EUROPEAN COMMISSION

Research Executive Agency (REA)

Inclusive, Innovative and Reflective Societies

HORIZON 2020

Simpatico

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

D3.2 – Basic Methods and Tools for User Interaction Automation

WP3	Front-end interaction and enrichment
Due Date:	28/02/2017
Submission Date:	28/02/2017
Responsible Partner:	HI Iberia Ingeniería y Proyectos (HIB)
Version:	1.0
Status:	Final
Author(s):	Raúl Santos de la Cámara (HIB), Tamara Martín Wanton (HIB)
Reviewer(s):	Giuseppe di Modica (BENG), Vincenzo Savarino (ENG)
Deliverable Type:	OTH: Other
Dissemination Level:	PU: Public

Version History

Version	Date	Author	Partner	Description
0.1	02/11/2016	Raúl Santos de la Cámara	HIB	Draft table of contents and delivery schedule.
0.3	10/02/2017	Raúl Santos de la Cámara, Tamara Martín Wanton	HIB	Revised table of contents post discussion at Santiago meeting. Inputs on sections on the Interaction Model and the Data Model. First inputs on the LOG, SF, DA and EE summaries of key functions and next steps.
0.4	13/02/2017	Raúl Santos de la Cámara, Tamara Martín Wanton, Enrique Sanz, , Unai López, Iván Pretel	HIB, DEUSTO	Added first draft of section 2 interaction model, Section 4 Interactive Front-End. Version for internal review of Sections 1 and 4 through 9.
0.5	15/02/2017	Raúl Santos de la Cámara, Tamara Martín Wanton	HIB	Section 2 LOG data model.
0.6	21/02/2017	Vincenzo Savarino, Giuseppe di Modica, Raúl Santos de la Cámara, Tamara Martín Wanton, Iván Pretel, Michele Trainotti, Raman Kazhamiakin	ENG, BENG, HIB, DEUSTO, FBK	Unified previous sections 1, 2 and 3 as per the request of reviewers, major overhaul of interaction model. Unification of design strategies for components to section 1.2. Addition of links for the components to subsections X.4 of each component.
0.7	23/02/2017	Raúl Santos de la Cámara	HIB	Format changes, fixes in the tables 2 and 3 in sections 1.1 and 1.2.1.
0.9	23/02/2017	Raúl Santos de la Cámara, Tamara Martín Wanton	HIB	Final draft for internal review. Changes in interaction data model and minor text updates.
1.0	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This deliverable contains original unpublished work except where clearly indicated otherwise.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 Programmer's Interface
CDV	Citizen Data Vault
CPD	Collaborative Procedure Designer
CTZ	Citizenpedia
DA	Data Analysis
EE	Enrichment Engine
IFE	Interactive Front-end
IN	Interactive Phase
LOG	Interaction Log (module)
LS	Landing Screen Phase
PA	Public Administration
REST	Representational State Transfer
SF	Session Feedback
TAE	Text Adaptation Engine
UP	User Profile
URL	Uniform Resource Locator
WAE	Workflow Adaptation Engine
WP	Work Package

Executive summary

This document is the deliverable **"D3.2 – Basic methods and tools for user interaction automation"** 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).

SIMPATICO addresses a strategic challenge towards the **innovation and modernization of the public sector**: the need to offer a more efficient and more effective experience to companies and citizens in their daily interaction with Public Administration (PA) by (i) offering a personalized delivery of PA online services; (ii) enabling a better comprehension of the complex processes and documents (forms, regulations, etc.) behind these services; (iii) engaging them to improve the administration processes and services. SIMPATICO's goal 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 and by promoting an active engagement of people for the continuous improvement of the interaction with these services.

This document deliverable reports the work undertaken in SIMPATICO for WP3 ("Front-end interaction and enrichment") in the course of the project period spanning from month 6 to month 12. In this period, the **first version of the SIMPATICO components for interaction automation were designed, implemented and preliminarily integrated**. The key results are the **interaction data model** for the SIMPATICO platform and a summary of the developments of each of the components developed in WP3 (**Interaction Front-End, Interaction Log, Session Feedback, Data Analysis and Enrichment Engine**). All of the prototypes are described and their interfaces and future plans for SIMPATICO outlined.

The current document is formally part of the work undertaken in the process of the tasks T3.2 ("Interactive front-end"), T3.3 ("Data/log analysis") and T3.4 ("Enrichment engine"). However, slight changes and revisions of the work undertaken in task T3.1 ("Modelling of interactions") during the period M6-M12 has been incorporated into this deliverable.

1 Design features

The current D3.2 is the second deliverable of work package WP3 in SIMPATICO. It collects the developments of the participants in the Work Package until the project month 12.

The goals of this deliverable are as follows:

- Primarily, providing a description of the first implementations of the modules produced in the course of the work period M1-M12 in the components that are provided for interaction modelling, monitoring and analysis (IFE, LOG, DA, SF and EE). This is the bulk of the document. The design is explained in section 1.2 and the details of the implementation are summarized in sections 2-6 of the document.
- Secondarily, providing updates on cross-cutting aspects of WP3 such as the user (citizen) interaction model (section 1.1) that defines the underlying sequence of interactions for a SIMPATICO system.
- Finally, providing some indications of the work that will follow during Y2 in the WP3. This is of crucial importance as this will be the last deliverable in WP3 until the ending of Y2.

It is important to note that the summaries of the components in sections 1.2 and 2 through 6 represent only a brief explanation of the overall work done, which includes prototypes, specifications and source code which is accessible over the Internet (and particularly in the SIMPATICO GitHub portal[1]) as documented in the appropriate sections.

In this chapter, we provide an outlook of different aspects that have been taken into consideration and that will be useful to understand the components to be presented afterwards. These design features are subdivided in (a) the underlying interaction model for the SIMPATICO system that was started earlier in WP3 and concluded in this deliverable and (b) the different design decisions for the individual components. These will be discussed in sections 1.1 and 1.2 respectively.

1.1 Interaction model

In this section we deliver a first integrated outlook of the Interaction Model that will be used in a SIMPATICO system. In order to define this we need to first provide agreements of several key aspects in interaction at the whole SIMPATICO WP3 level.

---

[1] https://github.com/SIMPATICOProject - SIMPATICO GitHub

The architecture for SIMPATICO as proposed in deliverable D5.1 is as follows:
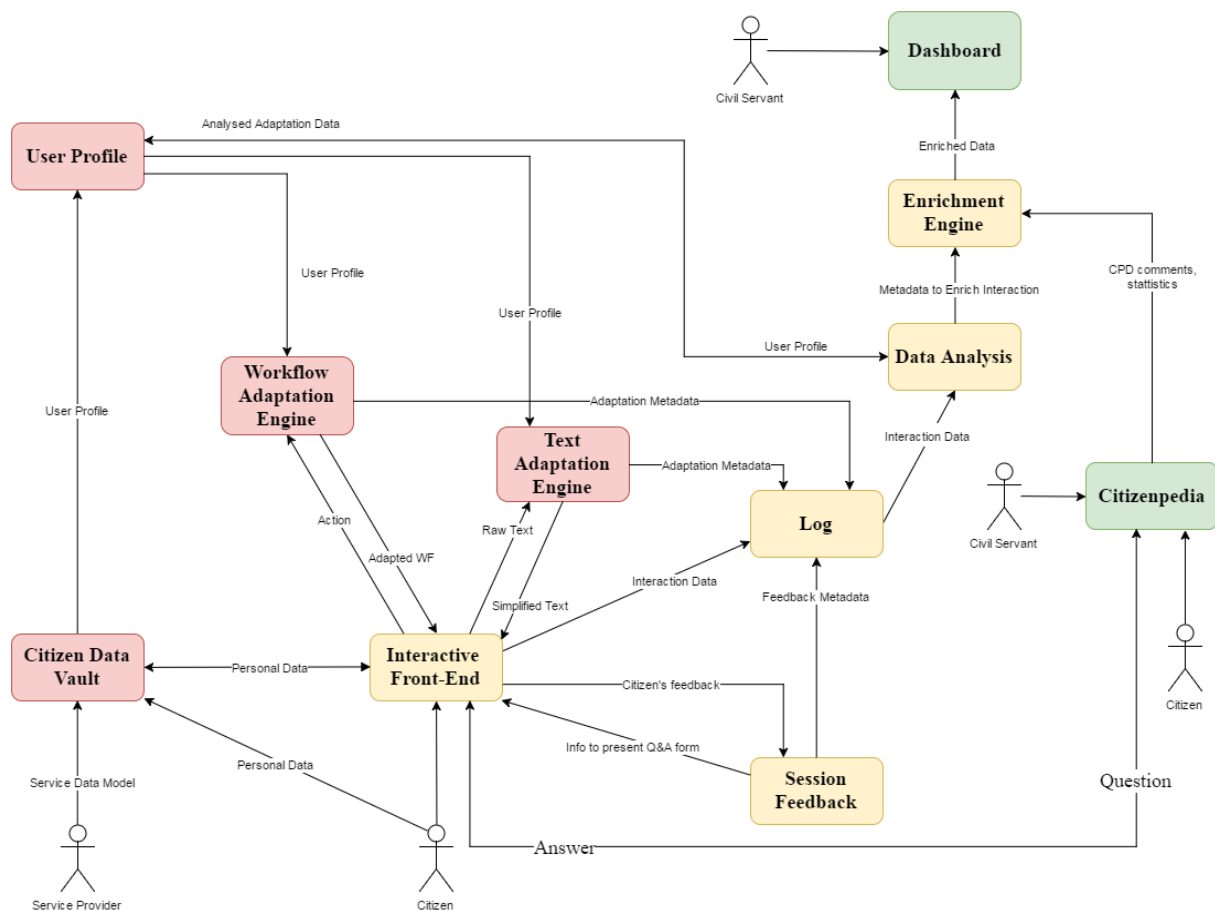

Figure 1 – SIMPATICO Architecture

There are three major interactive components in SIMPATICO: (a) the citizen interaction with the Interactive Front-End of the e-service (b) the Civil Servant interaction with the SIMPATICO Enrichment Engine and (c) the interactions with the Citizenpedia. In this report, and given the iterative design process we follow, we will focus on the details of the interaction model for the Citizen in the scenario (a) and as such, all of our references in this section 1.1 to *users* will mean *citizens*. The Civil Servant and the Citizenpedia interaction models is equally interesting for the overall SIMPATICO platform but it builds on the results obtained from the analysis in the feedback obtain from Citizens. Thus, it will be described in later documents.

As a start, we have to begin with the fact that in SIMPATICO, we can't work on an interaction model from scratch since our purpose is to complement, rather than to build fully new, e-services provided for the citizens by the Public Administrations. Thus, we first have to comprehend what is the offered interaction model of the legacy e-services and then we can propose how new elements proposed by SIMPATICO fit.

From our test interactions with the e-services and the more detailed analyses provided in deliverable SIMPATICO_D3.1 [1] we can see how, as a general norm, the proposed e-services use the following high level interaction model. We can see depicted the e-services proposed by the PAs for SIMPATICO in the Figure 2 below:
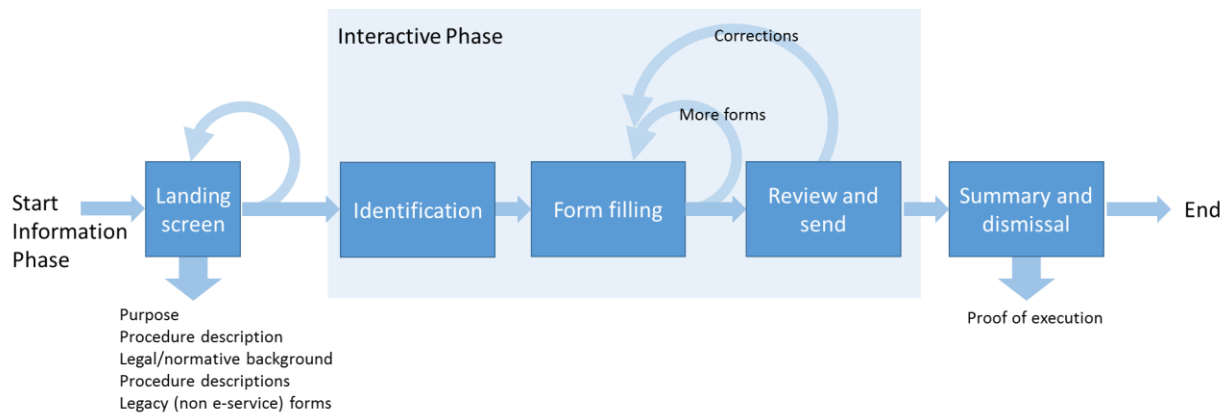
Figure 2 – Legacy e-service interaction model

We can see how the users begin their interaction with such e-services arriving at complex, information-packed **landing pages**. These are conceptually substituting the first contact with a real-world PA in which the user requests the basic information which is usually delivered in an annex or as a prelude to the paper forms. Since procedures are often complex and the legal language is necessarily verbose, this leads to the perceived complexity of these pages. We have signalled in the diagram with self-pointing loops that the information pages often contain many sub-pages or sub-divisions which are accessed separately, be it on different URLs or other organizational elements in the services: for example while the Trento services are often explained in one page but with clear functional separations[2], the Xunta de Galicia e-services[3] are presented in a single landing page which is then divided internally using tabs and the Sheffield e-services[4] feature a menu on the side of the e-service page that details the different options to navigate . Functionally, all serve the same purpose although the usability and discoverability aspects of every approach differ.

For SIMPATICO, the key takeaways for the interaction model here are:

- Information presented in these pages is very complex, both internally (as in the text that is used to describe each item of information) and structurally (many different elements of information are presented to users and different perspectives are used). SIMPATICO simplification features can help in simplifying both the text and clarifying some key terms.
- Given that such pages are always present, users possibly expect this to be a kind of "internal hub" for the service: the interactive part (e.g., parts of the procedure that require their input) are performed *from* here and possibly this is also the end area to land when execution of the process is done. Thus, possibly workflow adaptations at this level may be more risky.

The jump to the next step in the Figure 2 makes a clear break from the previous. In this, we can claim the user has decided to go into the execution of the service rather than merely acquiring

---

[2] http://trento.stu.globogis.eu/deroga-acustica-per-attivit%C3%A0-edilizia-temporanea → Trento information page for a city service. Note the different collapsible blocks that refer to different functional areas: *Descrizione dell'attività, Domande e comunicazioni* and others.

[3] https://sede.xunta.gal/detalle-procedemento?codCons=BS&codProc=607A → Xunta de Galicia e-service information page for BS607A. Here information is separated in different inner tabs within the page: *Solicitude, Documentación* and others.

[4] https://www.sheffield.gov.uk/education/information-for-parentscarers/at-school/attendance-information-for-parents-carers.html → Sheffield e-services for school attendance monitoring. Note how here the information stages are divided using a hierarchical menu on the side.

information. This change in orientation (from purely informational to interactive) is not always explicitly presented to the user but the dynamics of the process change considerably: before the user was *presented* with a great deal of information, while starting from now she/he is *asked* for a great deal of information. SIMPATICO can try and leverage this insight and inform users in a more explicit way to reduce any confusion.

The next step in the detected model is actually related to the **Identification** phase. Here, users are expected to provide some credential in order to validate their identity. This is by nature done very differently in unrelated e-service platforms. Regardless of the complexity of providing such credential (for example, the Xunta process can involve using a Smart ID card or browser certificate which is in itself very convoluted in many cases) the process is conceptually very simple: a credential is given and the user is granted access (or not) to a resource.

The process then jumps into the most time-consuming phase for the user: **Form** presentation and filling to gather the required information for the user. This is radically dependent on the e-service itself but some general aspects in the model can be extracted:

- The process consists on the filling of a series of information gathering requests, which are explicitly asked for in sequence (e.g., as "Steps" in the Sheffield e-services) or as a complex form in which different aspects are spatially separated (e.g., the complex forms in Galicia and Trento have large bounding boxes that group together related sets of information. This is why in the Figure 2 the **Forms** box has a self-referencing loop).
- The process consist always of one of three primitives, giving us options to group actuations depending on them:
  - Insertion of 'free text', that can be in itself (a) basic, immutable data such as name and date of birth, (b)data which changes rarely such as an address or (c) data which is specific for the request such a request for a given spa facility in the Galicia
  - Selection from a range of pre-defined options: yes/no questions or different application scenarios for the e-service. This is usually implemented using radial buttons in e-services.
  - Selection/de-selection of 'modifiers' that apply to one aspect and which are not mutually exclusive. This is usually represented as checkboxes in the forms.

After the filling in of all the required data, the user is usually redirected to a **summary, confirm and send** page in which the data introduced by the user is presented again for a quick review and the user is requested confirmation of the data. Then some interactive element to commit the request is presented.

After the user commits the changes, he or she is prompted to a debriefing page in which the committed data is presented again (this time without the ability to loop back into the form filing because changes are committed). Additionally, there is usually a way to generate and download/send the user a proof of execution of the procedure. After this, the user is dismissed and usually returned to the landing page.

Having described the main high level steps of the interaction model and focusing on the **citizen** interaction, the following table describes the level of applicability (1 - low, 2 - medium, 3 - high) of the main SIMPATICO components which are involved inside the mentioned interaction:

Table 1 – Applicability of the SIMPATICO components in the citizen interaction

	Interaction phase					
	Landing Screen	Identification	Form filling	Review and send	Summary and dismissal	Score
Interactive Front-End (IFE)	3	3	3	3	3	**16**
Workflow Adaptation Engine (WAE)	1	1	3	3	2	**10**
Text Adaptation Engine (TAE)	3	1	3	3	1	**11**
Citizen Data Vault (CDV)	1	1	3	3	1	**9**
User Profile (UP)	3	1	3	3	2	**12**
Log (LOG)	3	3	3	3	3	**16**
Session Feedback (SF)	1	1	2	2	3	**9**
Data Analysis (DA)	1	1	1	1	1	**5**
Enrichment Engine (EE)	1	1	1	1	1	**5**
Dashboard	1	1	1	1	1	**5**
Citizenpedia (CTZ)	3	1	3	3	1	**11**

According to the last column (Score), the most critical components from the interaction point of view are Interactive Front-End (it is the main interaction channel between SIMPATICO and citizens) and the Log component (it stores all the interactions produced by citizens).

The discussion from the Figure 2 until this point of the document describes the high level steps of the interaction model. However, more detail is needed to understand the SIMPATICO approach completely. For this purpose we require an analysis of the elements that is focused not only on the interactive flow proposed by legacy e-services but one that also integrates the novel features provided by SIMPATICO. This is presented in the following Figure 3.
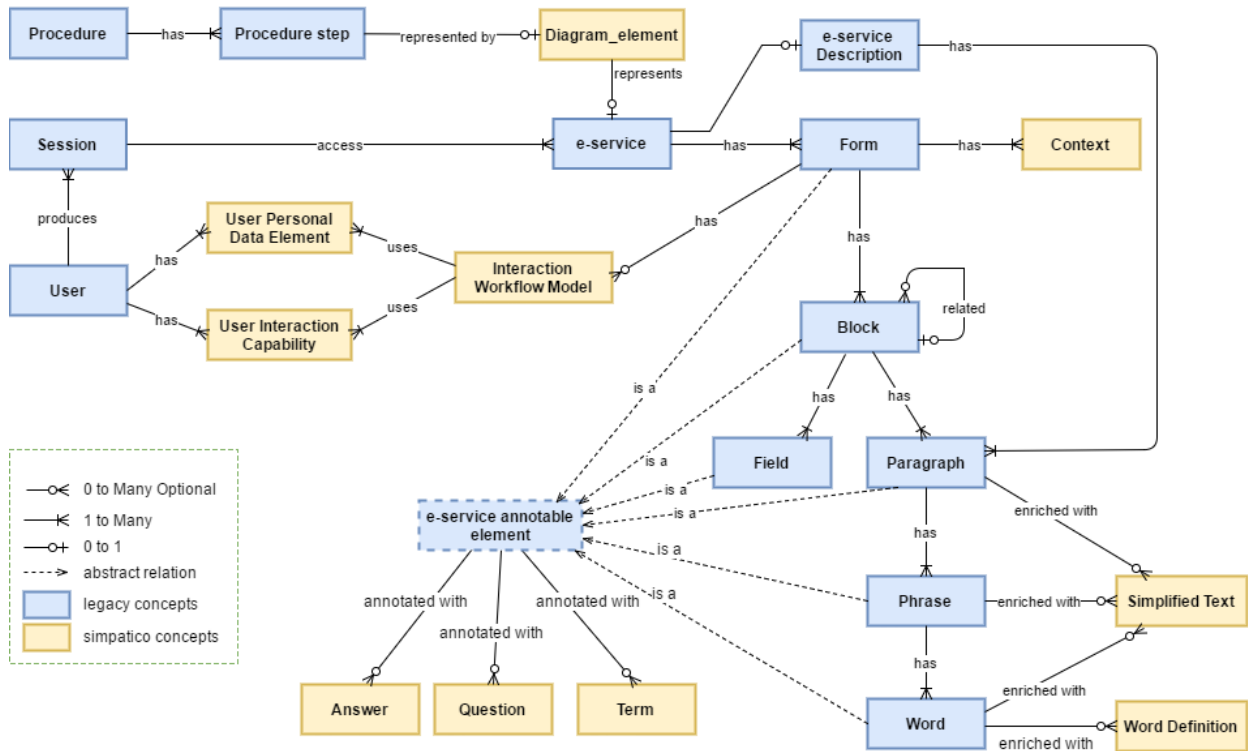
Figure 3 – SIMPATICO e-service interaction model

We see how the *Procedure* is divided into one or more *Steps* that, critically, have connections with important SIMPATICO features such as the *Diagram_elements* that are defined in the Collaborative Procedure Designer component (CPD). A *Diagram_element* may represent an *e-service*. Connecting to the high level overview in Figure 2, we see how *Users* produce *Sessions* associated with the *e-service* and how each of these relates to the different building blocks of the service interactive part (*Blocks* with *Fields* and *Paragraphs*, where the latter can contain *Phrases* and individual *Words*). These built-in elements of the legacy e-services connect to the Form Filling phase in the legacy model. They are complemented, though, with new SIMPATICO elements such as text-based help (*Simplified Text* and individual *Word Definitions*) that provide the user with augmented features.

In parallel with this, and with great importance to some of the analysis components in WP3 and WP4 such as the Enrichment Engine and the Citizenpedia, the users can interact through the Question-Answering system features. These are generalised by the higher level *e-service Annotable Elements* that signal all of the interactive elements susceptible of being annotated with questions and answers coming from users.

For the adequate understanding of this interaction model, several key concepts have been identified. These elements should be provided, consumed and enhanced by the SIMPATICO components, and taken into account during the citizen interaction model. In particular, when the citizen has to understand and interact the webpage: landing screen phase (LS) and interactive phase (IN). In the following table we present a definition of each of these concepts along with the most related components and their connection with the high level phases of the interaction as presented in Figure 2.

Simpatico

Table 2 – SIMPATICO interaction model concepts

Context	Concept	Owners/Consumers	Definition
General	**Procedure**	CTZ, IFE	It is a process defined by P.A. which contains several steps. It is represented by a CPD diagram
General	**Procedure step**	CTZ, IFE	It is an interaction step within the procedure. It is represented by a CPD diagram element (e.g. a node).
General	**e-Service**	CTZ, IFE	It is an electronic service used to complete one "procedure step" to achieve a procedure.
General	**Diagram element**	CTZ, IFE	It is a graphical element of a Diagram representing a specific step of a procedure. For instance it may represent: a) an entire e-service, a part of it or a subcomponent inside it (e.g. a form); b) a face-to-face interaction between the citizen and a civil servant; c) a decisional step which can divert the procedure flow
General	**Paragraph**	CTZ, IFE	It is a text describing a part of an e-service, law, or instructions inside them.
General	**Phrase**	IFE, TAE	In a HTML page or form it is a paragraph is text expressing conceptual unit, typically forming a component of a clause, delimited by dots or other tabulation elements (bullets, enumerations,...)
General	**Word**	IFE, TAE	In a HTML page or form it is a word single or part of a phrase.
Forms	**Field**	IFE, WAE	In a form it is an information the user has to fill in a e-service form, usually it as a caption and a editable element (e.g. text form, checkbox, combo box).
Forms	**Context**	WAE	It is a list of element depending to the actual e-service user interaction instance; each element can refer to a specific field content, it can refer to a specific user profile property or it can hold a value bound to the interaction status.
Forms	**Block**	IFE, WAE	In a form it is an atomic and coherent set of information the user has to fill in form (e.g. user registry, e-service specific information, fiscal information). The block contains a set of fields and paragraphs. The blocks can have relations and dependencies between each other with any context variable.
Forms	**Form**	IFE, WAE	It represents a single user interaction point and is bound to a unique e-service. The Form contains a set of blocks.

General	**e-service Description**	IFE, TAE, WAE	Element or group of elements inside an e-service which describes a part of an e-service or the e-service itself (e.g. Text explaining the aim and instructions of the e-service).
General	**Question**	CTZ, IFE	In Citizenpedia it is text representing an interaction element specification request (e.g. Procedure, Procedure Step, e-service, Form).
General	**Answer**	CTZ, IFE	In Citizenpedia it is a text reacting to an interaction element specification request. A request can have multiple answers.
Descriptions	**Term**	CTZ, TAE	A keyword defined and described in Citizenpedia or externally (e.g., Wikipedia) that can be related to an element of the e-service
Descriptions	**Word Definition**	TAE	A result of text enrichment applied to a single word/phrase
Descriptions	**Simplified Text**	TAE	A result of text simplification applied to a phrase or paragraph
Forms	**Interaction Workflow Model**	WAE	A simplified/adapted model of compilation of a form as a result of a workflow adaptation.
Forms	**User Personal Data Element**	CDV	Information stored and managed by CDV and used in the interaction
Forms	**User Interaction Capability**	UP	User profile characteristic used in the interaction for the workflow/text adaptation
General	**e-service Annotable Element**	IFE	The annotable element is a superclass that unifies all of the components presented by the IFE that can be annotated by the User.
General	**User**	IFE, CTZ, CPD	The User is the agent that interacts with any of the interactive components.
General	**Session**	IFE	A session is the execution through the IFE of an e-service. It comprises at least the fulfilling of one form. It is the basis for the definition of sequences of events that are recorded in the LOG.

As a summary of this subsection, we have presented the first attempt of an integrated citizen-oriented interaction model for SIMPATICO. Future work in WP3 will be simultaneously based on this model and, where needed, refining the details of it so that it fulfils better the needs of the components. In addition, a new interaction model for the Civil Servants interaction with their own interactive components such as the SIMPATICO Dashboard will be proposed for a more complete picture.

1.2 Component high level designs

From the Figure 1 we can see how the components associated with the WP3 are: Interactive Front-End, Session Feedback, Log, Data Analysis and Enrichment Engine. In the interim period between D3.1 and the current D3.2, these components have been outlined, designed and implemented.

While the implementation details will be discussed in the following section 2 and onwards, in this subsection we summarize the major design guidelines that have been followed to produce the components.

1.2.1 Log

In order to implement the required features presented in section 3.1, the LOG component was built using an instance of ElasticSearch [2]. This is a distributed, performance-oriented database that is commonly used for the storage and analysis of system logs in large web applications such as Facebook, GitHub and Netflix. Internally, ElasticSearch uses Apache Lucene [3] as an indexing engine and stores data internally using the NoSQL paradigm which makes it very efficient for distributing data with shallow structure. It permits very easy horizontal partitions of data (*sharding*) but on the contrary, it doesn't allow for distributed transactions.

All of these features are indeed attractive for a SIMPATICO data storage, as our data model (see Section 1.1 of this document) is not very deeply structured and we don't require a large distribution for the envisaged scale of the project.

Using the built-in ElasticSearch querying, a communications API is built on top of the component using Oracle Jersey [4], a well-known implementation of the JAX-RS 2.0 specification. All communications with the API are then RESTful and data is transmitted on top of this using JSON objects. This API is documented using swagger.io in order to automatize the process of generating documentation.

Regarding the format for the data to be stored within the LOG, this is a joint work in progress which is organised by the developers that require data to be stored in the LOG. The current draft snapshot of this is presented in Table 3, in which the events generated by the front-end are collected and later on analysed by the Data Analysis module.

Table 3 – M12 Draft model for events to be stored in the LOG

Event	Origin (front end, back end)	interaction concept	Logged Data	Description
paragraph_simplification	front end	Paragraph - SimplifiedText	userID (string), e-serviceID (string), paragraphID (string)	This event happens when user asks for the simplification of a paragraph. In LOG we will store for a given e-service how many paragraphs have been asked for simplification, and for a given e-service and a paragraph how many times users ask for simplification (if many users ask for simplification for the same paragraph then P.A must consider to change it

				in the appropriate way)
phrase_simplification	front end	Phrase - SimplifiedText	userID (string), e-serviceID (string), phraseID (string)	This event happens when user asks for the simplification of a phrase. The same consideration as for paragraphs applies for each phrase.
word_simplification	front end	Word - SimplifiedText	userID (string), e-serviceID (string), wordID (string)	This event happens when user asks for the simplification of a word. The same consideration as for paragraphs applies for each word.
free_text_simplification	front end	SimplifiedText	userID (string), e-serviceID (string), selected_text (string)	This event happens when user asks for the simplification of free text not tagged as a paragraph, sentence or single word.
citizenpedia_content_request	front end	Annotable Element	userID (string), e-serviceID (string), AnnotableElementID {Form, Block, Field, Paragraph, Phrase, Word} (String)	The user selects an annotable element (form, block, field, paragraph, phrase, word) in order to find questions related to that element or to formulate a new question.
citizenpedia_question_request	front end	Annotable Element - Question	userID (string), e-serviceID (string), AnnotableElementID {Form, Block, Field, Paragraph, Phrase, Word} (String), QuestionID (string)	The user selects a question related to an annotable element (form, block, field, paragraph, phrase, word)
citizenpedia_new_question	front end	Annotable Element	userID (string), e-serviceID (string), AnnotableElementID {Form, Block, Field, Paragraph, Phrase, Word} (String)	The user formulates a new question related to an annotable element (form, block, field, paragraph, phrase, word)
citizenpedia_new_answer	front end	Annotable Element - Question - Answer	userID (string), e-serviceID (string), AnnotableElementID {Form, Block, Field, Paragraph, Phrase, Word} (String), QuestionID (string)	The user inserts a new answer for a question related to an annotable element (form, block, field, paragraph, phrase, word)
citizenpedia_term_request	front end	Annotable Element - Term	userID (string), e-serviceID (string), AnnotableElementID {Form, Block, Field, Paragraph, Phrase, Word} (String),	The user selects a term of an annotable element in order to obtain the definition of the term from an external resource ()

			selected_term (string)	
user_eservice_ simplification	front end	Paragraph - SimplifiedT ext	*TBD*	Once a user accesses for second time to an e-service, TAE will obtain the words that a user has asked for simplification the first time and will automatically show the synonym of the word
session_start	front end	Session - eService	userID (string) , e-serviceID (string), timestamp (datetime)	Once a user starts a session the timestamp is stored
session_end	front end	Session - eService	userID (string), e-serviceID (string), timestamp (datetime), sessionDuration (int), averageTime (float)	Once a user ends a session the timestamp is stored. The average time of the use of this e-service is calculated
form_start	front end	eService - Form	userID (string), e-serviceID (string), formID (string), timestamp (datetime)	This event will store the timestamp of the beginning of the compilation of a form
form_end	front end	eService - Form	userID (string), e-serviceID (string), formID (string), timestamp (datetime)	This event will store the timestamp when a user end the compilation of a form. Average statistics are calculated
element_clicks	front end	AnnotableE lement	userID (string) , e-serviceID (string), AnnotableElementID {Form, Block, Field, Paragraph, Phrase, Word} (String), clicks (int)	Number of clicks on an annotable element
session_feedb ack	back end	User - eService	userID (string), e-serviceID (string), Complexity(TBD - SF component)	Once the user finishes with the session the Session Feedback component asks for some satisfaction measures.

1.2.2 Session Feedback

Session Feedback is one of the primary means for SIMPATICO to collect explicit information from the users about the quality of the offered e-services. This information is a fundamental source of knowledge to present to the Civil Servants which use the results provided by the Enrichment Engine through the system Dashboard. The information is captured from the citizens by means of interactive components which are presented with the IFE. Thus, the goals of this component are as follows:

- Generate strategies to collect feedback that can be used downstream in the Data Analysis to integrate with other sources of information (e.g., interaction metrics captured in the IFE) to

generate aggregated suggestions in the EE. These strategies should include elements that are specific to the current user and the last interaction that has occurred in the system.

- Generate atomic feedback collection elements (simple questions to the citizen end user) that fulfil individual topics of the strategy defined above.
- Generate a front-end for this feedback collection that aggregates the atomic elements into a feedback workflow that fulfils the strategy.
- Store the aggregated collection of this data into the LOG component for further processing.

The design of the Session Feedback will implement the following internal architecture:
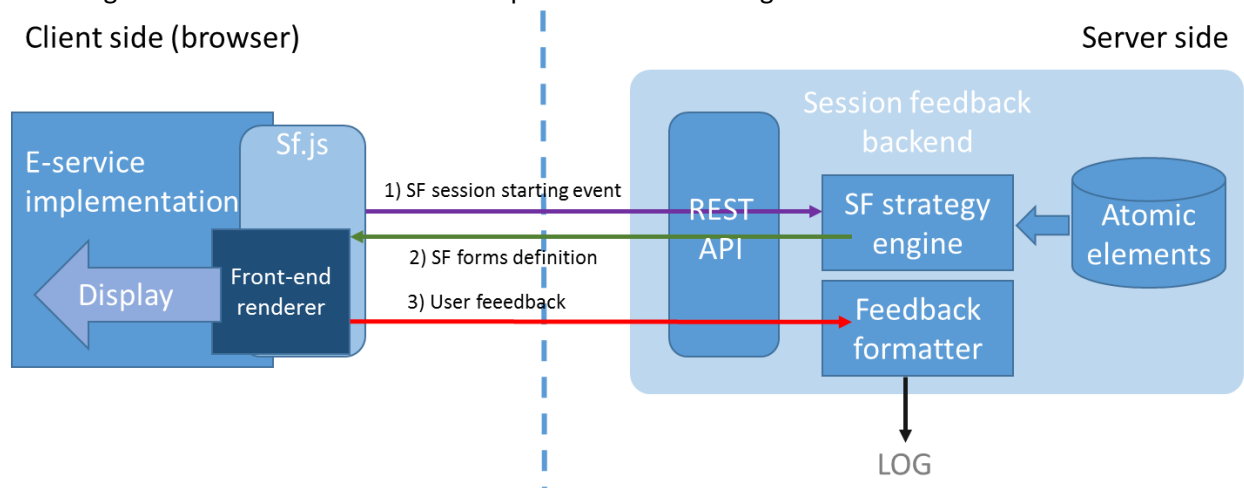

Figure 4 – Session Feedback internal architecture

We can see how the Session Feedback has components running both client-side (as part of the web application running in the browser for the e-service) and server-side. The operation is depicted with coloured arrows in the Figure:

1. The process is started by the user reaching the end of the e-service. This is notified by the IFE to the `sf.js` Javascript library which generates an event that is sent via the REST API of the SF component to the backend. The request, depicted as arrow (1), should contain elements that identify the session and the user gathered from the IFE.
2. In the backend, the request is analysed. The strategy engine calculates which feedback elements are most useful to elicit interesting feedback from the user. These are pre-generated and stored in a database of 'atomic' strategy elements. The most useful elements are selected and metadata to generate these elements is produced. This is sent in arrow (2) back to the client side.
3. The session feedback uses the received metadata in a rendering engine to produce an interactive form that is displayed to the user. When the user completes the form and clicks a 'submit' button, this is sent to the backend again via a call in the REST API. This is arrow (3) in the Figure 4.
4. Upon reception, the received feedback is post-processed and sent for storage in the LOG. This log entry contains the received feedback and some traces on the selected strategy for future evaluation of the strategy itself.

In the first implementation presented in this document, the sub-modules are just sketched in some aspects. The sf.js module exists and does the required data routing. For the front-end renderer we have used the jQuery [5] Javascript library which provides many standard and parametric UI

components we can use to render the feedback forms (e.g., modal dialog windows, text areas, radial buttons).

Server side, the REST API is a standard Oracle Jersey deployment. The Strategy Engine is fixed to show some standard questions with a range of simple atomic elements reusing some of the proposed strategies presented in SIMPATICO_D3.1 [1] Section 3.1. The feedback formatter in the first implementation is not deployed for simplicity, using a direct connection from the client-side to the LOG.
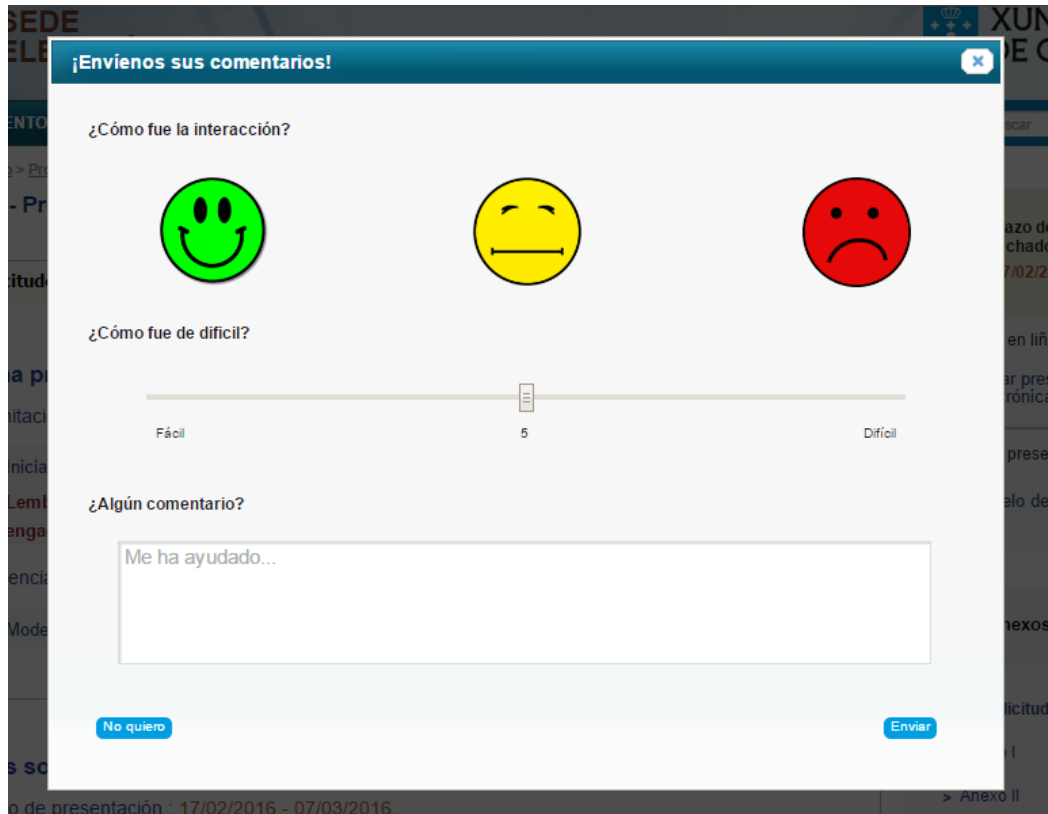

Figure 5 – Session Feedback client-side UI

In the Figure 5 we can see a rendering of the current status of SF client-side UI for the Galicia pilot. We can see how three different data collecting primitives are used (quantitative feedback as 'smiley faces' and as a satisfaction slider plus a free text form to collect more complex feedback). This is launched as a modal dialog on top of the e-service so that users don't lose context of the interaction when the feedback form is launched.

1.2.3 Data Analysis

Data Analysis uses the aggregated information from the interaction captured in the LOG incoming from the rest of the modules in the architecture of SIMPATICO to provide higher order interaction data that can be used downstream in the Enrichment Engine to present to the Civil Servant and improve the design of the e-service.

As such, Data Analysis is a metadata processor that uses a range of strategies linked to the detected Use Cases in the Enrichment Engine, to provide more aggregated data. This for example includes

merging the implicit analytics from the interaction, as generated by the IFE, TAE and WAE with the explicit feedback provided by the users in the SF. This will produce insight into the raw data, for example analysis of the LOG alone can tell how many times and when the text simplification was invoked, but joint analysis of a session LOG and SF data can also yield details on why it was invoked.

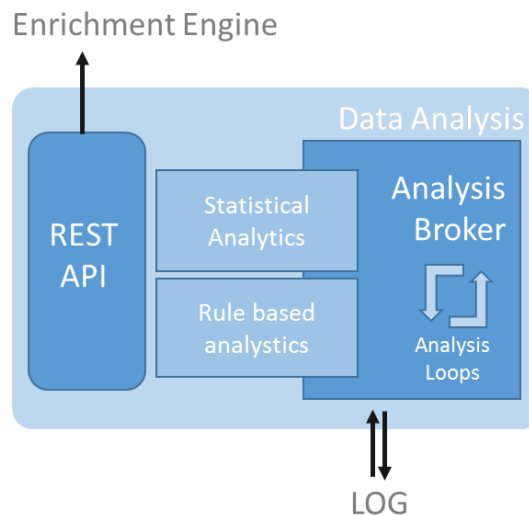The achieved design of the DA component in SIMPATICO follows the following internal architecture:

Figure 6 – Data Analysis internal architecture

The current model of operation of the Data Analysis component is iterative and batch-oriented: the available new data is processed on regular intervals to extract the new information. As it can be seen, the core of the DA module is an internal engine or Analysis Broker that gets data from the LOG component using a loop which defines a batch operation and then launches one or several analysis operations. The results of these analyses are then stored back in the LOG under a different branch of the storage. Upon request from the Enrichment Engine, these results are returned so that they can be used in the definition of updated services.

The system should undertake different kind of analyses as briefly outlined in [1]. For the current Y1 implementation, two distinct analysis engines have been considered and selected for implementation. One is a purely statistical analysis of the interaction data (times in different elements of the e-service workflow) which is calculated using the SciPy Python Library [6] which provides many widely used mathematical functions in a single package. For the moment only statistical functions are used. The other engine is just outlined at the closure of this document and it corresponds to rule engines that produce new results based on the gathering of new data and the application of logic reasoning. One well-known rule engine is JBoss Drools [7] that allows the definition of business rules in a standardized manner using the JSR-94 rule definition language.

1.2.4 Enrichment Engine

Situated at the end of the SIMPATICO feedback loop for the Civil Servants, the Enrichment Engine integrates the knowledge generated from the interaction analysis (through the IFE and SF through the DA) and the more process-oriented inputs from the Citizenpedia and the Collaborative Procedure

Designer. It aggregates these data into detailed reports that can be presented to the Civil Servant so that s/he incorporates the required changes into future revisions of the e-service for a better result.

The information produced in the Enrichment Engine is accessed through the system Dashboard. The dashboard is developed as a Kibana [8] module. Kibana is the visualization plugin for the ElasticSearch data storage system used in LOG, so integration is very straightforward. The internal architecture is as follows:
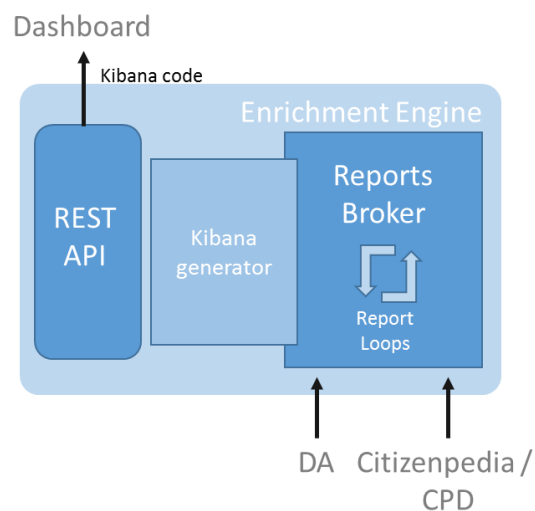

Figure 7 – Enrichment Engine internal architecture

As depicted in the Figure 7, the Enrichment Engine uses the same batch processing strategy explained for the Data Analytics. A series of Enrichment Use Cases are executed on a periodical fashion with the new data available. This analysis is produced in the form of new reports in Kibana that can be shown when the Civil Servant consults the Dashboard.

The current implementation of the Enrichment Engine can be seen in the following Figure 8:
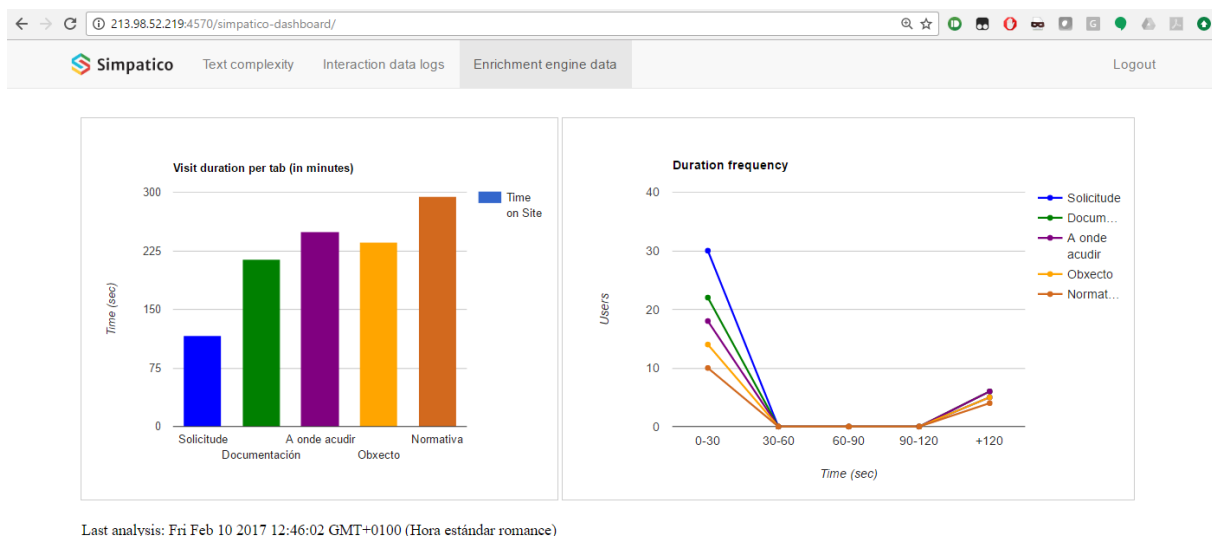

Figure 8 – Enrichment Engine internal architecture

It can be seen how the Enrichment Engine, as of the writing of this document, presents data which is mostly based on the statistical analyses undertaken in the DA. The depicted data is two representations of the per-tab visit durations to the Galicia IFE.

2 Interactive Front-End

2.1 Short summary of key functionality

This Interactive Front-End (IFE) component provides the following functionalities. Some of them fulfil several requirements defined in the deliverable D3.1, section 2.3.2:

- *Connections via REST API*: The IFE is able to make GET/POST/… RESTful request in order to communicate with other components of the SIMPATICO platform. This functionality fulfils requirement IFE.1.
- *User text capture*: The IFE is capable of capturing the text that a user has selected in a web form. This text shall be sent to other SIMPATICO components, e.g., to retrieve a simplified version of it. This functionality fulfils requirement IFE.2.
- *Modify text in a web form without page refresh*: The IFE is able to modify the DOM of a web form and modify text within it, without causing an entire refresh of the web form. The aim is to maintain a successful user experience. This functionality fulfils requirement IFE.3.
- *Connection with Citizenpedia*: The IFE enables to fetch questions, answers and comments posted in the Citizenpedia (WP4) related to the e-service that is being visualized automatically. It also enables to post questions in the Citizenpedia related to the current e-service in an easy manner. This functionality fulfils requirement IFE.4 and IFE.5.
- *Capture usability data*: The IFE can record interaction data, e.g. the time spent in each web form, the number of clicks per item, etc. This is achieved by capturing JavaScript events (e.g. onClick). This functionality fulfils requirement IFE.6.
- *Ease of integration with web browsers*: The IFE has been implemented as a JavaScript library that interacts through JavaScript events. This approach is browser-agnostic and eases the integration with popular browser in many of their recent versions. This functionality fulfils requirement IFE.7.
- *Ease of use*: The IFE has been implemented as a toolbar that appears on top of the legacy e-service. This toolbar contains a reduced set of icons (three to four) that enables the user to access the functionalities of the SIMPATICO platform. This functionality fulfils requirement IFE.4 and IFE.8.

2.2 Interfaces

This component exposes no interface, as it is the software piece that exposes the user interface of the SIMPATICO platform. However, it is a client of several components (see Figure 9).
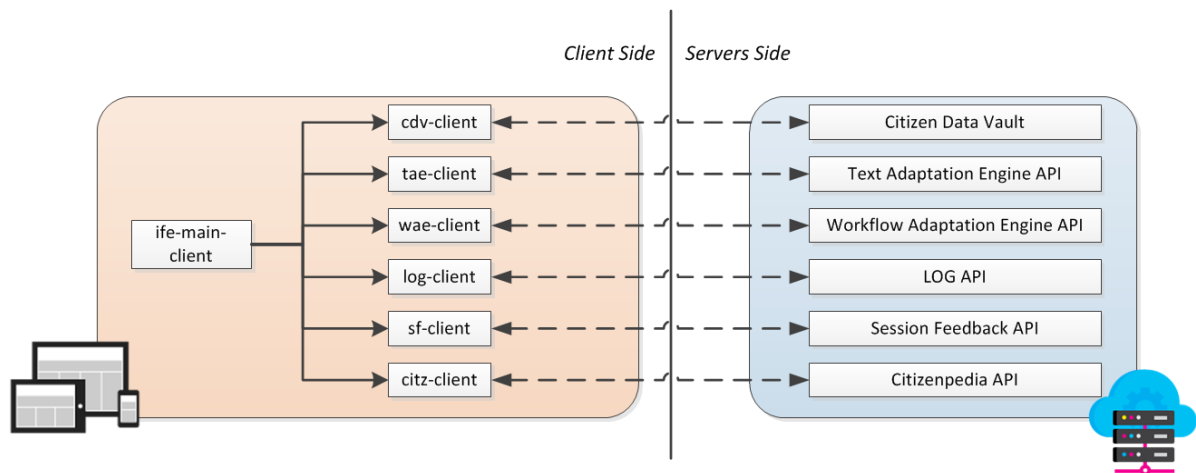
Figure 9 – Components used by IFE

2.3 Next steps

During this first phase, IFE has been developed following a modular approach (as exposed in Figure 9) in order to ease the development and integration of each feature. After defining it, the next steps (related to the quality check) to be followed in the development of this component will be:

- Bug fixes: ensure that the component is as stable and functional as possible.
- More browser tests: test the IFE with more browsers and different versions of them, in order to ensure a successful integration in situations beyond the scope of the project.
- Interaction and user experience enhancement: using the feedback gathered among the three pilots during the first phase the interaction will be enhanced.

2.4 Links

What follows are links to the IFE resources as of M12:

- Code repository (GitHub): https://github.com/SIMPATICOProject/IFE
- Swagger.io API reference: Not applicable.

3 Interaction Log

3.1 Short summary of key functionality

The Interaction Log component (LOG) is a central repository of information for the interaction matters in the SIMPATICO platform. It provides the following functionality:

- Collect information from modules in the architecture:
  - *Interactive Front-End:* from clicks and scrolls to time in different parts of the service.
  - *Session feedback:* textual and quantitative explicit feedback from the users as collected after the execution of the service.
  - *Workflow Adaptation Engine* and *Text Adaptation Engine*: storing metadata from the adaptation processes.
  - *Data Analysis:* storing the result of the execution of the analytics to extract knowledge from the raw data in the LOG.
- Internally provide a coherent data model that enables SIMPATICO to represent information regarding interaction of users.
- Provide a persistent data storage that is able to collect the information for further analysis either via automated means (by the Data Analysis component) or manually for developers to debug the behaviour of the system.
- Provide programmatic endpoints (APIs) to insert new data as explained above and to query for existing data.

3.2 Interfaces

What follows is a recount of the different methods available in the API of the LOG component, as generated by the swagger.io tool. The API follows the RESTful semantics for each of the HTTP methods. This is available with further information (code examples, parameters and error codes) in the link provided in section 3.4.

Table 4 – LOG Interfaces

Method	/remove
Type	DELETE
Description	Delete one document. JSON with 'id' key is required.

Method	/find
Type	GET
Description	Finds a document (a JSON file) in the database by words. A

Method	/insert
Type	POST
Description	Inserts one document. If in JSON there are '_id' key, the document will inserted with these id.

Method	/test
Type	GET
Description	This is a test API to check that the API and the component is up and running. An

	error code (500) is returned in case there are server errors and a correct code (200) returned in case everything is okay.

Method	/update
Type	PUT
Description	Updates a document's fields. JSON with 'id' and 'content' keys is required. If the field does not exist, it is added (preserving all previous ones).

3.3   Next steps

The presented LOG component technology allows for a basic usage. The internal data model (defined in the Section 3) is still an evolving work in progress and the APIs that allow developers to manipulate data are very basic. What follows is a recount of the next steps to be undertaken in the M12-M24 period in order to improve the implementation.

- *Alignment with the Interaction Data Model and provision of an internal data structure.* The component delivered at M12 uses a flat, JSON document based storage (that is, developers are free to include JSON files with arbitrary contents). This allows for basic experiments with other components such as the DA which are documented in this deliverable, but for more complex operation a standardized data model is required. This will follow the lines proposed in Section 3 of this document.
- *Provision of dedicated APIs.* In connection with the commentary in the previous point, the current API allows to insert and query full JSON documents with no account for the internal granularity of the data. When the internal data model is evolved, APIs will be constructed so that common operations (such as for example opening a new session from a user) can be accomplished with dedicated REST calls. This improves development and reusability of the component.
- *Performance testing and scalability plans.* The current deployment of the LOG component serves fairly simple testing means and thus consists of a single instance of ElasticSearch served by an uncomplicated API management mechanism. Since the tests of the SIMPATICO platform outlined in the project pilots and beyond (e.g., future prospective tests with real services) can involve a very large number of users and collected data, it is important that every component has a clear performance specification. For components that are both critical and central to many development flows such as the LOG, this is doubly important. Thus, a series of tests will be proposed and periodically executed on the component to check what is the expected performance and which could be the limits to propose solutions (e.g., distribution of the ElasticSearch backend in shards, provision of a more advanced API mechanism with priorities in query management, etc.).

3.4   Links

What follows are links to the LOG resources as of M12:

- Code repository (GitHub): https://github.com/SIMPATICOProject/logs
- Swagger.io API reference: http://213.98.52.219:4570/simpatico/

4 Session Feedback

4.1   Short summary of key functionality

The Session Feedback (SF) is the generator of interactive snippets presented at the end of the interaction for a session in the e-service with the purpose of gathering the user's opinion on the session. The key functionalities are as follows:

- Exposing an interface to the IFE so that it can the front-end side of SF (a Javascript function) can trigger the generation of a data capturing form.
- Generate in the backend the logic for displaying the form. This is currently done sending from the backend parameters so that an appropriate modal form with the required fields is displayed at the user's side.
- Display the form that was generated. In the implementation this is done by interpreting the parameters sent by the back-end into a JQuery modal dialog box (see Figure 5) that is presented on top of the legacy UI for the e-service.

4.2   Interfaces

What follows is a recount of the different methods available in the API of the SF component, as generated by the swagger.io tool. The API follows the RESTful semantics for each of the HTTP methods. This is available with further information (code examples, parameters and error codes) in the link provided in section 4.34.4.

Table 5 – SF Interfaces

Method	/selectform
Type	GET
Description	Select one form session feedback. Answers with the id of session feedback modal to use.

Method	/test
Type	GET
Description	This is a method to check the availability of the API.

4.3   Next steps

The Session Feedback available at month 12 at the closing of this deliverable has limited functionality. What follows are the next steps to be implemented following the delivery of this document.

- *Full client- and backend-side separation*. The current implementation has certain aspects of the proposed architecture simplified: the `sf.js` Javascript is not fully separated from the `simpatico.js`. Furthermore, the results of the feedback collection are sent from the Javascript directly to the LOG rather than sent to the backend for pre-processing.
- *Comprehensive definition of feedback primitives and strategies*. Some of the ones described in [1] are implemented in the current demo but some others (e.g., the usage of A/B testing techniques such as presenting different feedback forms to evaluate their usefulness) are only designed.

- *Incorporation of user details and session details to the strategy selection.* The current implementation is static with regards to these important elements to select a feedback gathering strategy. In the future, appropriate details of the user (such as past problems and past feedback) and of the interaction under question (such as where did the user spent more time or if text simplification was requested) will be implemented.

4.4   Links

What follows are links to the SF resources as of M12:

- Code repository (GitHub): https://github.com/SIMPATICOProject/logs
- Swagger.io API reference: https://213.98.52.219:4570/dist/sf.html

5 Data Analysis

5.1 Short summary of key functionality

Data Analysis (DA) is the module in charge of interpreting the stored interaction data in the LOG into higher level information that can be used upstream in the Enrichment Engine. The key functionalities are as follows:

- Collection of information from the LOG regarding past sessions interaction. For the current implementation, it collects discrete timing measurements for the different interactive elements in the Front-End (specifically, and pending the full detail of the IFE, the time spent in each tab in the Galicia front-end).
- Analysis of this data using statistical methods. For this, the statistical functions in the SciPy library are used to extract average times in different tabs and total visit durations.
- Storing the results of the analysis in the LOG using the provided REST API.

5.2 Interfaces

For SIMPATICO M12 RESTful interfaces were agreed to be defined for interaction between components that sit on client and server. Server-side components such as DA and EE that only interact with other server-side ones are not required to be defined until later stages of interaction.

Thus, as of the writing of this document the Data Analysis component does not provide RESTful external interfaces. Rather, and as explained in section 1.2.3, it operates in batch mode using the data available server-side in the LOG and inserting any useful results again in the LOG under a different branch of data. This data is then retrieved from the Enrichment Engine on demand directly from the LOG. As such, no interfaces are needed. In future steps we will consider interfaces for some operations (e.g., trigger an analysis on demand of the Enrichment Engine).

5.3 Next steps

After the completion of the M12 release, in next period (M12-M24) we will produce the following evolutions to our Data Analytics system:

- *Provision of a library of concrete analytics primitives*. The current system is essentially a proof-of-concept and connectivity trial that produces limited extra insight on the data stored in the LOG. It is expected that two main processes will lead the production of new analysis capabilities: (a) From the examination of LOG files from test trials with real end-users or with other persons, some potential areas in which analytics can be used can be inferred. (B) From the explicit requests of the Civil Servant end-users that will operate the Enrichment Engine we will learn more details about their needs at that level. From those high level needs we will extract the analytics primitives that will help us gain the required insight.
- *Incorporation of more sophisticated analysis techniques*. The usage for example of Machine Learning techniques is expected to yield more advanced insights into the captured data. We will implement these as parallel techniques to the statistical and rule-based analysis.

5.4   Links

What follows are links to the DA resources as of M12:

- Code repository (GitHub): https://github.com/SIMPATICOProject/logs
- Swagger.io API reference: Not applicable (internal API only)

6 Enrichment Engine

6.1 Short summary of key functionality

Enrichment Engine (EE) is the component that generates useful information for the Civil Servant based on the result of the Data Analysis and from the Citizenpedia. The implementation consists of two fragments: the front-end and the back-end:

- The backend is a batch process that consults the LOG for the results of execution of the Data Analysis and compiles them into statistical reports to present to the Civil Servant. This is done generating plots for representing the statistical data by using the SciPy library.
- The front-end is a tab inside the SIMPATICO Dashboard that renders the reports generated in the backend as instances of Kibana code. Kibana is the front-end layer for ElasticSearch and it is used to render the SIMPATICO Dashboard that presents other information than solely the Enrichment Engine's (e.g., results of Text Simplification).

6.2 Interfaces

For SIMPATICO M12 RESTful interfaces were agreed to be defined for interaction between components that sit on client and server. Server-side components such as DA and EE that only interact with other server-side ones are not required to be defined until later stages of interaction.

Thus, as of the writing of this document the Enrichment Engine component does not provide any RESTful external interfaces. Rather, and as explained in section 1.2.4, it operates in batch mode using the data available in the LOG which is incoming from the DA and is called programmatically from the Dashboard with dedicated interfaces (not RESTful). As such, no external interfaces are needed. In future steps we will develop RESTful interfaces for some operations such as generating reports on demand.

6.3 Next steps

Since it is located at the end of the processing pipeline for SIMPATICO, the Enrichment Engine has had less time to develop than the rest of components in the WP3 workflow. Thus, the available functionality is limited although a number of development lines are already defined:

- *More specific Use Cases*. The proposed *reports* that are the backbone of the results presented by the Enrichment Engine to the Civil Servant are based on specific Use Cases that might be useful for the Civil Servants themselves. Until M12, and given that the whole system was under construction, it was difficult to develop such use cases so we decided to implement a single, example Use Case which relied more on high level web analytics and statistical analysis than on actual needs from the end-users. In the future development, a more comprehensive range of such Use Cases will be derived from Civil Servants and also SIMPATICO developers and implemented into the already existing report generation engine.
- *Better connection with the Interaction Data Model and the e-service execution steps*. As part of the activities in WP3 and first documented in Section 1 of this document, a comprehensive interaction data model is being developed for SIMPATICO that can accommodate the different steps of the execution of the e-service by the citizens. When this is ready, it will enable the collection of data and execution of analytics that more purposefully reflect how users behave and interact on different stages of interaction. This is critical in the Enrichment

Engine as it will be presenting the Civil Servants with suggestions to improve the e-services so a presentation of the user feedback connected to these steps will be extremely useful.

- *Better connection with the CPD results*. The current implementation of the EE derives its data from mostly the statistical results of the DA and some minor stats from the Citizenpedia Usage. However, in order to have a maximum impact on the Civil Servants, more sources of information should be tapped such as the results of the Collaborative Procedure Designer. This is related to the comment above regarding the connection with the interaction model.

6.4   Links

What follows are links to the EE resources as of M12:

- Code repository (GitHub): https://github.com/SIMPATICOProject/ee
- Swagger.io API reference: Not applicable (internal API only).

7 Conclusion

This report is the final summary of the activities produced in WP3 in the period M1 up to M12. In this period, the work package has produced an initial deliverable [1] on the user interaction modelling which introduced the first ideas on the interactive components in the SIMPATICO platform and finally this D3.2 in which these ideas are implemented into the first release of interactive components.

In sections 1 we have presented iterations on the interactive model and design of components that were first presented in D5.1 in connection with the architecture. The Interaction Model is essential as it provides a shared understanding of many of the concepts such as e-service steps and interactive elements that are used in the analytics modules that produce the results of the platform.

In sections 2 through 6 we have presented the status of the modules that conform the interaction pipeline of SIMPATICO. This is the part of the whole approach that presents interaction elements for the citizens to operate the e-service (Interactive Front-End, section 2) and for these very same users to provide feedback on the system (Session Feedback, section 4). This is followed by modules that deal with interaction although they are not seen by the users: the system LOG (section 3) collects data coming from various sources and makes an archive for further analysis, while the Data Analysis (section 5) progressively refines this data into something that can be provided to Civil Servants through the use of the Enrichment Engine (section 6) that present e-service authors with reports that help them improve the e-services.

As of the end of the first year of the project all of these modules have been designed and introduced into the SIMPATICO architecture so that they collectively contribute to the project's main objectives. The implementations, although basic, demonstrate that it is possible to build an e-service infrastructure that supports these objectives.

7.1 Next Steps

In addition to the Next Steps presented per component in sections 4-8, it is important to note the global Next steps for the WP3 as a whole:

- Following the first implementation phase we have now the core components that should be packaged as the first SIMPATICO platform for the pre-validation phase of the system as a whole. Thus, the turn is now for WP5 activities regarding Integration. These were started before the writing of this document but they are expected to be resumed in full after the completion of this document.
- In parallel, the functionality of the components themselves will be expanded so that by the closure of the deadline for feature freeze for the First Iteration pilots they have a maximum of functionality.

The next release of results from WP3 will come at the end of year 2 of the SIMPATICO project in deliverable D3.3. For that document, the first version of the pilot tests will have been delivered and tested with real users so the document will focus on the improvements that will be introduced for the second iteration of pilots.

8 References

[1]   SIMPATICO Deliverable D3.1 "User interactions modelling and design" Santos de la Cámara [ed.] et al. 2016 SIMPATICO Consortium

[2]   ElasticSearch website   **https://www.elastic.co/**

[3]   Apache Lucene website   **http://lucene.apache.org/**

[4]   Oracle Jersey website   **https://jersey.java.net/**

[5]   JQuery website   **https://jquery.com/**

[6]   SciPy reference documentation   **https://docs.scipy.org/doc/scipy/reference/#**

[7]   JBoss Drools website   **https://www.drools.org/**

[8]   Kibana website   **https://www.elastic.co/products/kibana**