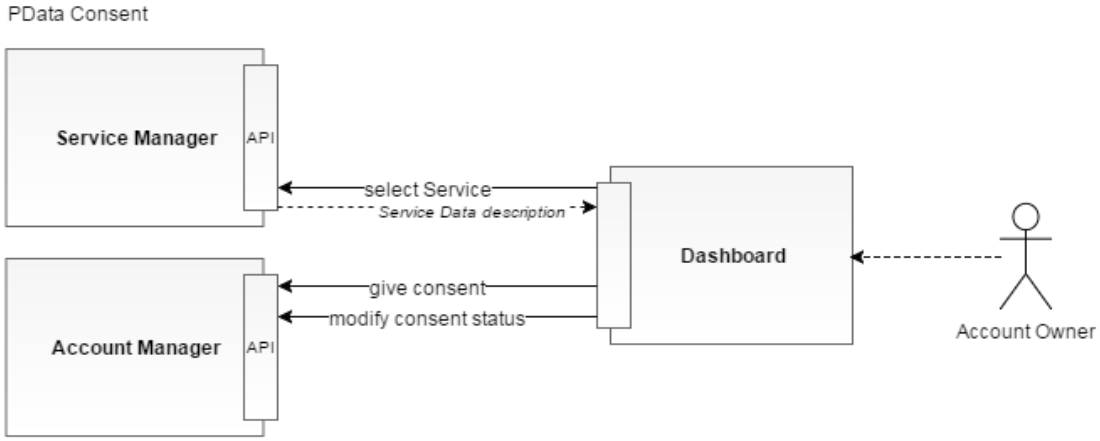

Figure 11 – Consent Manager interactions with other components

PData Manager

The PData Manager is responsible of providing a set of secure API for the retrieval and the storage of personal data by means of the CDV. Each external service, once registered with the CDV and associated with an account, can interact directly with the PData Manager.

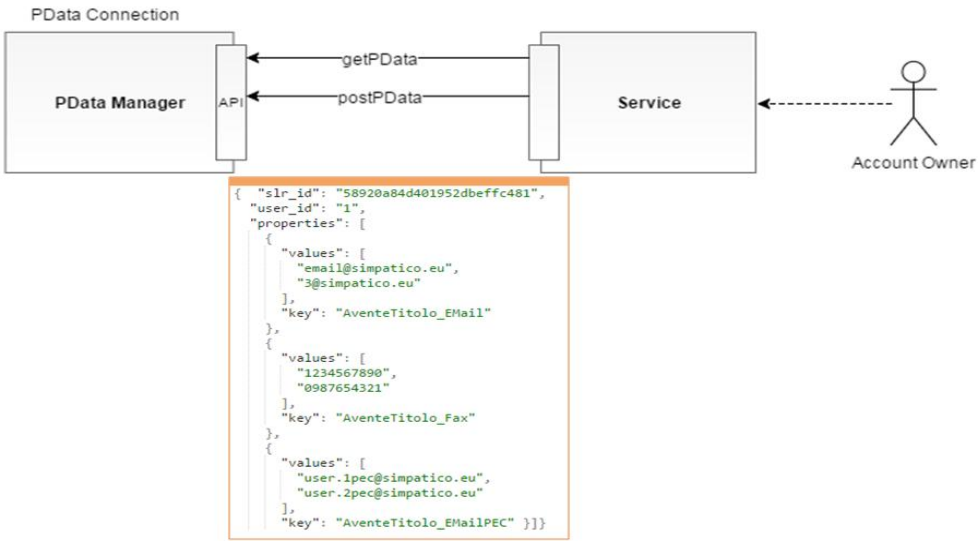

Figure 12 – PData Manager interactions with external service and related message exchanged

CDV Dashboard

The Citizen Data Vault provides a responsive graphical user interface (GUI) (Figure 13) that allow the citizen to manage his/her personal data and to grant and withdraw consent for access to personal

data. The CDV Dashboard interacts mainly with the PData Manager for the management of user data.

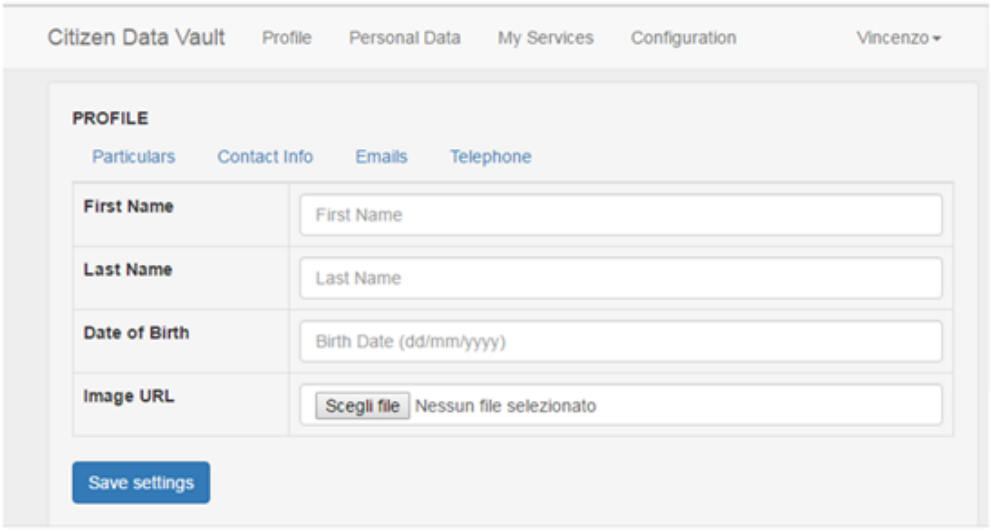

Figure 13 – CDV Dashboard GUI

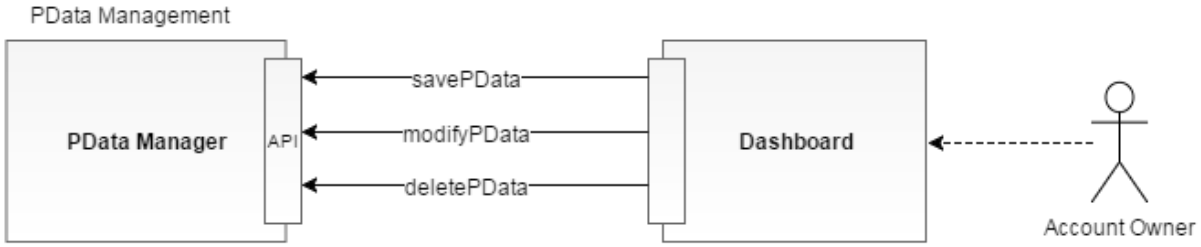

Figure 14 – Dashboard Interactions for data management

The Personal Data Service layer provides audit/log functionalities by means of the *Audit component*, by storing a set of information related the interactions of this layer with the dashboard and external components (service registration, service linking, data consent, data connection). Audit module provides user a means of tracking all the activities performed in order to interact with his/her personal data. The *Auth Manager* component is in charge of providing an OAUTH 2 mediation in the interaction of external components with the REST API exposed by the service layer.

The Personal Data Store layer is responsible of the secure persistence of personal data inside the CDV or other distributed data sources. To do this the PData Repository component interacts with the PData Source connectors and the PData Security Manager for a secure interaction with distributed data sources.

5.4 Interfaces

Each component provides a set of REST APIs, listed below.

PData Manager APIs

Technical specification of PData Manager APIs. These APIs provide functionalities for managing the Personal Data of an Account Owner. In addition, they provide functionalities for storing and getting these Personal Data, starting from the properties required by a previously linked service.

Method	/pdata-manager/api/v1/pData
Type	GET
Description	Gets all the Account owner's Personal Data in several formats. By default, the method returns a JSON array. If CSV format is specified, the response is the CSV representation of Personal Data as plain text (The first row contains the fields names, the following rows contain the related values).

Method	/pdata-manager/api/v1/pData
Type	POST
Description	Stores one or more Account owner's Personal Data

Method	/pdata-manager/api/v1/pData
Type	PUT
Description	Updates one or more Account owner's Personal Data

Method	/pdata-manager/api/v1/pData
Type	DELETE
Description	Deletes all the Account owner's Personal Data

Method	/pdata-manager/api/v1/pData/download
Type	GET
Description	Gets the dump file of all the Account and related Personal Data, in either CSV or JSON. By default, the method returns a JSON file. If CSV format is specified, the file returned is a CSV representation of Account and related Personal Data as plain text (The first row contains the fields names, the following rows contain the related values).

Method	/pdata-manager/api/v1/pData/{conceptId}
Type	GET, PUT,DELETE
Description	Get, Modify, delete a specific Account owner's Personal Data

Method	/pdata-manager/api/v1/postPData
Type	POST
Description	Stores one or more Account Owner's Personal Data for the linked Service, starting from service properties and Service Link Record. Resolves the service properties to the mapped Account owner's Personal Data. The service must be already linked through the Account Manager APIs.

Method	/pdata-manager/api/v1/getPData
Type	GET
Description	Gets one or more Account owner's Personal Data for the linked Service, starting from service properties and Service Link Record. Resolves the service properties to the mapped Account owner's Personal Data. The service must be already linked through the Account Manager APIs.

Table 13 – PData Manager APIs

Account Manager APIs

Technical specification of Account Manager APIs. These APIs provide functionalities for managing the Account Owner's account. In addition, they provide functionalities for performing the linking process between a Service and the Account Owner's account.

Method	/account-manager/api/v1/accounts
Type	POST
Description	It creates a new Account

Method	/account-manager/api/v1/accounts/{accountId}
Type	GET, PUT, DELETE

Description	Gets, modifies, deletes an existing Account

Method	/account-manager/api/v1/accounts/{accountId}/serviceLinks
Type	GET
Description	Gets all the Service Link Records of an existing Account

Method	/account-manager/api/v1/accounts/{accountId}/serviceLinks
Type	POST
Description	Creates a new Service Link Record for an existing Account. It represents the link between a Service and the Account Owner's account. It is the result of the linkage process provided by CDV (Account Manager itself). The goal is to store and retrieve Account Owner's personal data through the properties names required by the service.

Method	/account-manager/api/v1/accounts/{accountId}/services/{serviceId}/serviceLinks
Type	GET
Description	Gets a specific Service Link Record of an existing Account and specific Service

Method	/account-manager/api/v1/services/{serviceId}/users/{surrogateId}/serviceLink
Type	GET
Description	Gets a specific Service Link Record of an existing Account and specific Service, starting from the service ID and surrogate ID

Method	/pdata-manager/api/v1/accounts/{accountId}/serviceLinks/{slrId}
Type	GET
Description	Gets a specific Service Link Record of an existing Account

Table 14 – Account Manager APIs

Service Manager APIs

Technical specification of Service Manager APIs. These APIs provide functionalities for managing the Service Registration and its data mapping.

Method	/service-manager/api/v1/pdatafields
Type	GET
Description	Gets the list of available Personal data fields from taxonomy

Method	/service-manager/api/v1/pdatafields/{id}
Type	GET
Description	Gets information of a specific concept

Method	/service-manager/api/v1/pdatafields/search/
Type	GET
Description	"Like " search for specific pdata fields

Method	/service-manager/api/v1/pdatafields/category/{category}
Type	GET
Description	List of pdata fields of a specific category

Method	/service-manager/api/v1/services
Type	GET
Description	Get the list of the registered services

Method	/service-manager/api/v1/services
Type	POST
Description	Registers a new services

Method	/service-manager/api/v1/services/search
Type	GET
Description	"Like " search for specific registered services

Method	/service-manager/api/v1/services/{id}
Type	GET
Description	Gets service description by service ID

Method	/service-manager/api/v1/ services/{id}
Type	PUT
Description	Modifies service description by service ID

Method	/service-manager/api/v1/ services/{id}
Type	DELETE
Description	deletes service description by service ID

Method	/service-manager/api/v1/services/{id}/servicedatamapping
Type	GET
Description	Get service data mapping by service ID

Table 15 – Service Manager APIs

5.5 Links

The implemented external APIs are documented in:

https://simpatico.eng.it/SwaggerUI/pdata.html
https://simpatico.eng.it/SwaggerUI/account.html

Source code is available on line:

https://github.com/SIMPATICOProject/CDV

5.6 Next Steps

In the following months several features will be finalised. The CDV dashboard will be completed to allow users to manage their personal data, adding further information, modifying them, managing the consent for all linked e-services. The dashboard will be the front-end of CDV and it will accessible with the same credentials valid to access to SIMPATICO platform. The Data Consent formalization process flow will be finalized and managed by means of CDV Dashboard. The CDV client to be integrated in the SIMPATICO Interaction Front End (IFE) will be finalized addressing all the requirements provided by the three pilot use cases.

6 Conclusions

This deliverable covered work done towards the objectives of WP2 during the first 12 months of the project, resulting in a basic version of Text and Workflow Adaptation components, and their connection to the Citizen Data Vault component. More specifically, the deliverable reported the Lexical and Syntactic Simplification, Text Analytics, Workflow Adaptation and Citizen Data Vault components. For each of these software components, their key functionalities, architecture and interfaces were described, along with a brief survey of the state of the art and, where applicable, an initial evaluation on relevant datasets.

Overall, work was completed as planned for Month 12, with all components developed, tested and demonstrated in project meetings. The general plan for all for these components is to finalise their integration with the Interaction Front End, while refining them for the pilot evaluations, and - in the long run - improve them towards their second version (Month 24). Specific plans for each component were discussed in their respective Sections.

References

Barlacchi G. and Tonelli S. ERNESTA: A Sentence Simplification Tool for Children's Stories in Italian. In Alexander Gelbukh, editor, Computational Linguistics and Intelligent Text Processing: 14th International Conference (CICLing 2013), Samos, Greece, 2013.

Bentivogli L. and Pianta E. Exploiting parallel texts in the creation of multilingual semantically annotated resources: the MultiSemCor Corpus Natural Language Engineering, Special Issue on Parallel Texts, Volume 11, Issue 03, pp. 247-261, 2005.

Brusilovsky, P. and Schwarz, E. "User as Student: Towards an Adaptive Interface for Advanced Web-Based Applications", In Proceedings of the Sixth International Conference on User Modeling, UM97, 1997.

Chen, D. and Manning, C. A Fast and Accurate Dependency Parser Using Neural Networks. In Proceedings of EMNLP 2014, 2014.

De Bleser, F., De Smedt, T., Nijs, L. NodeBox version 1.9.5 for Mac OS X. Retrieved March 2010, from: http://nodebox.net.

De Bra, P., Stash, N. and Smits, D. "Creating Adaptive Web-Based Applications", Tutorial at the 10th International Conference on User Modeling, 2005.

Dell'Orletta F., Montemagni S., Venturi G. READ-IT: assessing readability of Italian texts with a view to text simplification. Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies, 2011.

Dell'Orletta F., Wieling, M., Cimino, A., Venturi G. and Montemagni S. Assessing the Readability of Sentences: Which Corpora and Features? Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications, 2014.

Dominique Brunato, Felice Dell'Orletta, Giulia Venturi, and Simonetta Montemagni. Design and Annotation of the First Italian Corpus for Text Simplification. In Proceedings of The 9th Linguistic Annotation Workshop, pages 31–41, Denver, Colorado, USA, 2015.

Faruqui, M. et. al. Retrofitting Word Vectors to Semantic Lexicons. Proceedings of 2015 NAACL, 2015.

Flesch, R. The Art of plain talk. Harper, 1946.

Gasperin, C., Specia, L., Pereira, T. and Aluísio, S. M. "Learning when to simplify sentences for natural text simplification," in Encontro Nacional de Inteligência Artificial, pp. 809–818, 2009.

Glavaš, Goran, and Sanja Štajner. "Simplifying lexical simplification: Do we need simplified corpora." Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, 2015.

Guenther, C. W., Rinderle-Ma, S., Reichert, M., van der Aalst, W. M. and Recker, J. "Using process mining to learn from process changes in evolutionary systems," Int'l Journal of Business Process Integration and Management, vol. 3, no. 1, pp. 61–78, 2007.

Hallerbach, A., Bauer, T. and Reichert, M. "Capturing variability in business process models: the provop approach," Journal of Software Maintenance, vol. 22, no. 6-7, pp. 519–546, 2010.

Henricksen, K. and Indulska, J. "Adapting the Web Interface: An Adaptive Web Browser", Proceedings Australasian User Interface Conference 2001, Australian Computer Science Communications, Volume 23, Number 5, 2001.

Horn, Colby, Cathryn Manduca, and David Kauchak. "Learning a Lexical Simplifier Using Wikipedia." Proceedings of the 2014 ACL. 2014.

Kamp, H. "A theory of truth and semantic representation," In J.A.G. Groenendijk, T.M.V. Janssen, B.J. Stokhof, and M.J.B. Stokhof, editors, Formal methods in the study of language, number pt. 1 in Mathematical Centre tracts. Mathematisch Centrum.

Kincaid, J.P., Fishburne, R.P., Rogers, R.L., and Chissom, B.S. Derivation of New Readability Formulas for Navy Enlisted Personnel. Research Branch Report, 1975.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., Herbst, E. "Moses: Open Source Toolkit for Statistical Machine Translation," In Annual Meeting of the Association for Computational Linguistics, demonstration session, Prague, Czech Republic, 2007.

Li, C., Reichert, M. and Wombacher, A. “Discovering reference models by mining process variants using a heuristic approach,” in Proc. BPM’09, pp. 344–362, 2009.

Lohmann, S., Wolfgang Kaltz, J. and Ziegler, J. “Dynamic generation of contextadaptive web user interfaces through model interpretation”, In Proceedings of Model Driven Design of Advanced User Interfaces, 2006.

Lu, R. and Sadiq, S. W. “Managing process variants as an information resource,” in Proc. BPM’06, pp. 426–431, 2006.

McLaughlin, G. H. SMOG grading: A new readability formula. Journal of Reading, 12(8):639–646, 1969.

McNamara, D. S., Kintsch, E., Songer, N. B. and Kintsch, W. Are good texts always better? Text coherence, background knowledge, and levels of understanding in learning from text. Cognition and Instruction, pages 1–43, 1996.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 2013.

Montes García, R., De Bra, P., Fletcher, G. H. L., and Pechenizkiy, M. “A DSL Based on CSS for Hypertext Adaptation”, Proceedings of the 25th ACM conference on Hypertext and social media, pp. 313-315, 2014.

Narayan, S., Gardent, C. "Hybrid Simplification using Deep Semantics and Machine Translation," In Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, pp. 435-445, 2014.

OASIS, Web Services Business Process Execution Language Version 2.0 – OASIS Standard, 2007.

Palmero Aprosio, A., G. Moretti. Italy goes to Stanford: a collection of CoreNLP modules for Italian. ArXiv e-prints, September, 2016.

Paetzold, G. H., Specia, L. Benchmarking Lexical Simplification Systems. Proceedings of 2016 LREC, 2016.

Paetzold, G. H., Specia, L. LEXenstein: A framework for lexical simplification. Proceedings of 2015 ACL, 2016.

Paetzold, G. H., Specia, L. SV000gg at SemEval-2016 Task 11: Heavy Gauge Complex Word Identification with System Voting. Proceedings of 2016 SemEval, 2016.

Paetzold, G. H., Specia, L. Unsupervised Lexical Simplification for Non-Native Speakers. Proceedings of AAAI-16, 2016.

Paetzold, G. H., Specia, L. Vicinity-driven paragraph and sentence alignment for comparable corpora. arXiv preprint arXiv:1612.04113, 2016.

Paetzold, G. H., Specia, L. Lexical Simplification with Neural Ranking. Proceedings of the 15th EACL. 2017.

Ploesser, K., Peleg, M., Soffer, P., Rosemann, M. and Recker, J. “Learning from context to improve business processes,” vol. 6, no. 1, pp. 1–7, 2009.

Rinderle, S., Weber, B., Reichert, M. and Wild, W. “Integrating process learning and process evolution - a semantics based approach,” in Proc. BPM’05, pp. 252–267, 2005.

Ronzano, Francesco, Abura'ed, Ahmed, Espinosa Anke, Luis and Saggion, Horacio. TALN at SemEval-2016 Task 11: Modelling Complex Words by Contextual, Lexical and Semantic Features. Proceedings of the 10th SemEval. 2016.

Rosa, M. L., van der Aalst, W. M. P., Dumas, M. and ter Hofstede, A. H. M. “Questionnaire-based variability modeling for system configuration,” Software and System Modeling, vol. 8, no. 2, pp. 251–274, 2009.

Schonenberg, H., Weber, B., van Dongen, B. F. and van der Aalst, W. M. P. “Supporting flexible processes through recommendations based on history,” in Proc. BPM’08, pp. 51–66, 2008.

Scott A. Crossley, David B. Allen, and Danielle S. Mc-Namara. Text readability and intuitive simplification: A comparison of readability formula. Reading in a Foreign Language, 23(1):84–101, 2011.

Shardlow, M. "A Survey of Automated Text Simplification," International Journal of Advanced Computer Science and Applications (Special Issue on Natural Language Processing), 2014.

Siddharthan, A. "Syntactic simplification and text cohesion," in Technical Report, Computer Laboratory, University of Cambridge - UK, UCAM-CL-TR-597, ISSN 1476-2986, 2004.

Siddharthan, A. A survey of research on text simplification. International Journal of Applied Linguistics, 2014.

SIMPATICO Consortium. “Description of Work”. H2020 project for call H2020-EURO-6-2015. Grant Agreement number 692819, 2015.

Tonelli, S., Palmero Aprosio, A. and Saltori, F. SIMPITIKI: a Simplification corpus for Italian extracted from Wikipedia. Proceedings of the Third Italian Conference on Computational Linguistics (CLIC-it), Naples, Italy, 2016.

Tonelli, S. and Tran Manh, K. and Pianta, E. Making readability indices readable. Proceedings of the First Workshop on Predicting and Improving Text Readability for Target Reader Populations, 2012.

van der Aalst, W. M. P., Pesic, M. and Schonenberg, H. “Declarative workflows: Balancing between flexibility and support,” Computer Science - R&D, vol. 23, no. 2, pp. 99–113, 2009.

Zhu, Z., Bernhard, D. and Gurevych, I. "A monolingual tree-based translation model for sentence simplification," in Proceedings of the 23rd International Conference on Computational Linguistics, Iaşi, Romania, pp. 1353–1361, 2010.