CHOReOS_Requirements for the CHOReOS IDRE (D5.1)

Amira Ben Hamida, Animesh Pathak, Andrea Polini, Daniel Cukier, Felipe Besson, Fabio Kon, Thiago Colucci, Darius Silingas, Guglielmo De Angelis, Giorgos Veranis, et al.

Large Scale Choreographies for the Future Internet

ICT IP Project

Deliverable D5.1

Requirements for the CHOReOS IDRE

http://www.choreos.eu














Project Number	:	FP7-257178
Project Title	:	CHOReOS Large Scale Choreographies for the Future Internet

Deliverable Number	:	D5.1
Title of Deliverable	:	Requirements for the CHOReOS IDRE
Nature of Deliverable	:	Report
Dissemination level	:	Confidential / Public
Licence	:	Creative Commons Attribution 3.0 License
Version	:	A
Contractual Delivery Date	:	M7
Contributing WP	:	WP5
Author(s)	:	A. BEN HAMIDA (EBM) A. PATHAK (INRIA) A. POLINI (UNICAM) D. CUKIER, F. BESSON, F. KON, T. COLUCCI (USP) D. SILINGAS (NME) G. DE ANGELIS (CNR) G. VERANIS (MLS) J. LOCKERBIE (CITY) M. AUTILI (UDA) P. CHATEL (THA) R. MAZZA (WIND) T. PARATHYRAS (VTRIP)
Reviewer(s)	:	A. BEN HAMIDA (EBM), A. ZARRAS (UOI), F. KON (USP), H. VINCENT (THA), P. CHATEL (THA), V. ISSARNY (INRIA)

Abstract

The goal of this document is to elucidate the requirements that the various actors involved with future Internet choreographies will have from the **CHOReOS Integrated Development and Runtime Environment (IDRE)**. Since the IDRE integrates the work performed in the work packages WP 2 - 4, the aforementioned requirements lead to the specification of requirements for WP 2 - 4, specifically those requirements which will govern how they will integrate with each other.

We base our work on the conceptual model of CHOReOS defined in D1.2, and first present the main concepts used while discussing the IDRE, including the actors and use cases. This is followed by an exhaustive list of requirements pertaining to each functionality that the IDRE will provide with regard to design, development and deployment of choreographies.

Keyword list

IDRE, Requirements, Service Choreography, Service Composition, Choreography Enactment, Runtime, Middleware, ESB, Future Internet, Large Scale, Design, Development, Governance, Monitoring

Document History

Version	Changes	Author(s)
0.1	Initial template and content for the document	P. CHATEL (THA)
0.2	Small corrections on chapter 2.	P. CHATEL (THA)
1.0 to 1.5	New requirements distribution and categories Changes and details added to Introduction. Changed system anatomy figure. Added requirement identifier nomenclature. Started the glossary. Added “Integration requirements” general category. Added “Development requirements” from EBM as integration requirements.	A. BEN HAMIDA (EBM), P. CHATEL (THA), M. AUTILI (UDA), F. KON (USP), T. COLUCCI (USP)
1.5.1 to 1.5.5	Merged choreography design and development requirements under section 3.1.1. Integrated new enactment requirements submitted by USP. New introduction text for section 1. Integrated new “Choreography development requirements” from USP. Updated “Choreography enactment requirements”.	P. CHATEL (THA), F. KON (USP), T. COLUCCI (USP), D. CUKIER (USP), F. BESSON (USP), A. PATHAK (INRIA), A. BEN HAMIDA (EBM)
1.5.6 to 1.5.10	Section 3.3 on Integration requirements removed, Choreography design requirements section added. New monitoring and enforcement requirements by MLS. New IDRE development requirements by MLS. New IDRE design requirements by BPI. Integration of Airport UC requirements (as an annex + references to business requirements added into Global Requirements descriptions)	A. BEN HAMIDA (EBM), P. CHATEL (THA), G. VERANIS (MLS), D. SILINGAS (BPI), G. DE ANGELIS (CNR), A. POLINI (UNICAM)
1.5.11 to 1.5.15	Proposing partner column added to requirement tables.	G. DE ANGELIS (CNR), A. POLINI (UNICAM), R. MAZZA (WIND), P. CHATEL (THA), A. PATHAK (INRIA), M.AUTILI (UDA), J. LOCKERBIE (CITY) , A. BEN HAMIDA (EBM)

Version	Changes	Author(s)
1.5.16 to 2.0	Multiple sections updated. Updated abstract. Updated section 3.3 on requirements by VTRIP and CNR-UNICAM. Updated IDRE requirements table (new text + deleted testing & debugging lines) Updated Requirements tables (according to Amira) Final updates in the document before QA review.	P. CHATEL (THA), R. BARTKEVICIUS (NME), A. PATHAK (INRIA), A. BEN HAMIDA (EBM), A. POLINI (UNICAM), A. PAPADAKIS – PESARESI (VTRIP), G. DE ANGELIS (CNR),
2.1	All previous comments removed by Hugues for draft submission Updated section 3 requirements according to document 2.1 “concerns”. Updated section 3 introduction. Integrated revisions from USP and VTRIP on section 3.	A. PATHAK (INRIA), A. BEN HAMIDA (EBM), P. CHATEL (THA), D. CUKIER, F. BESSON, F. KON, T. COLUCCI (USP), T. PARATHYRAS (VTRIP)
2.1.1	Integrated comments by Amira and Animesh	A. PATHAK (INRIA), A. BEN HAMIDA (EBM), P. CHATEL (THA)
2.2	Condensed document history. Removed distinction between FUN and NFUN from the requirement tables and the text. Updated Annex A. Started integration of Valerie’s comments. New Abstract. New introduction (section 1).	A. PATHAK (INRIA), A. BEN HAMIDA (EBM), P. CHATEL (THA)
2.3	Integration of USP and UDA final contributions. Updated introduction (section 1).	A. PATHAK (INRIA), A. BEN HAMIDA (EBM), P. CHATEL (THA), F. KON, T. COLUCCI (USP), M.AUTILI, D. DI RUSCIO (UDA)

Document Review

Review	Date	Ver.	Reviewers	Comments
Outline	28/02/11	1.0	P. CHATEL (THA)	N/A
Draft	20/04/11	2.0	A. BEN HAMIDA (EBM), P. CHATEL (THA)	N/A
QA	05/05/11	3.0	F. KON (USP), A. BEN HAMIDA (EBM), P. CHATEL (THA), H. VINCENT (THA), V. ISSARNY (INRIA)	N/A
PTC	06/05/11	A	PTC members	

Acronyms & abbreviations

Item	Description
API	Application Program Interface
BPEL	Business Process Execution Language (for Web Services)
BPMN	Business Process Modeling Notation
CA	Consortium Agreement
CSV	Comma Separated Value(s)
DL	Deliverable Leader
DOW	Description of Work
ESB	Enterprise Service Bus
FI	Future Internet
GPS	Global Positioning System
GUI	Graphical User Interface
IAC	Industrial Advisory Committee
IDRE	Integrated Development and Runtime Environment
IP	Internet Protocol
IT	Information Technology
JDK	Java Development Kit
JDT	Java Development Tools
LAN	Local Area Network
MDE	Model-Driven Engineering
MST	Management support team
OSS	Open Source Software
PL	Project Leader
PMC	Project Management Committee
PO	Project Officer
PTC	Project Technical Committee
SL	Scientific Leader
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol (XML protocol)
TCP	Transport Control Protocol
TDD	Test-Driven Development
TXT	Text (file)
ULS	Ultra-Large-Scale (Systems)
URI	Uniform Resource Identifier
WAR	Web Archive
WP	Work Package
WPL	Work Package Leader
WSDL	Web Services Description Language
XML	Extensible Markup Language

Glossary[1]

Choreography (in general): it is a multi-party and decentralized protocol that when put in place by the cooperating parties will permit to reach the overall choreography goal (i.e. its business purpose). In reaching the overall goal each party in general pursues a local goal, which often is related to its business. Each party in choreography can derive a precise protocol that it has to use in order to interact with the other partners with which it has a direct contact.

Choreography (of services): as a specialization of the previous definition in the SOA context, it is a composition of distributed (Web) Services in which there is no single point of control, i.e., control is distributed. Usually, choreographies are composed of heterogeneous collaborating services built by different groups and executed in multiple administrative domains..

Choreography Design: This process involves the specification of the choreography at a high level, using a dedicated modelling language, and based on the goal and requirement provided by the domain expert and consumers.

Choreography Development: This process produces (possibly with partial human intervention/inspection) the abstract proxies/adaptors that, supported by the CHOReOS service-oriented middleware, will enact choreographies at runtime.

Enactment (of a choreography): It is the process that makes a choreography come to existence at runtime, through the realization of its goals (global and locals). Enactment involves locating and instantiating services to participate in the composition. It also involves dynamically establishing the bindings among these services. This process includes deployment of proxies/adaptors and their execution. Typically, enactment of a complex choreography is performed incrementally at multiple times by multiple administrators or operators in multiple administrative domains.

Runtime: the meaning of this word depends on the context. Used as in "(...) *at* runtime", means that an event takes place during system execution. Used as in "*the* Runtime (...)", designates a system (e.g., middleware, tooling) that supports this execution.

Hereafter, the **CHOReOS Runtime** (with a capital 'R') is a collection of libraries, services, and middleware components that support choreography enactment. This comprises facilities for monitoring, V&V, fault-tolerance, and dynamic reconfiguration and adaptation of choreographies.

Service Contract: Is comprised of one or more published documents that express meta information about a service. The fundamental part of a service contract consists of the service description documents (as WSDL definition, XML schema definition, WS-Policy definition, and Service Level Agreement (SLA)) that express its technical interface. These form the technical service contract, which essentially establishes an API into the functionality, offered by the service.

[1] *Disclaimer: these definitions are non-normative and valid in the context of this document as our current acceptation of the following concepts. They are given as an aid to the reader for better understanding of the document.*

Table of Contents

1. Introduction: document rationale

Software integration is an important activity in the software engineering process. It relies on an integration plan, a set of collaborative development tools and well identified software development guidelines. This work package deals with the entire integration process of all software components produced in WP2, WP3 and WP4 into one platform, called the **CHOReOS Integrated Development and Runtime Environment** (IDRE). This process is based on four main tasks. The first fundamental task is to identify major integration requirements that need to be satisfied. Second, a design activity is undertaken on the basis of the requirements. Third, the valid execution of the IDRE needs to be assessed with implemented test beds. The fourth task of providing the collaborative development tools and platforms and disseminating the CHOReOS results will be carried out in parallel with other tasks.

This deliverable (D5.1) is the result of the first task mentioned above, and provides integration requirements. The functionalities introduced in Section 4 of D1.2 -- Conceptual model for choreography-based Future Internet -- are considered as a starting point. Also, since the CHOReOS IDRE is geared towards to Ultra Large Scale systems, we need to cope with Future Internet Challenges such as Scalability, Heterogeneity, Mobility, Awareness and Adaptabiliy (as defined in Section 2.1.2 of D1.2; and take them into account in the integration requirements.

The rest of this document is organized as follows:

- **Chapter 2** deals with the definition of an overall view of the CHOReOS IDRE. It recalls and classifies the functionalities discussed in D1.2. This chapter separates the IDRE into the Development and Runtime environments and identifies the specific functionalities related to them. We also identify actors and use cases, and classify the latter according to related requirement types and the CHOReOS concerns.
- **Chapter 3** provides a threefold approach to integration requirements. First in Section 3.1 the overall requirements from the IDRE (its main features) and their priority levels are introduced. Then, in Section 3.2, requirements are listed for each feature -- choreography design, development and enactment. Section 3.3 deals with the IDRE Implementation requirements, categorised according to whether they are related to the design, development or deployment feature of the IDRE.
- Finally, **Chapter 4** (an annex) presents an early work identifying the requirements in the “Passenger Friendly Airport” case study. An overview of the case study is presented and a list of requirements is elucidated.

2. CHOReOS IDRE overview

2.1. Introduction

In CHOReOS, WPs 2, 3, and 4 develop techniques for development, enactment, and monitoring of choreographies for the ultra large-scale future Internet. The entity that integrates the distinct but inter-related components resulting from this research is the *CHOReOS Integrated Development and Runtime Environment (IDRE)*. This section provides an overview of the CHOReOS IDRE, and provides description of IDRE actors and use cases based on consolidating the information introduced in the CHOReOS conceptual model in D1.2. The use cases and actor definitions are then used in Section 3 to elicit the initial set of IDRE requirements.

CHOReOS IDRE high-level overview

The high-level representation of the overall structure and functionalities of the CHOReOS IDRE is shown in Figure 1. The IDRE, as its name suggests, consists primarily of the **Development** and **Runtime** environments. The figure maps the concerns discussed in the CHOReOS conceptual model (see D1.2) onto these two components of the IDRE. It also associates the key actors in CHOReOS (further defined in Section 2.2) with the components.

From a functionality point of view, the IDRE will be used to **design**, **develop** and **enact** choreographies, as well as **monitor**, **test** and **debug** them. From a technical standpoint, the final purpose of the IDRE is to produce an enacted choreography, from a *context* (which includes expert requirements, choreography specification, available related models, etc.), taking into account the *execution environment* (which includes available services).

The enactment of a choreography will be supported by the CHOReOS Middleware defined in WP3. Also, the CHOReOS Runtime should also have support for *awareness* and *adaptability*; this means that if there are any modifications to the *execution environment*, the system should be able to adapt to them to ensure ongoing enactment of the choreography while still satisfying its global requirements. Another aspect is that a domain expert should be able to make modifications to the initial requirements during runtime; thus the development and runtime environment work in synergy.

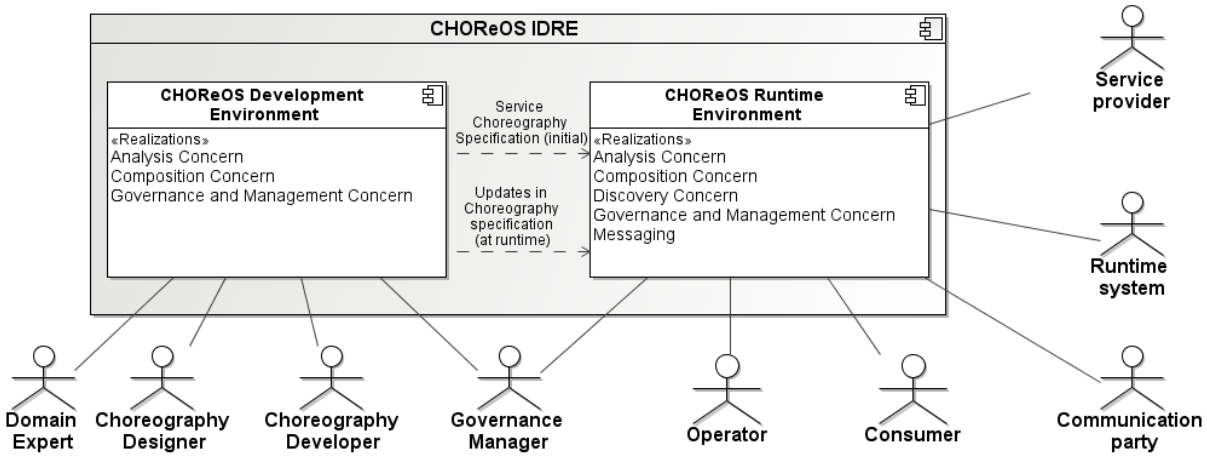

Figure 1 - CHOReOS IDRE Overview

2.2. Actors and use cases

Defining actors and use cases for the CHOReOS IDRE enables the identification of the main functionalities and stakeholders of the system. This section describes the actors involved in the CHOReOS use cases, **as defined in D1.2 various use case diagrams**, and discussed below. We recall their definitions from the CHOReOS conceptual model where necessary. Their associations with the IDRE components can be seen in Figure 1.

First, related to CHOReOS Development Environment, we identify the following actors, as was elicited in document D1.2:

- **Domain expert**. A domain expert has the responsibility of defining, among other things, global choreography requirements, in order to produce goal-based models of services and service-based applications. It includes the definition of high-level business needs where both the domain expert and the (service) consumer (see definition below) can specify business high-level requirements. These can be represented using a structured natural language, as will be investigated in WP2.
- **Choreography designer**. This actor has the responsibility of defining and/or finalizing the specification of the choreography, using the BPMN 2.0 modelling language and based on the refinement of the previously obtained domain expert specification by means of a top-down transformational process. The choreography designer is also responsible for the definition of the governance and analysis parameters that will be considered later by the CHOReOS work package 4 “Governance and V&V”.
- **Choreography developer.** The developer has the responsibility of writing source-code for Web services, requesting bytecode generation, creating configuration files, writing and executing tests, and debugging services.
- **Governance manager.** This actor manages the governance of services and their choreographies, Service Level Agreements (SLAs) and their Life Cycles, policies for services, choreographies and SLAs. This actor also interacts with the CHOReOS Runtime system.

The following actors in CHOReOS interact primarily with the CHOReOS Runtime Environment:

- **Operator**. An operator is an actor who has the responsibility of launching the enactment of a choreography, following the design phase. An operator may also monitor and enforce the system at runtime. The operator can also deploy, enable or disable a specific choreography. Finally, at runtime the operator is able to update the specification requirements that need to be taken into consideration.
- **Consumer.** A choreography consumer can be a person or an application or a service that calls some business functionality of a given choreography. Calling choreography can trigger choreography or an orchestration process.
- **Service provider.** A service provider is responsible for publishing services and SLA templates including the services behaviour according to runtime parameters. This actor relates to the discovery concern of D1.2 and also activates testing activities to test specific services.
- **Runtime system.** In this actor, we combine the actors “Broker”, “Message Broker”, and “Run-time system” from the CHOReOS conceptual model of D1.2. This actor then is involved in service discovery and publishing, messaging between services, as well as executing a testing session to evaluate a service.
- **Communication party.** The Communication party is an actor which performs exchange of messages with other Communication party.

The table below provides a list of use cases organized by concern. For each use case we indicate related actors and **requirements types**. These requirements types are the base for

the elicitation of the initial set of requirements in Section 3; they follow the naming convention described hereafter:

Choreography-related requirements:

- CHOR-DES: stands for choreography design requirements.
- CHOR-DEV: stands for choreography development requirements.
- CHOR-ENA: stands for choreography enactment requirements.
- CHOR-MON: stands for choreography monitoring requirements.
- CHOR-GOV: stands for choreography governance requirements.

IDRE realization related requirements:

- IDRE-DEV: stands for requirements for the development of the IDRE
- IDRE-DES: stands for requirements for the design of the IDRE

Concern	Use Case (from D1.2)	Actors Involved	Requirements section
Composition			
C1	*Goal and Requirement Specification*	*Domain Expert, Consumer*	*CHOR-DES,*
C2	*Creation of Service Choreography*	*Choreography Designer, Choreography Developer*	*CHOR-DEV, CHOR-DES*
C3	*Specification of Service Choreography*	*Choreography Designer*	*CHOR-DES*
C4	*Choreography Implementation*	*Choreography Developer*	*CHOR-DEV, CHOR-DES*
C5	*Synthesis of Service Choreography*	*Choreography Developer*	*CHOR-DEV, CHOR-DES*
C6	*Creation of Service Composition*	*Choreography Designer, Choreography Developer*	*CHOR-DES*
C7	*Enactment and Execution of Service Choreography*	*Consumer, Operator*	*CHOR-ENA*
C8	*Execution of Service Composition*	*Consumer, Operator*	*CHOR-ENA*
Presentation	None		
Analysis			
A1	*Test Activation*	*Runtime System, Service Provider*	*CHOR-GOV, CHOR-MON*
A2	*Test Case Selection and Derivation*	*Runtime System, Service Provider*	*CHOR-GOV, CHOR-MON*
A3	*Test Execution*	*Runtime System, Service Provider*	*CHOR-GOV, CHOR-MON*
Discovery			
D1	*Publication of Service Offering*	*Runtime System, Provider*	*CHOR-ENA*
D2	*Abstraction Recovery*	*Runtime System, Provider*	*CHOR-ENA*
D3	*Discovery of Service*	*Runtime System, Consumer*	*CHOR-ENA, CHOR-GOV*
D4	*Browsing Service Abstractions*	*Runtime System, Consumer*	*CHOR-ENA, CHOR-GOV*
D5	*Searching Service Abstractions*	*Runtime System,*	*CHOR-ENA, CHOR-*

Concern	Use Case (from D1.2)	Actors Involved	Requirements section
		Consumer	*GOV*
D6	*Publication of Service Demand*	*Consumer*	*CHOR-ENA, CHOR-GOV*
Messaging			
M1	*Exchange of Messages*	*Communication Party, Runtime System*	*IDRE-DEV*
M2	*Validation and Analysis of Messages*	*Communication Party, Runtime System*	*IDRE-DEV*
M3	*Routing and Forwarding of Messages*	*Communication Party, Runtime System*	*IDRE-DEV*
M4	*Adaptation and enhancement of Message*	*Communication Party, Runtime System*	*IDRE-DEV*
M5	*Adaptation of Data Format, transport protocol and interaction model*	*Communication Party, Runtime System*	*IDRE-DEV*
Service	None		
Resource	None		
Governance and Management			
G1	*Choreography/Service/SLA Governance*	*Governance Manager*	*CHOR-GOV*
G2	*Choreography/Service/SLA Governance*	*Consumer, Service Provider*	*CHOR-GOV*
G3	*Registry*	*Consumer*	*CHOR-GOV*
G4	*Discovery*	*Consumer, Governance Manager*	*CHOR-GOV*
G5	*Publication*	*Consumer*	*CHOR-GOV*
G6	*Lookup*	*Governance Manager*	*CHOR-GOV*
G7	*Enforcement*	*Governance Manager*	*CHOR-GOV, CHOR-MON*
G8	*SLA Management*	*Governance Manager*	*CHOR-GOV*
G9	*SLA Negotiation*	*Consumer, Service Provider*	*CHOR-GOV*
G10	*Policy Management*	*Governance Manager*	*CHOR-GOV*
G11	*Rules Management*	*Governance Manager*	*CHOR-GOV*
G12	*Runtime Management*	*Governance Manager*	*CHOR-GOV, CHOR-MON, CHOR-ENA*
G13	*Choreography/Service/SLA Monitoring*	*Governance Manager*	*CHOR-MON, CHOR-GOV*
G14	*Test Activation*	*Governance Manager*	*CHOR-MON, CHOR-GOV*
Security	None		

3. IDRE global requirements

The main features of the IDRE are detailed in the form of requirements in Section 3.1. Following that, the requirements pertaining to these features are described in Section 3.2 and its sub-sections (one for each main feature of the IDRE). These requirements will serve as input to software components designed and developed in:

- WP2 - Dynamic development of adaptable, QoS-aware ULS choreographies,
- WP3 - Service-Oriented Middleware for the Future Internet
- WP4 - Governance and V&V support for choreographies for the Future Internet

For each requirement we also point out the related concern(s), the involved CHOReOS work packages, and finally the proposing partners.

General requirements also encompass the design and implementation **of the IDRE itself** by the CHOReOS partners. These requirements, described in section 3.3, are oriented towards easing the integration of the components to be developed within WP2-3-4 during the lifetime of the CHOReOS project. Finally, in this section, we make reference, to already established early business requirements from the “Passenger Friendly Airport” use case from WP6 (see Annexe A – Airport Case Study requirements).

Terminology Reminder for requirements

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (http://tools.ietf.org/html/rfc2119):

- MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase "SHALL NOT", means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.
- Requirements priority levels:
  - VERY LOW: Do if there is enough available time before the end of the project
  - LOW: Nice to be delivered by the end of the project but not required
  - MEDIUM: Important to be delivered by the end of the project
  - HIGH: Essential to be delivered by the end of the project
  - VERY HIGH: Fundamental pieces that must be done first so we can progress

3.1. IDRE main features

Goal: describe the main features of the IDRE as requirements on the IDRE, while other tables in following section 3.2 describe all the specific requirements pertaining to these features.

ID	Name	Priority	Description	WP
IDRE-FEA-001	Choreography design	VERY HIGH	The IDRE SHALL support the design of ultra large scale (ULS) choreographies. This includes the fact of having the needed specification language elements expressing choreographies.	WP2
IDRE-FEA-002	Choreography development	VERY HIGH	The IDRE SHALL support the development and implementation of ULS choreographies.	WP2
IDRE-FEA-003	Choreography enactment	VERY HIGH	The IDRE SHALL support the enactment of choreographies on top of ultra large scale systems. The CHOReOS middleware needs to cope with Future Internet environments and hence targets the enactment of choreographies and services on top of scalable, hetergeneous and highly mobile infrastructures as distributed service bus, smart devices and cloud platforms.	WP2-3
IDRE-FEA-004	Choreography governance and monitoring	HIGH	The IDRE SHALL support the governance, V&V and monitoring of choreographies for the Future Internet.	WP3-4

3.2. Feature related requirements

Goal: describe the main requirements associated with the functionalities of the IDRE

The functionalities that the IDRE shall provide are related to the activities of the WP2 development process shown in Figure 2. Such a process is in an early stage and will be refined and enhanced by WP2 throughout the duration of the project. However, by exploiting model-driven techniques, the final goal is to devise a systematic development of choreographies from their design to their execution. The main activities of the process in Figure 2 are represented with ellipses, whereas the artifacts and the models, which are produced and that the IDRE shall be able to manage, are represented with boxes.

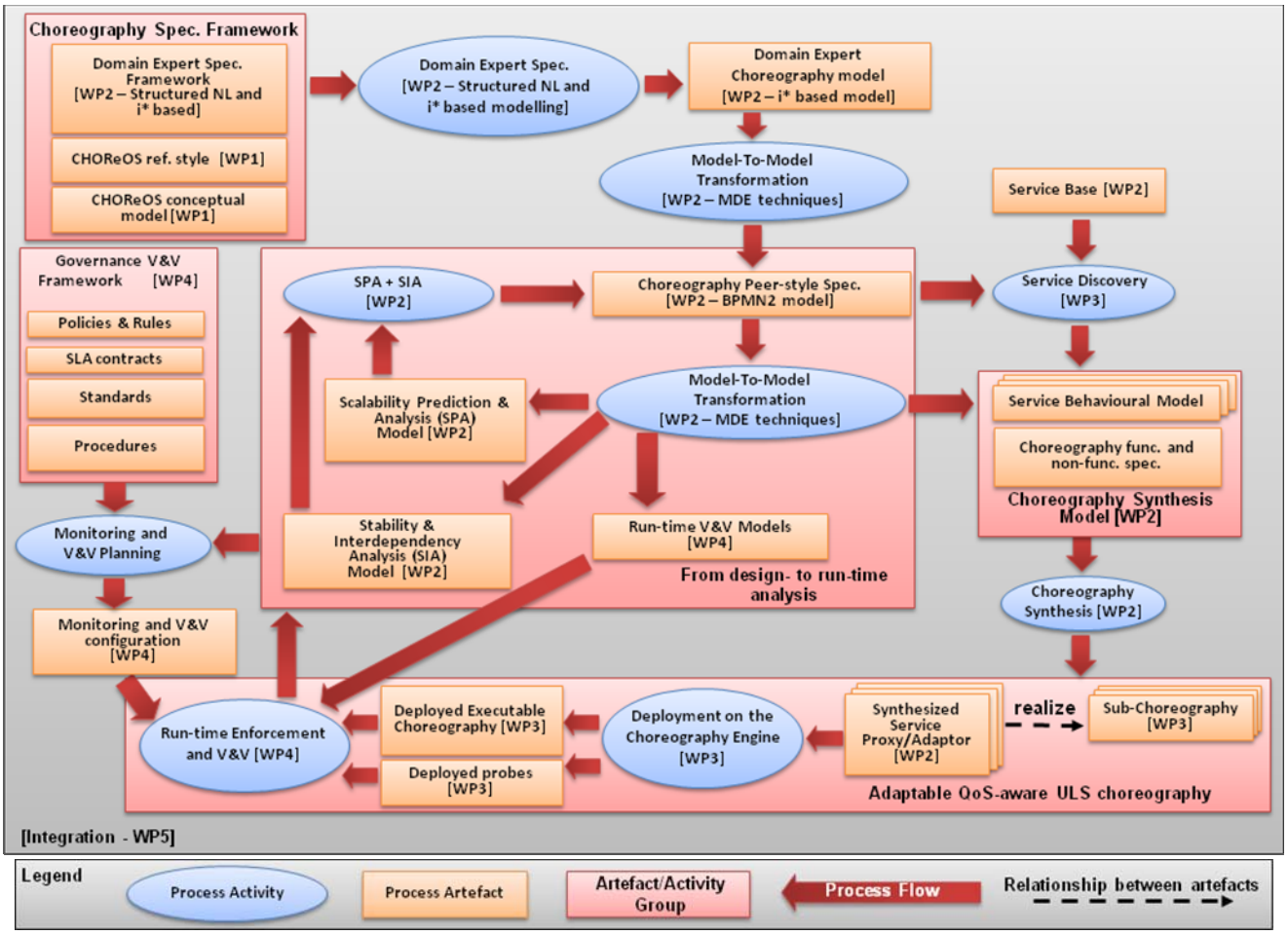

Figure 2 - WP2 development process

The IDRE shall support two orthogonal transformational approaches: (i) a top-down transformation process and (ii) a cross-cutting transformation process. The former will serve to refine a *domain-expert requirements* specification into *analysis-technique-specific models*. The latter will serve to integrate the different *modeling/reasoning technologies* by passing from a technique-specific model to a different one. This will bridge the gap between the various models that have to be used for choreography specification, analysis, validation, and synthesis. Moreover, the IDRE shall be able to manage a distributed *service base,* which organizes available services into functional and non functional views. Such a service base is used during the *choreography synthesis* activity. In fact, starting from the *choreography specification,* the required services are discovered in the service base with respect to functional and non-functional requirements. The synthesis produces (possibly with partial human intervention/inspection) the models of proxies/adaptors that, accounting for service functional and non-functional abstractions, distributedly support the enactment of choreographies. The proxy/adaptor models can then be deployed, e.g., as a set of wrappers, each of them local to each service involved in the choreography.

The process in Figure 2 has to be considered perpetual in the sense that it *extends to run-time* and does support domain expert developers and (end-)users in all the phases of the choreography life cycle. Consequently, the IDRE has to support *complex analysis and validation steps at run-time* when all the necessary pieces of information are available. In other words, some activities of the process may be executed at run-time, as specific internal capabilities of the choreography engineered through the process itself.

In the next sections the functionalities that the IDRE shall provide with respect to the previous discussion, are presented in detail.

3.2.1. Choreography design requirements

Goal: describe the requirements for choreography design, which shall guide the development of the choreography design tool integrated in the IDRE must support these requirements.

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-DES-001	Support Ultra Large Scale (ULS) choreography design	VERY HIGH	THA	IDRE user SHALL be able to design choreographies or modify existing ones using choreography design tool and exploiting repository of available artifacts. The IDRE SHALL support this functionality by providing the needed specifications and abstractions expressing choreographies. Abstractions for dealing with millions of services need to be provided and exploited for designing ULS Choreographies. See See AIRPORT-FUN-002, AIRPORT-DES-004 AIRPORT-DES-004, AIRPORT- FUN-010.	WP2	Composition
CHOR-DES-002	Describe the non-functional information of a choreography	HIGH	EBM	In order to achieve adaptability and awareness, services and users non-functional informations need to be taken into account. THE IDRE MUST provide the ability of describing the choreographies by adding non functional information.	WP2	Composition
CHOR-DES-003	Import choreography specification	MEDIUM	EBM	The IDRE SHOULD be able of importing and reading choreography specification files in the IDRE choreography design tool.	WP2	Composition
CHOR-DES-004	Provide discovery and governance tools at design time	MEDIUM	EBM	The IDRE SHOULD provide the user with newly discovered services and existing choreography artifcats at the design time of a choreography. This functionality enables the IDRE awareness and adaptability and can be achieved by connecting the IDRE design tool to the service discovery tool (from WP3). Moreover the IDRE SHOULD provide governance abilities at design time. This is achieved by connecting the IDRE design tool to the governance tool (from WP4).	WP2-4	Discovery, Governance and Management
CHOR-DES-005	Collaborative choreography Design	LOW	EBM	In a multi partner environment, it may be interesting to collaborate for designing a common choreography. The IDRE MAY provide such functionality allowing two or more partners to work on the same choreography	WP2	Composition

ID	Name	Priority	Partner	Description	WP	Concern(s)
				diagram. See AIRPORT-DES-003.		
CHOR-DES-006	Transform between collaboration-like and choreography-like exchanges	LOW	EBM	Transformations between collaboration-like diagrams and choreography-like diagrams MAY be interesting to be supported by the IDRE.	WP2	Composition
CHOR-DES-007	Identify error messages and status codification	HIGH	CNR-UNICAM	During the enactment of a choreography, error messages, and the status codification of response messages of any interaction MUST be identifiable as such, e.g. through a special field in the message header.	WP1-5	Analysis, Composition
CHOR-DES-008	Refine participants and actions identification	HIGH	CNR-UNICAM	During the enactment of a choreography, the participants and the actions involved in every message exchange MUST be identifiable as such, e.g. through a special field in the message header. This needs to be defined at design time	WP1-5	Composition

3.2.2. Choreography development requirements

Goal: describe the IDRE requirements for choreography development, including **writing** source-code for the mediators and other non-existing web services, **testing**, and **debugging** choreographies. In addition, a framework for applying automated unit, integration, and acceptance testing on running choreographies will support the automated testing of choreographies, enabling test-driven development (TDD) of choreographies.

CHOReOS will enable the development of choreographies in two different ways:

1. Programmer-based development of mediators and web services code: Not necessarily all web services needed by the choreography will be available previously. Thus a developer must implement them and also create manually the mediators used to adapt the already existing services, so that the choreography can be enacted. This way of enacting allows quick choreography prototyping and relies only on software developers. This it can be performed without requiring any high-level specification language for the development.
2. Automatic code generation of mediators using MDE and synthesis mechanisms: All choreography services already exist before the enactment and they are composed into a choreography through the use of automatically generated mediators that adapt those services to meet the high-level specification given by the Choreography Designer based on input from Domain Experts and the User. This way of enacting depends on the existence and maturity of the MDE and synthesis mechanisms; thus it will be implemented mostly during the second half of the project.

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-DEV-001	Use abstract choreography models	VERY HIGH	USP	The elements of a choreography (partners, services, messages exchanged, etc.) MUST be represented by an object in an object-oriented language (called abstract choreography). Through these objects, the developer will interact easily with the choreography elements to write automated tests.	WP4	Composition
CHOR-DEV-002	Generate automatically web service client (stubs)	VERY HIGH	USP	Given a web service URI, the framework SHALL build a client (stub) for this service automatically. Through this client, the tester will be able to invoke all web service functionalities.	WP4	Composition
CHOR-DEV-003	Provide Message interceptors	VERY HIGH	USP	SHALL intercept, and then, collect the service operation name and the content of the messages exchanged among the services the developer wishes to monitor.	WP4	Messaging Management
CHOR-DEV-004	Provide unit, integration, and Acceptance tests	HIGH	USP	MUST provide support for writing, managing, and execution of unit, integration, and acceptance tests. SHALL generate reports of the execution of tests, showing which passed, failed, and	WP4	Analysis

ID	Name	Priority	Partner	Description	WP	Concern(s)
				crashed.		
CHOR-DEV-005	Mock Services	MEDIUM	USP	MUST enable the creation of a service double (e.g., a mock). Through this object, all web service functionalities can be mocked. This will be useful at development time to test the choreographies when not all real services are available or when a specific scenario needs to be simulated.	WP4	Composition
CHOR-DEV-006	Generate a mocked service for part of a choreography	MEDIUM	USP	MUST generate a mocked service for part of a choreography. This service will simulate the process performed in that part of the choreography. This will be particularly important for integration tests.	WP4	Composition
CHOR-DEV-007	Interact with version control client	MEDIUM	USP	MUST provide a client for interacting with version control systems such as svn, git, and bazaar.	WP5	IDRE implementation
CHOR-DEV-008	Provide GUI – Graphical User Interface for collecting technical requirements	MEDIUM	USP	All the technical requirements of this section will be available through a GUI with views, buttons, graphics, editors, lists, etc.	WP2	Presentation, Composition
CHOR-DEV-009	Manage Web Servers	LOW	USP	Starts and stops web servers through the IDRE and deploy the developed web services.	WP3-5	IDRE Implementation,runtime
CHOR-DEV-010	Create a wizard for tests	LOW	USP	A wizard will aid the developer in writing different kinds of tests by generating skeletons of parts of the tests and guiding the developer requesting the necessary information to create the tests.	WP4	Presentation, Analysis
CHOR-DEV-011	Export Web Service as WAR file	Low	USP	SHOULD export a web service project as a WAR file for later use on a choreography.	WP5	IDRE Implementation
CHOR-DEV-012	Support Ultra Large Scale (ULS) choreography development	VERY HIGH	THA	The development of choreographies of Ultra Large Scale dimension MUST be supported. See AIRPORT-FUN-002.	WP2	Composition
CHOR-DEV-013	Generation of proxies/adaptors models	VERY HIGH	UDA	Starting from the choreography specification, and by taking into account service functional abstractions, the models of the required proxies/adaptors SHALL be generated.	WP2	Composition

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-DEV-014	Generation of proxies/adaptors code	VERY HIGH	UDA	To force the collaboration of the discovered services and to guarantee the specified choreography, the models of the generated proxies/adaptors SHALL be automatically transformed into actual software artifacts	WP2	Composition
CHOR-DEV-015	Model management	VERY HIGH	UDA	The IDRE SHALL provide an infrastructure to manage the different models which are produced/analyzed/transformed during the overall choreography life-cycle (Figure 1)	WP2-4	Composition, Analysis

3.2.3. Choreography enactment requirements

Goal: describe the requirements for choreography enactment. These requirements should be closely related (but at a higher abstraction level) to the ones that will be defined by WP3, since enactment is enabled and supported by the CHOReOS middleware.

These features will enable the instantiation of choreographies at runtime. The Middleware will provide all the required software infrastructure to instantiate the required services such as mediators, proxies and other choreography-aware services. It will also provide the bindings from these services to existing web services that will be composed in the choreography.

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-ENA-001	Deploy a Web service	VERY HIGH	THA+USP	Given a web service executable code, e.g., jar file or Petals service assembly package, deploys it on the network using the CHOReOS middleware.	WP3	Composition
CHOR-ENA-002	Enact a Choreography	VERY HIGH	THA+USP	The IDRE MUST provides capabilities for choreography enactment.	WP3	Composition
CHOR-ENA-003	Synthesize intermediate models	HIGH	THA+USP+UDA	Given multiple seminal specifications, including domain expert requirements and peer-style choreography description, the IDRE SHALL synthesize intermediate models required for choreography enactment.	WP3	Composition
CHOR-ENA-004	Create web service executable code from a BPMN2 or BPEL description	HIGH	THA+USP	Given a web service description, defined in a BPMN2 Process specification or in a BPEL specification, creates executable code (probably bytecode and/or Petals service assembly package) that can be executed at runtime.	WP3	Composition

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-ENA-005	Instantiate Choreographies in a Cloud Infrastructure	HIGH	THA+USP	The IDRE MUST provide capabilities for Instantiating the new CHOReOS middleware nodes in virtual machines of a given cloud infrastructure, e.g., Amazon EC2, Open Cirrus, or a private Cloud using Open Nebula.	WP3	Composition
CHOR-ENA-006	Shutdown an Enacted choreography	MEDIUM	THA+USP	The IDRE SHALL provide capabilities for Shutting down an enacted choreography (the part of a choreography automatically generated and also the part deployed) by stopping to receive messages, undeploying the web services, and releasing the resources reserved for this part of the choreography. See AIRPORT-FUN-006, AIRPORT- FUN-007, AIRPORT- FUN-008.	WP3	Composition
CHOR-ENA-007	Pause an Enacted choreography	LOW	THA+USP	The IDRE MAY provide capabilities for suspending an enacted choreography by stopping to process the received messages. See AIRPORT-FUN-006, AIRPORT- FUN-007, AIRPORT- FUN-008.	WP3	Composition
CHOR-ENA-008	Continue an enacted choreography	LOW	THA+USP	The IDRE MAY offer capabilities to continue a previously paused enacted choreography. See AIRPORT-FUN-006, AIRPORT- FUN-007, AIRPORT- FUN-008.	WP3	Composition
CHOR-ENA-009	Discover dynamically services	VERY HIGH	THA+USP	The IDRE MUST support the dynamic discovery of services during the choreography enactment and also at runtime. See AIRPORT-FUN-001, AIRPORT-RUN-001.	WP3	Discovery
CHOR-ENA-010	Support Ultra Large Scale (ULS) choreography enactment	VERY HIGH	THA+USP	The IDRE MUST support the enactment of choreographies in an Ultra Large Scale context. See AIRPORT-FUN-002, AIRPORT-RUN-004.	WP3	Composition

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-ENA-011	Control dynamic access to services	LOW	THA+USP	According to priority constraints at the choreography level, services MAY have their access control (i.e. security) relaxed to authorize temporary consumption. See AIRPORT- FUN-003.	WP4	Analysis
CHOR-ENA-012	Compile a choreography	HIGH	THA+USP	Given a BPMN2 Choreography description (message flow and description of roles) generates an in-memory representation for this choreography, so this data structure can be reused throughout the IDRE without needing to parse XML files every time. If needed, this object can be serialized in a persistent storage.	WP3	Composition
CHOR-ENA-FUN-013	Reconfigure dynamically	VERY HIGH	USP+MLS	In order to enable adaptability, the IDRE SHALL be able to replace parts of a running choreography without shutting it down. This can be used for example, to cope with failures, to optimize performance, to upgrade components, etc. See AIRPORT-RUN-001.	WP3	Management

3.2.4. Choreography governance and monitoring requirements

Goal: Describe the requirements for choreography governance and monitoring at runtime.

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-MON-FUN-001	Monitor the system at runtime	VERY HIGH	USP	At runtime, the IDRE SHALL collect information on running choreographies and offer ways (visual feedback, API, …) to take actions according to the monitored data. The monitoring infrastructure needs to be able to deal with ULS choreographies and services. See AIRPORT- FUN-009, AIRPORT-RUN-003.	WP4	Management
CHOR-MON-FUN-002	Perpetual Testing	HIGH	USP	The IDRE MUST support perpetual (i.e. event-driven, periodical) and automated V&V activities.	WP4	Management

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-MON-FUN-003	Monitor message interchanges	VERY HIGH	USP	Monitor the flow of messages in an enacted choreography at runtime, including content, timestamp, sender, and recipient of exchanged messages. The monitoring infrastructure needs to deal with a scalable number of exchanged messages.	WP3	Management
CHOR-MON-FUN-005	Monitor resource availability	VERY HIGH	USP	The IDRE MUST provide abilities for monitoring nodes availability and associated resources to an enacted choreography, such as CPU usage, RAM usage, disk space. Choreos Middleware needs to run on top of a highly scalable platform (DSB, IoST, Cloud/Grid) and hence needs to manage a high number of heterogeneous and distributed resources.	WP3-WP5	Management
CHOR-MON-FUN-006	Handle exceptions	VERY HIGH	USP	The IDRE MUST provide mechanisms for exception handling and compensation of transactions for the enforcement of enacted choreographies at runtime.	WP3	Management
CHOR-MON-FUN-007	Handle Faults	VERY HIGH	USP	Detect failures in the enactment of a choreography and automatically reconfigure it to fix the faults detected as failures.	WP3-4	Analysis Management
CHOR-MON-FUN-008	Monitor time availability	MEDIUM	MLS	Monitoring time on an enacted choreography will be effective for load balancing issues in cooperation with monitoring resource availability.	WP5	Management
CHOR-MON-009	Handle Verification and Validation	VERY HIGH	MLS+CNR UNICAM	Provide mechanisms for verification and validation handling of a running ULS choreography.	WP4-5	Analysis
CHOR-MON-011	Handle metadata from services or choreographies	MEDIUM	CNR-UNICAM	The IDRE SHOULD be able to collect and convey service description and its metadata [including perhaps a abstract model] from the service and/or choreography developer to the monitoring subsystem [and any other subsystems which need it]	WP3-5	Composition
CHOR-MON-012	Provide hooks to the service registration process	LOW	CNR-UNICAM	The CHOReOS middleware, an integral part of the IDRE, SHOULD provide hooks to the service registration process so, for example, validation processes can be added in the service registration phase	WP2-5	Discovery Analysis

ID	Name	Priority	Partner	Description	WP	Concern(s)
CHOR-MON-013	Monitor scalability	HIGH	CNR-UNICAM	The monitoring infrastructure of the IDRE MUST be scalable and distributed	WP3-4	Management
CHOR-GOV-001	Governance Scalabilty	HIGH	EBM	The governance infrastructure MUST be scalable and distributed	WP4	Mangement
CHOR-GOV-002	Govern Services, Choreographies and SLAs	VERY HIGH	EBM+CNR-UNICAM	The IDRE MUST provide a governance infrastructure for a very large number of services, choreographies and Services Level Agreements. Indeed, this infrastructure MUST be able to provide governance abilities (such as registries, rules, etc.) in a highly distributed and scalable environment dealing with large numbers of dynamically discovered services.	WP4	Management
CHOR-GOV-003	Manage Governance Policy	HIGH	EBM	The IDRE MUST provide management capabilities for creating, updating, removing governance policies. This feature may also include Governance rules management as creation, update, removal, etc.	WP4	Management
CHOR-GOV-004	Provide a registry for choreographies, services and SLAs	VERY HIGH	EBM	The IDRE MUST Provide a registry for choreographies, services and SLAs. This allow their discovery at design and run-time. The registry must be able to deal with a scalable number of services and choreographies.	WP3-4	Management
CHOR-GOV-005	Provide test activation	HIGH	CNR-UNICAM	The IDRE MUST Provide test activation mechanisms.	WP4	Analysis

3.3. IDRE implementation requirements

Goal: This section describes those requirements that need to be satisfied in the activities related to the implementation of the IDRE. They add technical and methodological details to what is described in section 3.1, which lists important features that have to be considered by the IDRE to support the whole lifecycle of a choreography.

Such requirements aim at making technically easier the integration of the software artifacts developed within the different WPs,

3.3.1. IDRE design requirements

Goal: describe the requirements for the design **of the IDRE**.

ID	Name	Priority	Partner	Description	WP
IDRE-DES-001	Provide application independence	HIGH	CNR-UNICAM	The IDRE MUST be application independent	WP5
IDRE-DES-002	Transform data formats	LOW	CNR-UNICAM	The data formats e.g. for service level agreements, exchanged messages, and governance policies, MAY be converted into a different target data formats. In such transformation, the IDRE SHOULD grant for preserving the semantic of the data.	WP1-5
IDRE-DES-003	Legacy System	LOW	CNR-UNICAM	The IDRE MAY support facilities for legacy integration using a wide range of technologies, e.g. migration or wrapping, using various technologies, e.g. model driven approaches.	WP1-4-5
IDRE-DES-004	Identify different software components	VERY HIGH	VTRIP	Software components derived by WP2-4 MUST be defined explicitly, in order to be integrated in the IDRE platform.	WP2-4
IDRE-DES-005	Maintain interoperability for IDRE components	VERY HIGH	VTRIP	The IDRE software components MUST be able to communicate together through well defined interfaces ad behaviour	WP2-4
IDRE-DES-006	Define a specific IDRE communication protocol	VERY HIGH	VTRIP	The communication between different software components of the IDRE MUST be delivered with a predefined set of rules and admissions constituting the protocol	WP2-4
IDRE-DES-007	Account for fault tolerance in the IDRE protocol	HIGH	VTRIP	Protocol definition SHOULD encompass error handling towards the direction of sustaining interoperability between the software components.	WP2-4
IDRE-DES-008	Maintain data consistent between IDRE processes	MEDIUM	VTRIP	Any data that is used across the IDRE MUST be consistent. Input, configuration, models representation, rules & policies that are used by more than one component must have the same content.	WP2-4
IDRE-DES-009	Design the Architecture Model in UML	HIGH	NME	IDRE architecture MUST be captured using UML based models	WP2-4-5
IDRE-DES-010	Conceptual Model	HIGH	NME	IDRE architecture MUST capture its major concepts and their relationships in Class Diagrams.	WP2-4
IDRE-DES-011	Process Model	HIGH	NME	IDRE architecture MUST capture processes for developing and running choreographies using IDRE capabilities in Activity Diagrams.	WP2-4
IDRE-DES-013	Component Model	HIGH	NME	IDRE architecture MUST contain its decomposition into components and specify their interfaces, interconnections, and data flows.	WP2-4

ID	Name	Priority	Partner	Description	WP
IDRE-DES-014	Use Case Model	HIGH	NME	IDRE architecture MUST capture use cases in Use Case Diagrams defining actors and use cases within components.	WP2-4
IDRE-DES-015	Data Model	MEDIUM	NME	IDRE architecture SHOULD precisely capture data structures in Class Diagrams.	WP2-4
IDRE-DES-016	Test Case Model	MEDIUM	NME	IDRE architecture SHOULD capture test cases in form of packages with Object Diagrams containing instance specifications for inputs and expected outputs.	WP2-4
IDRE-DES-017	Interaction Model	MEDIUM	NME	IDRE architecture SHOULD capture component interactions for selected use case scenarios in form of Sequence Diagrams.	WP2-4
IDRE-DES-018	User Interface Model	LOW	NME	IDRE architecture MAY capture design of specific user interface fragments in User Interface Modeling Diagrams and design of user interface navigation in Statemachine Diagrams.	WP2-4
IDRE-DES-019	Implementation Model	LOW	NME	IDRE architecture MAY capture mapping of logical components to the implementation elements reversed from source code.	WP2-4
IDRE-DES-020	Deployment Model	LOW	NME	IDRE architecture MAY capture information on how logical components are manifested by deployable artifacts that are deployed on hardware nodes.	WP2-4
IDRE-DES-021	Architecture Simulation Model	LOW	NME	Complex parts of IDRE architecture MAY be investigated using detailed architecture models prepared for simulation using tools as Cameo Simulation Toolkit for example.	WP2-4
IDRE-DES-022	Modeling Guidelines	HIGH	NME	IDRE architecture model SHOULD follow modeling guidelines that are captured in document. Modeling tools such as Magic Draw could be used.	WP2-4
IDRE-DES-023	Architecture Model Documentation	MEDIUM	NME	Significant elements of IDRE architecture model SHOULD be documented with textual descriptions providing definitions, motivation, and other important details.	WP2-4
IDRE-DES-024	Architecture Report Suitable for Web Browsers	HIGH	NME	IDRE architecture model MUST be published in any format in order to be accessible via a regular Web browser.	WP2-4
IDRE-DES-025	Architecture Model Governance	HIGH	NME	IDRE architecture model MUST be governed according to agreed procedures for versioning, review, approval, and change management.	WP2-4
IDRE-DES-026	Architecture Model Compliance	HIGH	NME	IDRE architecture model MUST be in sync with code.	WP2-4

ID	Name	Priority	Partner	Description	WP
IDRE-DES-027	Distributed architecture	HIGH	CNR-UNICAM	The IDRE MUST support the exchange of information in a highly distributed way. Thus, the IDRE MUST NOT rely on any centralized architectural component or service. The IDRE architecture needs to cope with highly scalable environments. The runtime middleware may benefit from the elasticity of cloud-aware infrastructures (WP3).	WP3-5

3.3.2. IDRE development requirements

Goal: describe the global requirements for the development **of the IDRE** and its sub-components.

ID	Name	Priority	Partner	Description	WP(s)
IDRE-DEV-001	Use common software libraries	VERY HIGH	EBM	All components of the IDRE, across each WP, MUST be developed by using common software libraries shared between partners.	WP2-3-4-5
IDRE-DEV-002	Adopt common developing rules	VERY HIGH	EBM	All components of the IDRE, across each WP, MUST be developed by using common developing rules shared between partners.	WP2-3-4-5
IDRE-DEV-003	Apply the separation of concerns	VERY HIGH	EBM	Each software library MUST be developed by separating the implementation from the interfaces. Interfaces of components are exposed in order to be used by consumers.	WP2-3-4-5
IDRE-DEV-004	Realize Functional tests	HIGH	EBM	Functional tests for each software project/components MUST be implemented.	WP4-5
IDRE-DEV-005	Realize automatic tests	HIGH	EBM	Automatic tests MUST be continuously realized while developing the software libraries.	WP5
IDRE-DEV-006	Separate Use Cases projects	HIGH	EBM	Each use case MUST be put in a separate project.	WP5
IDRE-DEV-007	Use non duplicated resource files for the use cases	HIGH	EBM	Resource files of use case MUST be unique and not duplicated or cloned.	WP5
IDRE-DEV-008	Adopt a common Choreography language	HIGH	EBM	A common choreography language MUST be adopted. BPMN2.0 is a graphical language for the choreography and orchestration design.	WP2-3-4-5
IDRE-DEV-009	Consider a common development and execution object oriented kit	HIGH	EBM	A common development object oriented language is REQUIRED. Object oriented language integrates the separation of concerns paradigms that are helpful for SOA programming. JDK 1.6.x is a stable version of java development Kit.	WP5
IDRE-DEV-010	A common subversion system	VERY HIGH	MLS	All components of IDRE MUST be developed by using a common subversion system. RapidSvn is a commonly used open source software that achieves this requirement.	WP4-5

ID	Name	Priority	Partner	Description	WP(s)
IDRE-DEV-011	Documented code	HIGH	MLS	The code that will be developed should be documented in classes and entities level. The entire code documentation SHOULD come out of a software. An open source software that satisfies this goal is Doxygen	WP4-5
IDRE-DEV-012	Documented installation procedure	MEDIUM	MLS	The installation procedure for the IDRE as well as its components SHOULD be documented.	WP5
IDRE-DEV-013	Programming language independence	HIGH	CNR-UNICAM	The IDRE MUST be programming language agnostic	WP5
IDRE-DEV-014	Use of standard components	HIGH	CNR-UNICAM	The IDRE SHOULD adopt and deploy standard software components to perform well-defined tasks instead of integrating these tasks into custom software.	WP2-3-4-5
IDRE-DEV-015	Use of standard protocols	HIGH	CNR-UNICAM	The IDRE MUST adopt standard protocols to for the communications among both its services and components.	WP2-3-4-5
IDRE-DEV-016	Data format	LOW	CNR-UNICAM	Independently from the communication protocol, the data itself may be described in multiple formats. For exchanging order information, the most common formats found in real world examples are XML, TXT, and CSV.	WP2-3-4-5

3.3.3. IDRE deployment requirements

Goal: describe the requirements for the deployment **of the IDRE**.

ID	Name	Priority	Partner	Description	WP
IDRE-DEP-OO1	Deploy on top of a service oriented middleware dedicated to Future Internet	VERY HIGH	EBM, USP, INRIA	The CHOReOS Runtime Middleware MUST be able to be deployed on top of a service oriented middleware dedicated to Future Internet. Existing middleware such as Enterprise Service Bus, Cloud-aware infrastructure and IoST provide scalable and distributed runtime environments. The CHOReOS IDRE MUST be able to run choreographies on top of these.	WP3
IDRE-DEP-002	Run on top of a highly distributed Service Bus	VERY HIGH	EBM	The CHOReOS Runtime Middleware MUST support the deployment of choreographies on top of a highly distributed service bus. Distributed service bus enables integration of heterogeneous and distributed applications and services.	WP3
IDRE-DEP-003	Run on top of highly scalable Cloud infrastructure	VERY HIGH	USP	Instantiate the CHOReOS Runtime Middleware and the nodes of a choreography in virtual machines of a given cloud infrastructure, e.g., Amazon EC2, Open Cirrus, or Open Nebula. The cloud infrastructure enables elastic and scalable resource management.	WP3
IDRE-DEP-004	Run on top of heterogeneous and mobile 'smart' things	VERY HIGH	INRIA	The CHOReOS Runtime Middleware SHOULD be able to execute on devices with small form factors and limited capabilities -- the so called smart "things". Examples include smart phones, and constituent nodes of Wireless Sensor Networks such as Sun SPOTs.	WP3

ID	Name	Priority	Partner	Description	WP
IDRE-DEP-005	Be platform independent	HIGH	INRIA	The CHOReOS Development Environment SHOULD be able to work on Windows, Linux, and Mac desktop OSs.	WP3-4-5

4. Annexe A – Airport Case Study requirements

As an identified case study for CHOReOS deployment, the “Passenger Friendly Airport” case study (herein also called “Airport CS) is of particular interest for this deliverable as it gives early business requirements that can be linked, as examples, to the higher level requirements defined in this document.

Overview of the case study

The Airport CS takes place in the context of air transportation, which is becoming a major activity in our everyday life either for business or leisure. Air transportation has indeed proven to be particularly safe and reliable, which will remain the primary objective for all actors of a flight. **However, improvements in the services provided by air transportation are much needed,** especially when one considers capabilities brought by the networking of the various services provided by the multiple stakeholders involved.

One can consider that there are two main directions for improvements: (i) continue optimisation of air traffic to, in particular, reduce delays, and (ii) better serve passengers.

We propose to focus this scenario in the latter direction where the choreography approach will prove to be an efficient means for improving services provided to passengers. Thus, it aims to **demonstrate and assess the exploitation of choreographies in day-to-day large-scale coordination of air transportation stakeholders** (Air traffic control authorities, Airports, Airlines…) and associated logistics partners (ground transportation, hotels…).

For more details about the particular scenario followed by the Airport CS and the involved actors, please refer, for the time being[2], to section B.1.3.2 of the CHOReOS DoW – Part B – Narrative Part document.

Requirements

In this annex, we identify a preliminary list of business-related requirements dedicated to the Airport CS. From these requirements, we may infer global requirements for the CHOReOS IDRE (in a typical bottom-up fashion), or even provide concrete example to support these global requirements (top-down approach).

Still, note that at this relatively early stage, the provided list of requirements is only partial and will be completed as part of the dedicated task within WP6[2] , which is to be initiated after the publication of the first version of this document. However, by collaborating early with a domain expert, who happens to also be a computer scientist, we inform the early elicitation of the IDRE requirements reported in this document, so that they can be effectively reused and form a solid basis for other work packages.

In the following, business requirements are categorized according to:

- **Functional requirements**: requirement that pertain to the intrinsic features of the system.
- **Non-functional requirements**: requirement that pertain to the realization modalities of the feature of a system (i.e. its Quality of Service).
- **Design requirements**;
- **Runtime requirements**.

[2] Work Package (WP) 6, starting at T0+6 will provide more details on the Airport UC scenario and will refine this preliminary list of requirements.

Airport UC – Functional requirements		
ID	**Name**	**Description**
AIRPORT-FUN-001	Dynamic service discovery	In the Airport UC, a flight can be rerouted and critical services from new destination infrastructures need to be discovered since they differ from the original destination. As such, the IDRE MUST support the dynamic discovery of services during the enactment of orchestrations.
AIRPORT-FUN-002	Ultra Large Scale (ULS) choreography support	Due to its large list of actors (ATC, Airline and Airport entities, including all its members, as well as all passengers in need of information), the Airport UC MUST have encompassing ULS support at design and at runtime.
AIRPORT-FUN-003	Dynamic access control to services	In the context of airport crisis, services SHOULD have their access control relaxed to authorize temporary consumption.
AIRPORT-FUN-004	Dynamic access control to passenger data	In order to personalize services provided to the passenger, access to travel data SHALL be granted to applications/operators due to circumstances. Modification of access rights SHALL be temporary and attached to the context.
AIRPORT-FUN-005	Search for flight related choreographies	Administration interface SHALL provide easy way to search for choreographies related to a flight call sign
AIRPORT-FUN-006	Control of flight related choreographies	Administrator SHALL be able to suspend, stop, or restart any choreography.
AIRPORT-FUN-007	Choreography with human interaction	Choreography SHOULD allow for human interaction in the loop in order to apply modifications at runtime.
AIRPORT-FUN-008	Choreography manual control	When decision is made to reroute a flight, the decision SHOULD appear on a supervision interface, with a list of associated choreographies. Operator would be able to confirm the start of all choreographies or even cancel some of the list.
AIRPORT-FUN-009	Choreography overview	At any time, administrator SHALL be able to see the status (failed, successful, aborted, suspended…) of all choreographies associated to a flight
AIRPORT-FUN-010	Choreography definition	A choreography designer SHALL be able to design new choreography (or modify existing ones) using a graphical notation and exploiting repository of available assets (organisations, services…)

Airport UC – Non-functional requirements		
ID	**Name**	**Description**
AIRPORT-NFU-001	Circle of trust	Organisations collaborating within the scenario SHALL share user identities or system identities to grant access to their services and data (No central identity management)
AIRPORT-NFU-002	Confidentiality	Interactions between service consumers and providers SHALL be secured to guarantee confidentiality of exchanged data.
AIRPORT-NFU-003	Message queuing	Exchanges between organisations SHALL be performed in a secured way so that interactions may recover from temporary unavailability of services

AIRPORT-NFU-004	Integrity	Exchanges between organisations SHALL be secured to avoid any corruption of data in between

Airport UC – Design requirements		
ID	**Name**	**Description**
AIRPORT-DES-001	BPMN 2.0 support	From an industrial standpoint, the BPMN 2.0 specification is well known, supported by OW2, and provides the paradigms to define elaborated and nested choreographies necessary in the Airport UC. As such it MUST be included in the IDRE.
AIRPORT-DES-002	BPMN 1.x support	BPMN 2.0 is quite new and already existing BPMN 1.x choreographies from Airport Authorities and actors, if they exist, SHOULD be integrated in the IDRE.
AIRPORT-DES-003	Collaborative choreographies	Since many different actors (from ATC down to personnel) are involved in (or in charge of) parts of the journey of a typical passenger, each of these actors SHALL define their own parts in choreographies, since they are the experts to talk to.
AIRPORT-DES-004	Ultra Large Scale (ULS) choreography design (inherits from AIRPORT-FUN-002, AIRPORT-DES-003)	Support of an ULS choreography in the Airport UC also mean that each of its "decision making participant" (e.g. ATC, Airline and Airport entities) could also be one of its potential designer. As such, CHOReOS IDRE MUST also support the design of ULS choreographies, in a collective fashion.

Airport UC – Runtime requirements		
ID	**Name**	**Description**
AIRPORT-RUN-001	Runtime flexibility	The coordination of all services SHALL be highly flexible due the heterogeneity of services between airports.
AIRPORT-RUN-002	Runtime Distributivity	Since there MUST NOT be any single point of control or failure in a choreography (especially critical airport regulation ones), the responsibility of the execution MUST be distributed through all its participants. The Runtime MAY support such an *enactment* feature (e.g. by translation of a given choreography into multiple orchestrations).
AIRPORT-RUN-003	Runtime monitoring of choreographies	The architecture SHALL support the monitoring of enacted choreographies at runtime, particularly according to nun-functional characteristics specific to the Airport UC.
AIRPORT-RUN-004	Ultra Large Scale choreography enactment (inherits from AIRPORT-FUN-002)	Due to ULS constraints in the Airport UC, the Runtime environment MUST support ULS choreographies.
AIRPORT-RUN-005	Surveillance of choreography	Any running choreography SHALL be monitored and controlled from remote client position.
AIRPORT-RUN-006	Traceability	Choreographies SHOULD log all service calls and their status changes with time and unique ID (per choreography).