Building the Legal Knowledge Graph for Smart Compliance Services in Multilingual Europe

D1.2 Technical requirements analysis report

PROJECT ACRONYM	Lynx
PROJECT TITLE	Building the Legal Knowledge Graph for Smart Compliance Services in Multilingual Europe
GRANT AGREEMENT	H2020-780602
FUNDING SCHEME	ICT-14-2017 - Innovation Action (IA)
STARTING DATE (DURATION)	01/12/2017 (36 months)
PROJECT WEBSITE	http://lynx-project.eu
COORDINATOR	Elena Montiel-Ponsoda (UPM)
RESPONSIBLE AUTHORS	Filippo Maganza (ALP), Kennedy Junior Anagbo (ALP)
CONTRIBUTORS	Several colleagues from the Lynx consortium
REVIEWERS	Socorro Bernardos(UPM), Julián Moreno-Schneider(DFKI)
VERSION | STATUS	V1.0 |Final
NATURE	Report
DISSEMINATION LEVEL	Public
DOCUMENT DOI	10.5281/zenodo.1745370
DATE	28/11/2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780602

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	First draft	15/09/2018	Filippo Maganza
0.2	Document structure	28/09/2018	Filippo Maganza
0.3	Added Introduction	01/09/2018	Filippo Maganza, Kennedy Junior Anagbo
0.5	Added Chapter 1	07/10/2018	Filippo Maganza, Kennedy Junior Anagbo
0.6	Added Chapter 2	22/10/2018	Filippo Maganza, Kennedy Junior Anagbo
0.8	Added Chapter 3	5/11/2018	Filippo Maganza, Kennedy Junior Anagbo
0.9	Added Chapter 4, Annex 1, Annex 2 and Annex 3	9/11/2018	Filippo Maganza, Kennedy Junior Anagbo
1.0	Integration of reviewers' comments	28/11/2018	Filippo Maganza, Kennedy Junior Anagbo

ACRONYMS LIST

BB	Building Block
REST	Representational State Transfer
API	Application Programming Interface
HTTPS	Hypertext Transfer Protocol Secure
TCP	Transport Control Protocol
LKG	Legal Knowledge Graph
UI	User Interface
QoS	Quality of Service
DAG	Direct Acyclic Graph
TR	Technical Requirement
KPI	Key Performance Index
EU	European Union

EXECUTIVE SUMMARY

The impact of requirements gathering and specification in the development of software systems as in the case of the Lynx project cannot be overemphasized. Generally, software requirements are grouped under functional and technical (non-functional) requirements. The former specify the individual actions that a system or component must accomplish whereas the latter define the quality attributes that a system shall fulfill.

In this document, we focus our attention on the gathering and specification of the technical requirements of the Lynx system. The document also describes the methodologies and models used in the gathering and specification of these requirements.

The methods we used to gather the technical requirements include the analysis of the documentation presented in D1.1 and D4.1, the technical analysis of similar existing systems (e.g. MULTISENSOR), some interviews conducted with other partners and a survey to gather specific QoS requirements.

The model we adopted for the requirement specification is based on a set of rules: unambiguity, completeness, ranking, consistency and verifiability. All the requirements must comply with these rules.

From the requirements that have been collected, we were able to deduce some key qualities that the design of the Lynx platform shall comply with: the system shall be scalable, flexible, interoperable, the reliability of the data must be guaranteed, the resources of the platform (data and services) shall be accessible only to authorized clients and the architecture shall comply with REST architectural style.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF TECHNICAL REQUIREMENTS

1 INTRODUCTION

The main purpose of this document is the specification of the technical requirements of the Lynx system and the description of the methodologies and models that have been used to gather and specify those requirements.

The underestimation of the usefulness of requirements is among the most expensive and challenging errors to amend once a software system is completed [1] and, in the worst case, can lead to project failure.

Software requirements are grouped into two main categories: functional and technical (non-functional) requirements. Technical requirements define the qualities that a system shall fulfil while functional requirements determine the specific actions that a system or component must accomplish.

In practice, technical requirements are considered and characterized as hard to define and sometimes difficult to quantify. In this regard, they are often given little priority; although they are determinants for the success of software projects.

Integrability of these requirements during the design and development process must be realized in different design abstraction levels. For example, there are requirements that must be considered during the design of the architecture and others that are relevant only to the design of single components of the system.

1.1 STRUCTURE OF THE DOCUMENT

The rest of this document is organized as follows:

- Chapter 2 summarizes the methodologies and models used in the gathering and specification of the technical requirements; with its focus oriented to pilot partners and technological partners.
- Chapter 3 specifies and describes the technical requirements of the system. For each requirement, a detailed specification is provided with information about the necessity of the requirement and its importance level.
- Chapter 4 concludes the document and presents future work.

2 METHODOLOGIES AND MODELS

In this chapter, we first give a definition of system requirements in general and the term "technical requirement" in the Lynx project context; we then explain the adopted methodologies for the gathering and the specification of the requirements.

2.1 REQUIREMENT DEFINITION

According to [2] [3], a system requirement is:

1. A condition or capability needed by a stakeholder to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a solution or solution component to satisfy a contract, standard, specification, or other formally imposed documents.
3. A documented representation of a condition or capability as in (1) or (2).

It is important to observe that a requirement, in general, should state what is needed and not how it is to be provided.

2.2 TECHNICAL REQUIREMENT DEFINITION

A technical requirement is a requirement that specifies system operating characteristics and qualities used in the evaluating of a system. These requirements cover the non-functional aspects that the system shall fulfil such as performance, availability, interoperability, reliability, capacity and other quality attributes of the system.

Some examples of technical requirement could be:

- the system shall be available 99% of the uptime.
- the response time for the search of a document shall be less than 3 seconds.
- the system shall be accessible by remote users and client applications through REST APIs with the HTTPS protocol.

Quality attributes can potentially affect the structure of the system; therefore, the overall quality of service demands by the clients of the platform must be adequately met by the provisions of the system design: the system must be well structured and designed to meet the overall key quality metrics defined under the technical requirements as well as the business and mission goals of the system.

2.3 REQUIREMENT GATHERING

Requirement gathering is the process of discovering specific requirements pertaining to a system's design from clients and other stakeholders.

There are many methods and techniques proposed for requirement gathering. Amongst these techniques are: questionnaires/surveys, analysis of already existing literature, interviews and requirements workshops.

2.3.1 Preventive vs curative approach

Generally, there are two approaches that have been identified over the years in the literature relating to the achievement of software requirements. These are curative approach and preventive approach.

A curative approach is employed both during the development stages and after a software has been fully developed, tested and shortfalls determined in connection with the requirements in order to rectify that software; whereas, in preventive approach, the architecture and model task designs are structured in a careful manner to integrate requirements to prevent defects accompanied with it; thereby realizing a system which fulfils the technical requirements as described in [4].

A curative approach integrates well in agile software development methodology[1]; instead, a preventive approach is well suited for the traditional waterfall model.[2]

Taking into consideration the complexity of the Lynx project, we adopted a hybrid method that combines the curative and the preventive approaches; in particular, the curative approach will be used for all the technical requirements that cannot be precisely specified and quantified before the end of the T1.2; in all other cases we used the preventive approach.

2.3.2 Adopted techniques

The technical requirement gathering and specification process in Lynx took nine months to complete with different tasks scheduled involving pilot and technological partners. The techniques we adopted in gathering the Lynx's technical requirements are explained below:

- **Analysis of the pilots requirements and functional requirements of Lynx**: We analysed Lynx deliverables D4.1 (Pilots Requirements Analysis Report) and D1.1 (Functional Requirements Analysis Report) to get an initial idea of what the Lynx platform should accomplish.
- **Analysis of similar existing systems**: We analysed the documentation of some existing systems (e.g. MULTISENSOR) with similar operational objectives to the Lynx system to complement our initial information gathered in D4.1 and D1.1. This enabled us to gather the first list of basic technical requirements that would be refined.
- **Interviews with other project partners**: Pilot and technical partners provided useful answers to questions regarding their expectations of the Lynx platform. A rigorous analysis of the information gathered was carried out and a more detailed specification of requirements relevant to the Lynx platform was documented. The responses to the main questions posed to partners are summarized in Annex 1 of this document.
- **Survey questions to the customers**: Answers were solicited from pilot partners regarding specific questions related to QoS aspects required of the system. The questions and answers are recorded in Annex 2 of this document.

The schedule of the requirement gathering, and specification process is shown in the Gantt diagram in Figure 1.

Tasks	Mar. 2018	Apr. 2018	May. 2018	Jun. 2018	Jul. 2018	Aug. 2018	May. 2018	Oct. 2018	Nov. 2018
Study of similar already existing systems									
Analysis of D1.1									
Analysis of D4.1									
Circulation of the QoS questionnaire									
Analysis of feedback from the QoS questionnaire									
Specification of the requirements									
Write up of D1.2									

Figure 1 Gantt diagram of the requirement gathering and specification process

[1] Agile Software development can be considered as a development methodology which satisfy the circumstances where requirements are increasingly evolving [7] .

[2] Waterfall model in software development is an iterative approach where work progress tends to move towards one direction like a waterfall through the different phases of the software design [8].

2.4 REQUIREMENT SPECIFICATION

A written specification of the requirements gives a clear indication of the levels of quality one expects and how they are supposed to be tested. It also serves as a benchmark for specific standards that are applicable to the execution of the project making it possible to explicitly state the kind of compliance that is required.

By transforming requirements into specification, it becomes easier to minimize risks regarding the project execution and even after the project has been completed.

A software development project depends largely on the quality of the requirement specification since it is used across all project stages.

It is crucial to create a specification that documents technical requirements which can be easily understood by software architects and developers, enabling them to clearly understand the constraints in the design and the quality levels required from the system.

In a requirement specification process, there are rules that govern the description and documentation of the requirements. In this regard, we define a set of rules that all specifications must comply with. Our target is to present requirements that are not too rigid but achieve the result we expect from the system.

These are the rules we adopted for our technical requirement specification:

- **Unambiguity**: It helps to define a clear, concise and most importantly understandable specification that is in the interest of all the stakeholders. When a requirement is not clear, concise or grammatically correct, there is the tendency of misinterpretation and unverifiability. **A specification should not have ambiguous statements.**
- **Completeness**: A requirement is complete if it is **fully captured in just one place without misplaced or missing information**. Thus, the response of the software and all the necessary tasks that the project is envisioned to accomplish should be stated and well documented in one place.
- **Consistency**: In every jurisdiction of work, consistency is paramount in the execution of work plans. In our case, there is an absolute need to maintain extreme consistency in the use of words and terms so as not to have two requirement statements contradict each other. Thus, **words and terms should have the same meaning throughout the specification document.**
- **Ranking**: It is of a great importance to define clearly which requirements are of premium importance and those which are just supplementary requirements and therefore could be ignored if the project is, for example, constrained by budget and time. It is therefore expedient that we label our requirements **to show clearly those ones that will cause fatal deficiency if not executed.**
- **Verifiability**: A requirement specification that cannot be verified could be due to the requirement being incomplete or written poorly. A requirement that is not verifiable could result in the developers or testers verifying something other than what was intended. Furthermore, the system may pass verification but fail validation thus resulting in rework and impacting budget and schedule [5]. Example of a non-verifiable requirement is a statement like *The Lynx platform shall guarantee good performance with regards to all pilot use cases*. This statement cannot be verified because it is impossible to define or quantify the concept of *good performance*. In the light of this, **the specification of the requirements should be verifiable through basic possible methods** such as inspection, demonstration, test (instrumented) or analysis (to include validated modeling and simulation) [3].

3 TECHNICAL REQUIREMENTS

This chapter presents the technical requirements that the Lynx system shall fulfil to ensure the overall operation of the system. The chapter starts with the description of the template used for the requirement specification followed by the specification of requirements grouped by category.

3.1 SPECIFICATION TEMPLATE

The template adopted for the specification of the technical requirements (see Table 1) helps to comply with the specification rules stated in section 2.4.

ID	TR xx (Identification code of the technical requirement)
Short name	Short name of the requirement
Description	A brief description of the technical requirement
Priority	Must (it is compulsory and needs to be implemented) / Nice to have (it is complementary and therefore not obligatory)
(Optional) Examples	One or more examples of the technical requirement
Sources	One or more sources for the requirement

Table 1 Template used for documenting the requirements

3.2 SPECIFICATION OF THE TECHNICAL REQUIREMENTS

This section presents the specifications of the Lynx technical requirements. For a better organization of the requirements, we categorized each requirement according to a quality attribute of the system.

The testing and evaluation period will provide an appropriate verification and validation of the technical requirements specified in T1.2. In general, the validation of the requirements relevant to one particular BB (Building Block) will be carried out within the corresponding WP (Work Package), the remaining ones will be validated during T5.4 “Testing and evaluation plan”.

3.2.1 Modularity

Modularity requirements are highly prioritized requirements considering the complexity of the Lynx system; complexity can be reduced by designing the system as a set of components that can exhibit a different level of interdependence therefore hiding the complexity of each modular part behind an abstraction.

ID	TR1
Short name	Modular architecture
Description	The architecture shall be composed of a collection of building blocks. BBs are loosely coupled architecture components with only one main responsibility and a well-defined interface. BBs can exhibit different levels of interdependence and are divided into two categories: foundational and peripheral BBs. Foundational BBs form the infrastructure of the system and are unlikely to be changed or modified while peripheral BBs depend strictly on foundational BBs for their operation and functioning.
Priority	Must
Examples	The functionalities concerning authentication and those concerning named entity extraction cannot belong to the same peripheral BB.
Source	The Lynx project has complex and diversified business requirements; thus, it involves many different teams with different skills and technical backgrounds. We would like to emphasize these skills by adding a level of technical abstraction between the architecture components that enables each team to work on a particular BB independent of other teams.

TR 1 Modular architecture

ID	TR2
Short name	Less technological constraints on the design of BBs
Description	The system shall allow for the design of BBs with different technology stacks and patterns.
Priority	Must
Source	The Lynx project has complex and diversified business requirements; thus, it involves many different teams with different skills and technical backgrounds. We would like to emphasize these skills by adding a level of technical abstraction between the architecture components that enables each team to design each BB without technological constraints.

TR 2 Less restriction on the design of BBs

ID	TR3
Short name	Independent deployment of the BBs
Description	The system shall allow for the independent deployment of each BB.
Priority	Must
Examples	Development team A wants to test the 'NER' peripheral BB they have just finished implementing while development team B is still working on the 'Class' peripheral BB. Development team A can proceed to deploy and test the 'NER' BB independently from development team B.
Source	We want to make different teams more independent in the development and testing of the BBs.

TR 3 Independent deployment of the BBs

3.2.2 Interoperability

Interoperability is the characteristic that helps several systems to smoothly interact with each other. This quality attribute is a key factor for the success of the Lynx project since the Lynx platform will have different types of clients and different systems to integrate with.

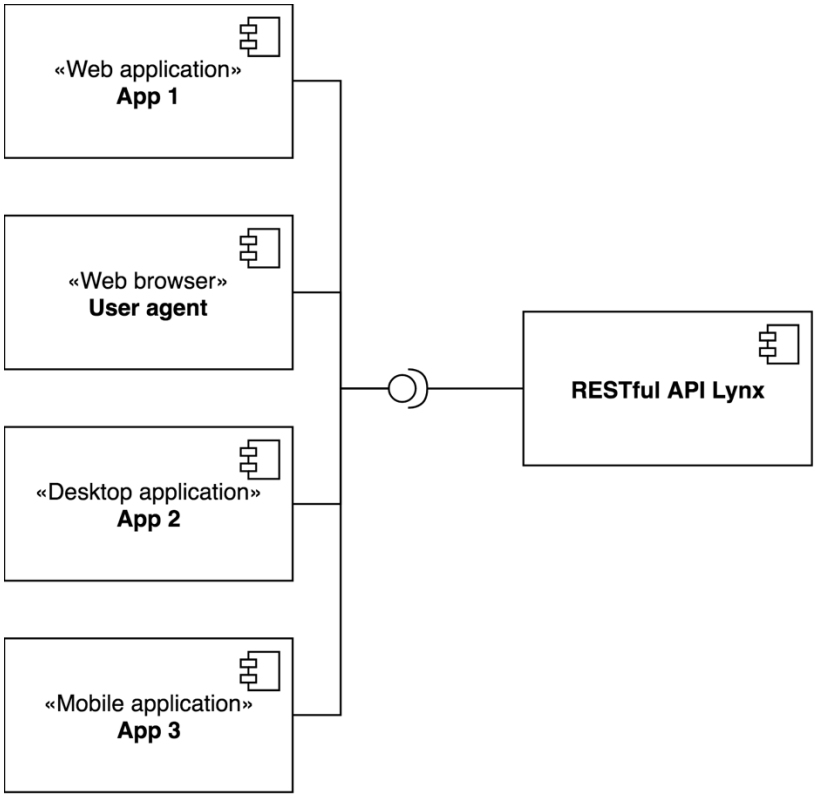

Figure 2 The Lynx system and different types of clients

Figure 2 shows different types of applications (boxes on the left) that smoothly interact with the Lynx system (depicted by the box on the right).

ID	TR4
Short name	RESTful architecture
Description	The system shall comply with the REST architectural style. REST is a software architectural style that defines a set of constraints to be used for creating web services.
Priority	Must
Sources	Interoperability between BBs shall be ensured. Web services that conform to the REST architectural style provide interoperability [6].

TR 4 RESTful architecture

ID	TR5
Short name	Well documented Lynx APIs
Description	The APIs of Lynx shall be well documented (all the attributes of the API must be accurately explained).
Priority	Must
Sources	The documentation of the Lynx APIs is very important as it is useful for all systems that have an integration with Lynx platform.

TR 5 Well documented APIs

ID	TR6
Short name	Well documented BBs APIs
Description	The APIs of the BBs shall be well documented (all the attributes of the API must be accurately explained).
Priority	Must
Sources	The documentation of the APIs of the BBs is very essential for the architects and the developers during and after development of the system.

TR 6 Well documented BBs APIs

3.2.3 Flexibility

The ability of a system to adapt well to both internal and external changes (uncertainties) influencing its worth delivery, at the right time and in a cost-effective manner is referred to as the flexibility of the system.

One of the main objectives of the Lynx project is to provide a business environment for small-medium enterprises solving problems related to compliance; thus, the three pilots could only be a starting point

for the project and therefore, the number of use cases of the platform could increase over time requiring a flexible system for adaptation.

The diagram in Figure 3 illustrates how the agile methodology can be utilized in combination with the Lynx flexible architecture for better adaptation and integration of new use cases, and the expansion of already existing use cases.

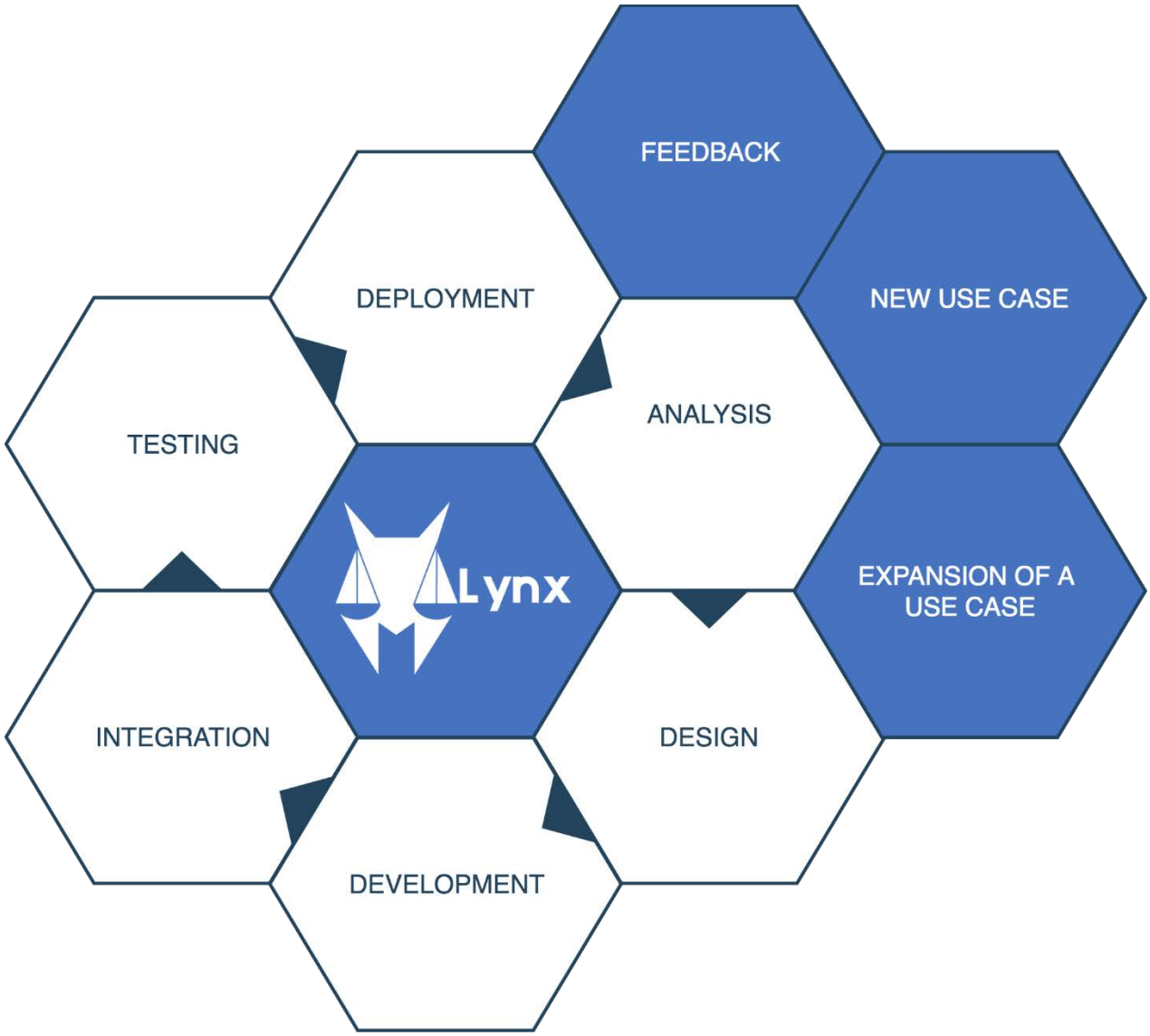

Figure 3 Taking advantage of a flexible system using the agile methodology

ID	TR7
Short name	Flexible architecture
Description	The architecture shall be designed to support new features with minimum efforts. A new feature can be assumed to be a new peripheral BB to be added to the system or a modification of an already existing one. In all cases, the new feature must not impose changes on the overall architecture and infrastructure of the system.
Priority	Must
Examples	Company A wants to analyse a contract in the same way described in the *'contract analysis' use case 1.3* (see D4.1) but the contract and the retrieved information should be in French; therefore, it is important that only the peripheral BB responsible for language translation shall be modified to provide French translation.
Source	We won't limit the business cases of Lynx to what we initially planned, so, the architecture shall be designed to be flexible. The achievability of this requirement depends to a large extent on TR 1 Modular architecture. In general, the Lynx system must respond to potential future uncertainties in a way to sustain or increase its value of service delivery.

TR 7 Flexible architecture

ID	TR8
Short name	Workflow model
Description	The workflows shall be organized into a sequence of tasks that process information; each task is associated with exactly one peripheral BB of the system.
Priority	Must
Examples	An example of a workflow represented using a DAG is shown in Figure 4.
Source	D4.1 (Pilots requirements analysis report), D4.2 (Initial version of workflow definition)

TR 8 Workflow definition and organization

ID	TR9
Short name	Easy specification of new workflows
Description	The system shall provide a way to easily define new workflows or modify already existing ones.
Priority	Nice to have
Source	The system shall be designed to incorporate with ease new specification of workflows.

TR 9 Easy specification of new workflows

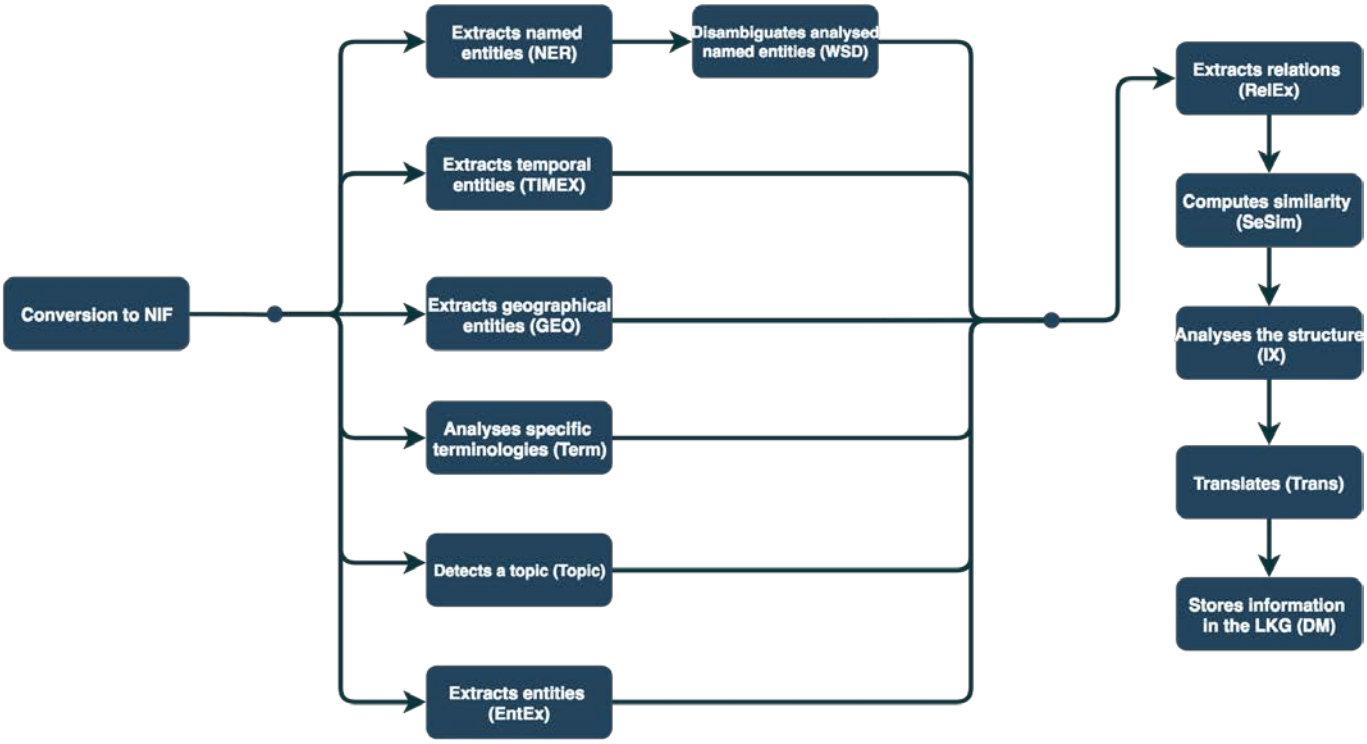

Figure 4 A possible representation of a workflow using a DAG (Direct Acyclic Graph)

Figure 4 shows a workflow represented using a DAG; each block represents a task that the system has to execute.

3.2.4 Scalability

A system is considered scalable when it can efficiently handle an increasing workload, or it can be expanded to absorb the growth.

For the Lynx project, scalability is of great importance because the overall workload of the platform has the potentiality to grow substantially over time, and it could also have a high variance during short intervals of time.

ID	TR10
Short name	Scalable system
Description	The system shall be scalable. The adding on of resources to the system shall be easily accomplished without difficulties and requiring no changes to the architecture.
Priority	Must
Source	The design of the Lynx system to scale in response to increase demand is very important since future growth in workload is envisioned to augment (Annex 1 – Q & A). The system will need to scale to match the growth of the business.

TR 10 Scalable system

3.2.5 Availability

The availability of a system is its ability to guarantee an authorized entity the access to a system, a service or a resource whenever it requires it.

ID	TR11
Short name	System availability
Description	The system shall be available 99,9% from Monday-Friday between the hours of 8:00 - 23:59; with maintenance outside business hours preferably during European night time or weekends.
Priority	Must
Source	Survey (Annex 2)

TR 11 System availability

3.2.6 Security

Security, in general, describes the ability of a system to guarantee access to resources only to authorized clients and to be free from potential external attacks or its ability to be resilience against these attacks or harm.

ID	TR12
Short name	Fine-grained access control
Description	The resources (include both services and data) of the system shall be accessible only by authorized users and client applications.
Priority	Must
Examples	The Cuatrecasas client application will be authorized to access Pilot 2's services but not Pilot 1's services. The Lynx administrators will be the only ones who can create, delete and update the Lynx users' identities and roles.
Source	Lynx will be a multiclient solution; therefore, the platform must control the access to the resources to ensure that only authorized clients can access a particular resource.

TR 12 Fine-grained access control

ID	TR13
Short name	Models trained with confidential data
Description	The system, for the execution of any workflow, shall not consider trained models and information built from confidential data that are not accessible to the client that made the request.
Priority	Must
Source	Vienna Plennary Meeting (November 2018) discussion.

TR 13 Models trained with confidential data

3.2.7 Usability

Usability refers to the degree to which a software system can be effectively used by clients to harness all its potentiality with maximum efficiency and satisfaction to the intent of achieving quantified objectives.

It is necessary to design the Lynx system to provide users an easy to use and an easy to remember user interface to satisfy users' desire.

The initial user interfaces that the Lynx platform shall provide are the following:

- **'Free search' UI**: provides an interface for searching public documents available in the LKG.
- **'Back office' UI**: provides an interface for the Lynx administrators for managing documents and user identities.

ID	TR14
Short name	Attractive and easy to use 'Free search' UI
Description	The 'Free search' UI of the Lynx platform shall be attractive and easy to use.
Priority	Nice to have
Source	An attractive and easy to use interface for all citizens who want to search among the Lynx's public documents is a good exposure and an avenue of advertisement to attract potential clients and premium users.

TR 14 Attractive and easy to use 'Free search' UI

ID	TR15
Short name	Simple and clear 'Back office' UI
Description	The 'Back-Office' UI of Lynx shall be simple and clear. Each operation a user shall perform should be well described.
Priority	Nice to have
Source	Administrators of Lynx shall not be confused when performing operations like the registration of a new application or the deletion of a document related to the LKG.

TR 15 Simple and clear 'Back office' UI

ID	TR16
Short name	English localisation for all the Lynx's UIs
Description	The text of the UIs of the Lynx platform shall be localized in British English.
Priority	Must
Source	Lynx is a project that aims at EU small/medium enterprises; thus, the British English localization is important since it is widely used in the EU.

TR 16 English localisation for all the Lynx's UIs

ID	TR17
Short name	EU languages localization for the 'Free search' UI
Description	The text of the 'Free search' UI of the Lynx platform shall be localized into other EU languages (German, Dutch and Spanish).
Priority	Nice to have
Source	To increase Lynx usability and attract potential new customers, it might be useful to localize the Lynx platform user interface in several languages.

TR 17 EU languages localization for the 'Free search' UI

3.2.8 Performance

The performance of a system is a measure of several contributing factors depending on the output desired. In general, system performance can be characterized by response time, processing speed, latency, throughput, and bandwidth.

In addition to the performance requirements described below, we've also gathered some useful information to estimate the initial workload of the system; these information can be found in Annex 2.

ID	TR18
Short name	Usage limits
Description	The system shall be able to limit the number of requests that each client shall make in a defined interval of time.
Priority	Must
Examples	The Cuatrecasas client application shall be allowed to make a certain number of requests within a defined time frame during peak hours of the day. If the limit for the number of requests is reached, no more requests from Cuatrecasas shall be accepted until the next allocated time frame.
Source	It is necessary to put a limit on the number of requests in order to prevent the congestion of the system.

TR 18 Usage limits

ID	TR19
Short name	Average response time (Pilot 1.1, 1.2)
Description	The system shall be able to process and return search results within 500ms for the Pilot use case 1.1 and 1.2.
Priority	Must
Source	Survey (Annex 2), Pilot 1.1 and 1.2

TR 19 Average response time (Pilot 1.1, 1.2)

ID	TR20
Short name	Average response time (Pilot 1.3)
Description	Normal document analysis (10 pages) should be processed within 30s for the Pilot use case 1.3.
Priority	Nice to have
Source	Survey (Annex 2), Pilot 1.3

TR 20 Average response time (Pilot 1.3)

ID	TR21
Short name	Average response time (Pilot 3)
Description	The system shall be able to process within one minute all use case requests related to scenarios 3a and 3b under Pilot 3.
Priority	Must
Source	Survey (Annex 2), Pilot 3

TR 21 Average response time (Pilot 3)

ID	TR22
Short name	Average response time (Pilot 2)
Description	The system shall be able to respond within 5-10 seconds to all requests related to Pilot 2 use cases
Priority	Must
Source	Survey (Annex 2), Pilot 2

TR 22 Average response time (Pilot 2)

3.2.9 Reliability

Reliability specifies the ability of a system to guarantee the provision of reliable data and services for a specified period of time. Ensuring a system that is reliable is paramount to the achievability of the target goals of the Lynx system; considering that the system is a data and service platform.

A diagrammatic illustration of data reliability in relation to time is presented in Figure 5. The diagram illustrates a labour law that has been amended but the Lynx platform still has the unamended data. Until the data is updated in the Lynx platform with the new amendment, the data remains unreliable.

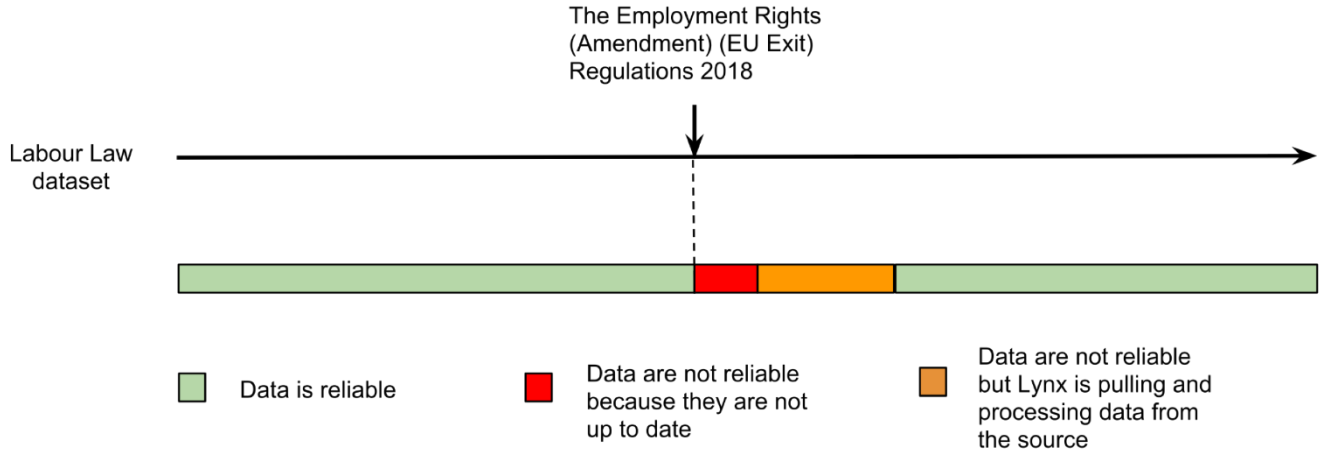

Figure 5 Data reliability in relation to time

ID	TR23
Short name	Data reliability
Description	The system shall ensure the reliability of data.
Priority	Must
Source	It is important to take into consideration that, the reliability of data can be dependable for a certain period of time and also under a specified context of use. This property is certainly true if we take into consideration documents that are subject to change.

TR 23 Data reliability

3.2.10 Recoverability

The ability of a system to recover data and processes from any unforeseen breakdown of any component or the entire system is referred to as recoverability. Below we specify and describe some recoverable requirements that the Lynx system shall fulfil.

ID	TR24
Short name	Data recovery (development environment)
Description	The system shall allow for the back-up of data manually in the development environment.
Priority	Must
Source	The primary working resource of the system is data, therefore, there is a need for manual recovery of these data during the development stages of the system.

TR 24 Data recovery (development environment)

ID	TR25
Short name	Data Recovery (Production Environment)
Description	The system shall allow for the back-up of data in production environment once every week.
Priority	Must
Source	Q & A (Annex 1)

TR 25 Data Recovery (Production Environment)

3.2.11 Capacity

The capacity of a system is a quality that describes the measure of the amount of data it can possibly store.

ID	TR26
Short name	Storage capacity (Data)
Description	The system shall be able to store not less than 100.000 documents with an average size of 1MB.
Priority	Must
Source	KPI (Key Performance Index) Lynx Proposal Document.

TR 26 Storage capacity (Data)

ID	TR27
Short name	Storage capacity (Metadata)
Description	The system shall be able to store metadata (triples and annotations) not less than three times the size of the data.
Priority	Must
Source	Q & A (Annex 1)

TR 27 Storage capacity (Metadata)

3.2.12 Fault tolerance

The property of a system that makes it perform well even in the presence of one or more components breakdown or failure is termed fault tolerance of the system.

ID	TR28
Short name	Fault-tolerant design
Description	The system shall be able to continue with its intended operation, possibly at a reduced level, rather than failing completely when some part (BB) of the system fails.
Priority	Nice to have
Source	The Lynx system is composed of more than 10 BBs of which some of them are not needed for the execution of some workflows; therefore, if one of these BBs fails, the system should maintain a limited functionality rather than failing completely.

TR 28 Fault-tolerant design

4 CONCLUSION

This deliverable presents the technical requirements of the Lynx system. It first explains what technical requirements are and describes the methodologies and models used for their gathering and specification.

The Lynx technical requirements can be classified according to several quality attributes that are relevant to the success of the system.

Table 2 presents key qualities required of the Lynx system, their corresponding related technical requirements and their specific identification under the headings 'Constraint description', 'Related to TR' and 'ID' respectively.

The technical requirements specified here, the pilots requirements specified in D4.1 and the functional ones, specified in D1.1, will be used in the design of the technical architecture (T1.3) and the development of the BBs of the platform in the WP3.

ID	Constraint description	Related to TR
QA1	The system shall be **modular**	TR1, TR2, TR3
QA2	The architecture of the system shall comply with the **REST architectural style**	TR4
QA3	The system shall be **flexible**	TR7, TR8, TR9
QA4	The system shall be **scalable**	TR10
QA5	The system shall support **access control of data and services**	TR12, TR13
QA6	The system shall provide **reliable data**	TR23

Table 2 The key qualities required of the Lynx system

5 REFERENCES

[1] R. B. Svensson, *Managing quality requirements in software product development, Ph.D. dissertation,* Department of Computer Science, Lund University, 2009.

[2] IEEE Computer Society (1990), IEEE Standard Glossary of Software Engineering Terminology, IEEE Standard, 1990.

[3] Wikipedia contributors, "Requirement," Wikipedia, The Free Encyclopedia, 23 August 2018. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Requirement&oldid=856155051.

[4] M. A and L. R, "A Survey of Non-Functional Requirements in Software Development Process," Technical Report, TR-LACL-2008-7, 2008.

[5] Reqexperts, "Is the Requirement Verifiable," [Online]. Available: https://reqexperts.com/2013/02/11/is-the-requirement-verifiable/. [Accessed 27 September 2018].

[6] Wikipedia, "Representational state transfer," [Online]. Available: https://en.wikipedia.org/wiki/Representational_state_transfer. [Accessed 05 November 2018].

[7] A. Rauf and M. AIGhafees, "Gap Analysis between State of Practice and State of Art Practices in Agile Software Development," in *IEEE 2015 Agile Conference*, Washington, 2015.

[8] Wikipedia, "WaterFall model," [Online]. Available: https://en.wikipedia.org/wiki/Waterfall_model. [Accessed 26 September 2018].

[9] R. B. Grady, "An economic release decision model: Insights into software project management,," in *Proceedings of the Applications of Software Measurement Conference*, Orange Park, Software Quality Engineering, 1999, pp. 227-239.

[10] d. P. L. J. Chung L., "On non-functional requirements in software engineering. In Conceptual modeling: Foundations and Applications (pp.363-379)," Springer , Berlin Heidelberg, 2009.

[11] D. Lindstrom, "Five ways to Destroy a Development Project," in *IEEE Software*, September 1993, pp. 55-58.

APPENDIX

Figure 6 presents a logical view of the Lynx architecture landscape that satisfies the architecture-relevant requirements gathered in this task. From the diagram, the API management BB provides an access interface to the other foundational BBs for which the task orchestrator is asynchronously connected to the peripheral BBs.

It is important to observe that there are two major critical points in this architecture: the task orchestrator and the document manager. The former will be responsible for the orchestration of the services associated to the peripheral BBs whereas the latter will allow the services associated with the LKG accessible BBs to access the documents and semantic annotation stored in the LKG as shown in Figure 6.

The design of a proper technical solution for the task orchestrator and the document manager will be carried out in WP3 and WP4.

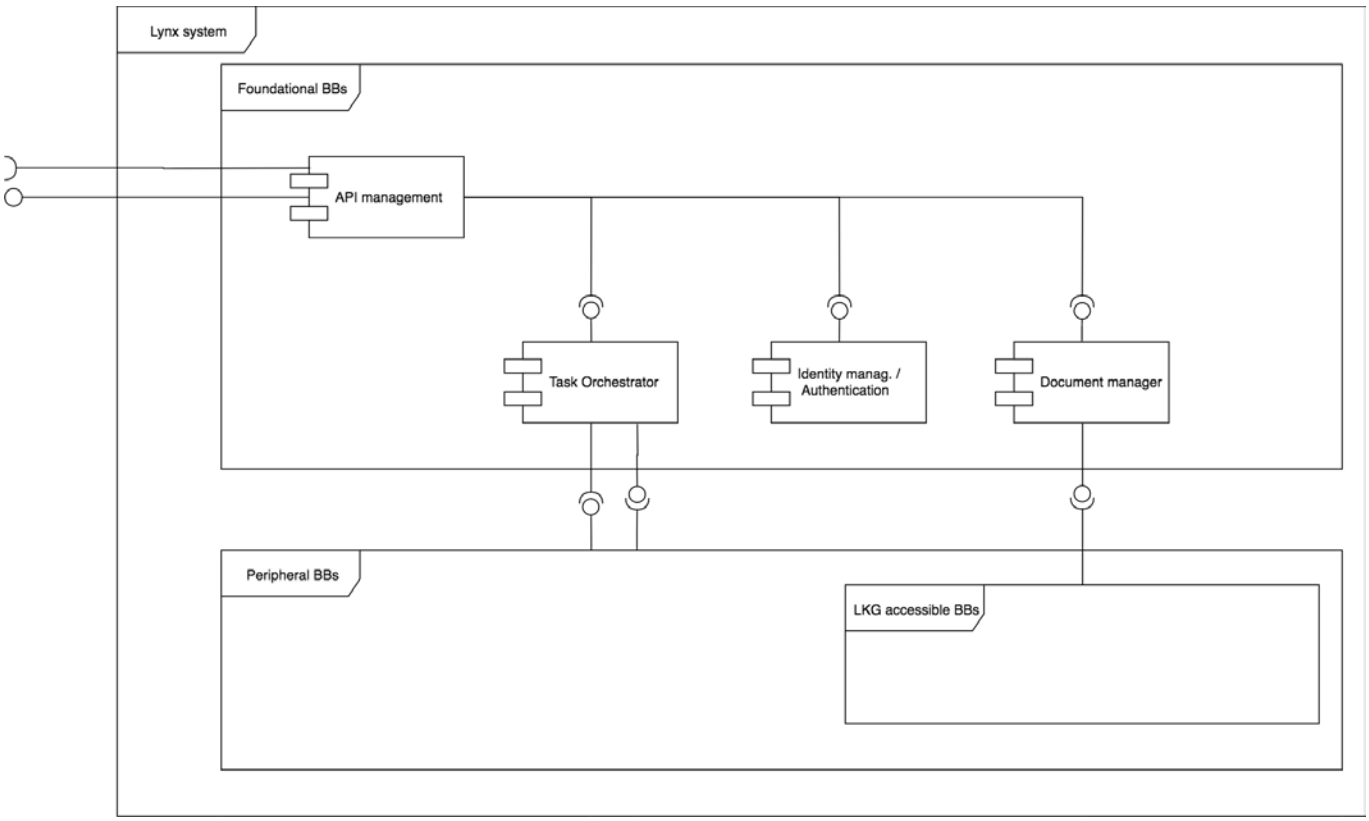

Figure 6 A representation of the logical view of the Lynx architecture using UML

ANNEX 1: Q & A

Questions	Answers	Topic
What are your requirements for the performance of the system?	The system shall be able to scale.	Performance
What type of application do we have (Web app, desktop application)? Do the application manages users of pilots?	The application that the pilot uses belongs to them. The goal of the architecture is just to give them an endpoint to connect to.	Interoperability
How do you expect the usability of the Lynx platform should be?	It should be relatively adaptable in terms of functionality in the future. It should have a simple interface to specify these future workflows easily.	Usability, Flexibility
How many numbers of request per hour should the system handle?	The system should be built to adjust the scaling.	Performance, Scalability
Back-up of the data, how frequent should it be?	The back-up is depended on the amount of data that is included in the database. A weekly period back-up can be a good starting point.	Recoverability
What will be the data growth rate?	The system will be able to scale to match the growth of the business.	Capacity, Scalability
What will be the workload growth rate?	The system will be able to scale to match the growth of the business.	Scalability
What is the size of metadata in relation to the size of the data?	The data size is upper limited by 3 times the size of the data.	Capacity
What should be the initial number of documents?	Estimated to 100000.	Capacity

ANNEX 2: SURVEY

Questions	Answers (Openlaws)
What are your requirements for the performance of the system? (In terms of peak/avg response time. It is also important to consider the input size when it is relevant)	**UC 1.1 and 1.2:** Search results must be returned within 500ms. **UC 1.3:** For document analysis there are not direct requirements, normal document (10 pages) should be processed within 30s (up to discussion).
What are your requirements for the availability of the system?	99.9% during business hours (8:00 – 19:00), Monday – Friday with maintenance outside business hours
Have you any other particular QoS requirements for the Lynx system?	n/a
How many requests per day can you estimate for an initial start?	**UC 1.1 and 1.2:** 1/sec **UC 1.3:** Bulk processing: 1000 in a batch

Questions	Answers (DNV GL)
What are your requirements for the performance of the system? (In terms of peak/avg response time. It is also important to consider the input size when it is relevant)	For the DNV GL use case, we believe that approximately one-minute maximum response time should be acceptable for the pilot situation
What are your requirements for the availability of the system?	During working days, it should be up. Maintenance/downtime preferably during European night time or weekends
Have you any other particular QoS requirements for the Lynx system?	n/a
How many requests per day can you estimate for an initial start?	10 – 15 requests/day

Questions	Answers (Cuatrecasas)
What are your requirements for the performance of the system? (In terms of peak/avg response time. It is also important to consider the input size when it is relevant)	We consider the time of 5-10 seconds to receive the answer (with the paragraphs and relations to articles) in the selected language, and the original plus the translated source document
What are your requirements for the availability of the system?	The platform should be available from 8:00-23:59 with a maintenance/downtime only at night times.
Have you any other particular QoS requirements for the Lynx system?	n/a
How many requests per day can you estimate for an initial start?	For an initial start, we expect 10-30 request/day on an average; but this request can increase for the use case 2.3.

ANNEX 3: MEETINGS OVERVIEW OF T1.2

Date	Title	Attendees	Relevant points related to T1.2
17.10.2018	WP1 Status call	Filippo Maganza (ALP) Anagbo Kennedy Junior (ALP) Elena Emontiel (UPM) Heidi Hobel (SWC) Ilan Kernerman (KDictionaries) Victor Rodriguez Doncel (UPM)	Questions regarding technical requirements
01.10.2018	WP1 Status call	Filippo Maganza (ALP) Anagbo Kennedy Junior (ALP) Elena Montiel (UPM) Victor Rodriguez Doncel (UPM) Christian Segeder (OLS) Ilan Kernerman (KDictionaries)	Discussion of deliverables
28.09.2018	Technical Requirements Discussion	Filippo Maganza (ALP) Anagbo Kennedy Junior (ALP) Elena Montiel (UPM) Victor Rodriguez Doncel (UPM) Victor Mireles-Chavez (SWC) Julian Moreno Schneider (DFKI)	Discussion of the technical requirements gathered
19.09.2018	On Functional and Technical Requirements	Matteo Zanioli (ALP) Victor Mireles-Chavez (SWC) Julian Moreno Schneider (DFKI) Victor Rodriguez Doncel (UPM) Filippo Maganza (ALP)	Discussion of the meaning of technical requirement