vf-OS: virtual factory Operating System

WP1: Vision, Scenarios, and Requirements

D1.5: Requirements Specification (M6)  
Vs: 1.0.1

Deliverable Lead and Editor: Carlos Coutinho, CMS  
Contributing Partners: All partners  
Date: 2017-05  
Dissemination: Public  
Status: EU Approved

Short Abstract
The requirements of the vf-OS project, this document, provide an essential tool for fixing the project scope by eliciting the needs of pilots and software developers as final users of vf-OS applications, vApps. Expectations and assumptions are included to reflect what the project will actually do and not do. It is also a critical source of information for the design, development and implementation activities to be carried out as well as to validate the project outcomes.

Grant Agreement: 723710
Document Status

<table>
<thead>
<tr>
<th>Deliverable Lead</th>
<th>Carlos Coutinho, CMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Reviewer 1</td>
<td>Andries Stam, ALM</td>
</tr>
<tr>
<td>Internal Reviewer 2</td>
<td>Eduardo Saiz, IKER</td>
</tr>
<tr>
<td>Internal Reviewer 3</td>
<td>Oscar Garcia, ICE</td>
</tr>
<tr>
<td>Type</td>
<td>Deliverable</td>
</tr>
<tr>
<td>Work Package</td>
<td>WP1: Vision, Scenarios, and Requirements</td>
</tr>
<tr>
<td>ID</td>
<td>D1.5: Requirements Specification (M6)</td>
</tr>
<tr>
<td>Due Date</td>
<td>2017-03</td>
</tr>
<tr>
<td>Delivery Date</td>
<td>2017-05</td>
</tr>
<tr>
<td>Status</td>
<td>EU Approved</td>
</tr>
</tbody>
</table>

History
See Annex A

Status
This deliverable is subject to final acceptance by the European Commission.

Further Information
www.vf-OS.eu

Disclaimer
The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.
Project Partners:
Executive Summary

The deliverable “D1.5 Requirements Specification (M6)” is intended to summarise the main requirements to be provided by the vf-OS framework (which includes the vf-OS Platform and related tools, and a set of applications, “vApps”). This deliverable sums up the results of the work carried out in task T1.5 (Requirements Specification (M6)) of the project.

This task aimed to cover three of the objectives defined in the Description of Action (DOA) for the development of WP1, specifically O1.2, O1.3, and O1.5, devoted to the elicitation, analysis, and establishment of the set of requirements to be fulfilled by the vf-OS results.

The task includes the gathering of specific characteristics and needs of the industrial pilots of the project, coming from vf-OS tasks T1.2 (Users Scenarios Characterisation) and T1.3 (Providers Scenarios Characterisation). It uses this information to challenge the vf-OS vision defined in T1.1 (Vision Consensus), and the specifications, functional solutions and limitations of the technological solutions defined in T1.4 (Existing Solutions and Technology Identification/Analysis).

With those inputs, this task defined the set of functional and non-functional requirements to be satisfied by the vf-OS framework comprising the vf-OS Platform and vApps, covering:

- The definition of the methodology used to define the functional and non-functional requirements to consider during the development of the vf-OS Platform
- The description of the needs and requirements provided by the different industrial pilots participating in the project
- The compiled functional requirements addressed for the development of the vf-OS components
- The identification of non-functional requirements to be taken into account

The main result of this task is a comprehensive list of requirements (see Annex C) grouped into 3 categories:

- “User” requirements, provided by the different selected pilots (vApps users) and by the vApps software developers. In this document, the term “User” and “Customer” shall be used as synonyms, referring to the stakeholders that are the target of the vf-OS framework, eg coming from the project’s Pilots, from industry, or other interested parties that want to use vf-OS
- “System” requirements, aiming to provide the architectural structure for satisfying the “User” requirements
- “Software” requirements that implement the “User” requirements according to the features and limitations imposed by the “System” requirements

This deliverable is the starting point of the vf-OS project towards the development activities to be carried out in WP2 to WP8.
Table of Contents

0 Introduction ......................................................................................................................... 1
1 Scope and Methodology ........................................................................................................ 5
  1.1 Introduction ..................................................................................................................... 5
  1.2 Objectives ....................................................................................................................... 5
  1.3 Methodology to Produce the Deliverable ........................................................................ 6
    1.3.1 Inputs Analysis ........................................................................................................... 7
    1.3.2 Definition of Requirements Sources .......................................................................... 8
    1.3.3 Definition of Requirement Levels and Traceability .................................................. 8
    1.3.4 Requirements Post-processing and Change Management ......................................... 9
  1.4 Literature analysis ........................................................................................................... 9
  1.5 Main Results and Outputs ............................................................................................. 9
2 Elicitation Activities ............................................................................................................ 11
  2.1 Strategy ........................................................................................................................ 11
    2.1.1 User Requirements ..................................................................................................... 12
    2.1.2 System Requirements .................................................................................................. 12
    2.1.3 Software Requirements ............................................................................................. 13
  2.2 Guidelines on how to Perform the Requirements Elicitation ......................................... 13
    2.2.1 Analysis of the Scope of Requirement Types ............................................................. 13
    2.2.2 Requirements Elicitation Best-Practices ................................................................. 14
  2.3 Requirements Scope ...................................................................................................... 15
  2.4 Analysis and Post-Processing ....................................................................................... 16
3 Requirements Specification ................................................................................................ 17
4 Conclusions and Future Work ............................................................................................ 19
Annex A: History ..................................................................................................................... 20
Annex B: References ............................................................................................................... 21
Annex C: List of vf-OS Requirements .................................................................................... 22
0 Introduction

0.1 vf-OS Project Overview

vf-OS – virtual factory Open Operating System – is a project funded by the H2020 Framework Programme of the European Commission under Grant Agreement 723710 and conducted in the period October 2016 until August 2019. It engages 14 partners (Users, Technology Providers, Consultants and Research Institutes) from 7 countries with a total budget of circa 7.5M€. Further information can be found at www.vf-OS.eu.

The World is facing the fourth industrial revolution based on ICT, specifically architectures and services, as key innovation drivers for manufacturing companies. Traditional factories will increasingly be transformed into smart digital manufacturing environments but currently the full potential for ICT in manufacturing is far from being fully exploited. Factories are complex systems of systems and there is a need to develop a platform on which future manufacturing applications can be built. Examples of platforms exist in some industrial sectors but there is a lack of cross cutting platforms based on open standards for creating an ecosystem for cooperative innovation. Innovative open platforms to attract talent from solution developers and to provide accessible manufacturing smart applications to European SMEs are examples of the kind of solutions being sought.

The goal of vf-OS is to develop an Open Operating System for Virtual Factories composed of a kernel, application programming interface, and middleware specifically designed for the factory of the future. An Open Applications Development Kit (OAK) will be provided to software developers for deploying Manufacturing Smart Applications for industrial users, using the vf-OS Manufacturing Applications Store all operated through a Virtual Factory Platform.

The Virtual Factory Platform is an economical multi-sided market platform with the aim of creating value by enabling interactions between four customer groups:

- **Software Developers** (independent or within individual manufacturers) which will build Manufacturing Apps either through innovation or from manufacturing user demand
- **Manufacturing and Logistics Users** which will explore the marketplace for already created solutions, ready to be run on the vf-OS
- **Manufacturing and Logistics Solution Providers** which will provide ICT interfaces and manufacturing connections
- **Service Providers** (vf-OS innovators and third parties) will make available services (hosting, storage, connected cloud services, etc.) including those based on developed solutions
The Virtual Factory Platform will provide a range of services to the connected factory of the future to integrate better manufacturing and logistics processes. The Manufacturing Applications Store will be open to software developers who, using the free Open Applications Development Kit provided, will be able to quickly develop and deploy smart applications to enable and optimise communication and collaboration among supply networks of all manufacturing sectors in all the stages manufacturing and logistic processes.

vf-OS aims to become the reference system software for managing factory related computer hardware and software resources and providing common services for factory computational programs. This operating system will be the key component of the software system in a real factory system where all factory application programs will run.

0.2 Deliverable Purpose and Scope

The purpose of this document “D1.5 Requirements Specification (M6)” is to define the requirements scope of the vf-OS project, ie of the tools, platforms and applications which comprise the framework that needs to be developed on this project. This definition encompasses the elicitation and analysis of needs and requirements. These are coming from multiple sources, which include the project’s pilots, added to a generic pilot built from the teams’ expertise and from bodies of knowledge and best practices, and also from the vf-OS Vision and Mission ideas. These will be most useful for the forthcoming project design and implementation, but also for the definition of what the target businesses will expect from the project, how they will verify and validate the implementation outcomes and what evidence is needed to collect for the acceptance of these requirements.

The work done for vf-OS task T1.5 started by analysing the specific characteristics and needs of the different industrial pilots of the project, and the relevant feedback collected in deliverables D1.2 (User Scenarios) and D8.1 (Validation Scenarios). This provided the main industrial needs to be addressed, and helped define the set of functional and non-functional requirements to be provided by the vf-OS framework. This deliverable thus reflects the approach taken to define the functionalities to be provided by vf-OS as well as the non-functional requirements to be fulfilled by its different components. The deliverable includes:

- The definition of the methodology used to define the requirements to consider during the development of the vf-OS framework
- The description of the needs and requirements provided by the different industrial pilots participating in the project (vApps users) and vApps developers
- The compiled functional and non-functional requirements addressed to the development of vf-OS components

The main results reported in this deliverable can be grouped into three categories:

- User-level needs (or User Requirements)
- System Requirements
- Software Requirements

This deliverable, despite being targeted for delivery at project month M6, will be a living document, subject to changes during the project life cycle which will mainly affect the list of requirements defined in Annex C. These will be marked as different versions in the
corresponding field, so the users will not have to revise the entire document but only search for the changes concerning each version.

0.3 Target Audience

The document “D1.5 Requirements Specification (M6)” is aimed primarily at project developers and describes the user needs that will guide their work on developing the project framework. It will also support the verification and validation procedures to be performed later in the project life cycle (in WP8).

0.4 Deliverable Context

This deliverable D1.5 has grown side-by-side with two other major input specifications for the vf-OS project:

- **User Scenarios Characterisation (D1.2):** Report defining the use-case scenarios that are being addressed in vf-OS and its industrial pilots
- **Validation Scenarios (D8.1):** Report describing what type of analysis and validation will be performed by the industrial pilots for the acceptance of the specified requirements.

These three deliverables were a direct input for all remaining design and implementation tasks of Work Packages WP2 to WP8.

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1:** *Scope and Methodology:* Performs a literature analysis and describes the adopted methodology for conducting the whole process of requirements gathering
- **Section 2:** *Elicitation Activities:* Describes the set of activities that were performed for the execution of the elicitation and collection of requirements
- **Section 3:** *Requirements Specification:* Describes the actual list of requirements, its structure, and meaning
- **Section 4:** *Conclusions and Future Work:* Provides the conclusions of this task’s work

Annexes:

- **Annex A:** Document History
- **Annex B:** Reference
- **Annex C:** Spreadsheet of vf-OS requirements

0.6 Document Status

This document is listed in the Description of Action (DOA) as public access.
0.7 Document Dependencies

This document has no preceding documents or further iterations. Although the requirements of the vf-OS project may evolve with the project life cycle, leading this document to change to properly document this evolution, no other formal iterations of this deliverable are expected. Should this evolution occur, it will be formalised by providing a new version of this deliverable at the end of the project.

0.8 Glossary and Abbreviations

A definition of common terms related to vf-OS, as well as a list of abbreviations, is available in the supplementary and separate document “vf-OS Glossary and Abbreviations”.

Further information can be found at http://www.vf-OS.eu/glossary

0.9 External Annexes and Supporting Documents

External Annexes and Supporting Documents:

- Annex C: Requirements Spreadsheet

0.10 Reading Notes

None
1 Scope and Methodology

1.1 Introduction

Requirements’ gathering is one of the main activities to be carried out for the definition of a project’s scope, and the basis of project design. Its purpose is to elicit, analyse, and establish the user needs and transform them into the vf-OS framework requirements. The requirements gathering is an iterative process of defining statements about an intended functionality of the system which specify what should be implemented to fulfil and accomplish that functionality.

Taken together, the requirements address the needs of relevant stakeholders, including the needs that are pertinent to the various product lifecycle phases [CKS11]. CMMI (Capability Maturity Model Integration) defines the following major steps for Requirements’ Development:

- Elicitation of needs, using document analysis and reviews, interviews, questionnaires, and analysis of use-cases and User stories
- Transformation of stakeholder needs into Customer Requirements
- Refinement of the Customer Requirements into System and Software Requirements
- Analysis of the requirements, ensuring they are necessary and sufficient
- Balance stakeholder needs and constraints
- Validation of the requirements by the major stakeholders

The methodology used for defining the vf-OS requirements included all above actions and best-practices, including the analysis of documentation (eg, D1.2 User Scenarios Characterisation), face-to-face meetings, and remote talks between the eliciting team and pilots’ stakeholders. It also included strict actions of both converting the acquired knowledge to User Requirements, and of refinement of the User Requirements to System and these last to Software Requirements. Each of these actions was finalised by a validation from the stakeholders of each customer pilots, which included checks for completion and sufficiency and validation of the correctness of the requirements.

Additionally, the PMI PMBoK (Project Management Institute’s Project Management Body of Knowledge) [PMI13] states that requirements should always include a traceability matrix that links the product requirements to their source (and, additionally, to the component implementation that satisfies them).

In the case of vf-OS requirements, this traceability was defined, including the connection between the User Requirements and its source (eg, interview, document), as well as the connection between the User, System, and Software Requirements. In future vf-OS tasks, namely in WP2 and WP8, it is expected that this traceability will be extended to trace and interconnect the requirements with product design and development evidences, and with test scenarios and test cases.

1.2 Objectives

The main objective of the task T1.5 is to establish clear boundaries for the scope of the vf-OS project. This means the elicitation, analysis, validation, and registration of customer needs, expectations, and constraints to obtain prioritised customer requirements that constitute an understanding of what will satisfy the stakeholders of the vf-OS project (aka
“customers”). These customers will be the users of the vf-OS pilots and all potential users from future businesses that may use the vf-OS framework (e.g., vApps software developers).

The objective is therefore to obtain a clear definition of the set of vf-OS functionalities that are sufficient and realisable as well as the definition of the project’s scope and boundaries (i.e., what is in-scope and what is out-scope), resulting in a traceable network of requirements.

As the vf-OS consortium is very heterogeneous in terms of the characteristics of the different partners, it was important to use this value as an add-on advantage, where the following can be found:

- Industrial partners that are close to the industrial reality and can provide valuable insights into the needs of the real industry. On the other hand, perhaps these partners are too close to the detailed problems and intricacies and are therefore more likely to miss “the big picture” or new opportunities that can be raised by such a platform.
- Research institutes that provide the vf-OS project with a mission and a clear vision, an objective view over the target achievements and functionalities to be reached, and the results that will be considered a breakthrough over the existing state of the art. These partners might however be too bold and ambitious, neglecting the real needs of the real scenarios and by doing so, might be considered as vague, utopic, incompatible, or unachievable.
- SMEs and other technical partners that have a lot of experience supporting the technology industry and are always updated on the latest trends, which can help with the analysis of the market alternatives, existing services, platforms, and methodologies.

Hence, the added-value of having such a valuable panel of partners is to be able to capture the best of each partner’s contributions, and to balance between the expertise and savvy of the experienced, and the novelty of the hi-tech rookies. The activities comprised in this task were aimed at using the complemented contributions from these profiles in brainstorming sessions, in order to obtain multiple, alternative, and heterogeneous visions of the needs and project scope.

### 1.3 Methodology to Produce the Deliverable

This task is a joint effort with all vf-OS partners involved. The initial phase of this task was aligned with the starting activities of the vf-OS project, which were about gathering the knowledge, and expertise of all partners, during common face-to-face meetings, on the pilots’ premises, via remote teleconference, and (even more remotely) with the use of sets of questionnaires and interviews with the pilots’ industrial partners. From this joint effort, at least three deliverables were compiled: the vf-OS User Scenarios Characterisation (D1.2), the vf-OS Validation Scenarios (D8.1), and the Requirements Specification (D1.5).

Particularly for the latter deliverable, the vf-OS team defined a methodology for the whole requirements development process. This process included several steps as described in the sections below:

- To perform an analysis of the project inputs
- To identify the requirements sources
- To determine the type of elicitation to perform for each requirement type
• To define the traceability on the requirements
• To define post-processing actions over the elicited requirements
• To define requirements’ change management rules

By using this methodology, the vf-OS requirements life cycle follows the project life cycle, and encompasses the steps of:
• Elicitation of requirements
• Requirements specification validation from stakeholders
• Design with traceability towards requirements
• Development of the functionalities that fulfil the requirements
• Testing the requirements, with traceability between test cases and requirements
• Test results communication to stakeholders
• Acceptance of requirements implementation from stakeholders

1.3.1 Inputs Analysis

The first stage regarding the preparation for the requirements elicitation was to clearly identify the relevant stakeholders. Besides identifying the stakeholders, there was a need to determine a representative of each stakeholder - ie somebody that can help to understand the stakeholder views and who can validate the finalised requirements.

The identification of the relevant stakeholders took into account three separate views of the project, provided by:
• vf-OS Pilots: The “customers” identified more clearly were the vf-OS pilots, as one of the objectives of the project is to develop vApps that aim to satisfy the needs of these manufacturing companies:
  • Pilot 1: MASS + VS, represented by Jon Altube (MASS)
  • Pilot 2: CON, represented by Fernando Monteiro (CON)
  • Pilot 3: APR + TARDY, represented by Ibrahim Benali (TARDY)

• vf-OS Vision Pilot: Important inputs for the development of the project were the vf-OS Mission and Vision. It is essential that the project would not resume to simply being a solution to fit the needs of the pilots; it must have its own objectives and vision. Hence, this major stakeholder needs to care about:
  • The vf-OS ideas, objectives, purpose, and mission
  • What the vf-OS consortium wants for future exploitation purposes
  • How to achieve the desired step beyond the state of the art
  • Technological limitations
  • The definition of what is in and out of the scope of vf-OS
  • Business policies
  • Other constraints (eg project budget, project time, EC regulations)

The whole vf-OS consortium should be the main client here, but as there was the need for pointing out one accountable “customer”, ICE was selected not only for being the consortium lead but also because ICE led vf-OS D1.1 (vf-OS Vision consensus). This “customer” was represented by Óscar Garcia (ICE), the editor and lead of D1.1.
• **Generic (abstract) Pilot**: An important objective of vf-OS, besides developing its own vision and satisfying its industrial partners by means of the three pilots, is to become a solution that is widely adopted by the manufacturing industry, which goes largely beyond the project partners. To achieve this (and given the large expertise that most partners have in supporting manufacturing companies), a Generic Pilot was conceived, based on:
  
  • Best-practices, bodies-of-knowledge, and standards
  • Experts in the manufacturing area
  • State-of-the-art in technology and related solutions

These five pilots and their representatives were responsible for providing inputs regarding the specific needs related to its own scope, to defend their views and positions, and to validate the completeness and sufficiency of the elicited requirements.

1.3.2 **Definition of Requirements Sources**

The elicitation of requirements described in detail in Section 2 was performed taking into account the analysis of multiple tools and artefacts, which included:

- Interviews
- Questionnaires and surveys
- Documentation analysis
- Legacy software
- Stakeholders contracts, commitments, SLAs (Service Level Agreements) with suppliers, partners, customers
- Brainstorming

1.3.3 **Definition of Requirement Levels and Traceability**

When performing the requirements elicitation, the team needed to clearly scope the level of each requirement that was being elicited, which was intrinsically tied to the elicitation phase (see section 2.1). These levels reflect some of the characteristics that are associated to the requirement itself, such as:

- **User/Customer/Business Requirements** (from the five pilots) are specified using a language that is very close to the business itself. They specify the direct needs of the pilots regarding the functionalities and constraints the pilots would like to see implemented by vf-OS. They form the High-Level User Needs Specification, and are specified on Annex C as “User Requirements”.

- **System Requirements** refer to specifications over the environment implementing the vf-OS, eg interfaces, APIs, high-level modules, and life cycle of the products and functionalities. They are mainly derived from the previously defined User Needs, but some of these requirements may be new and not related to the business requirements. These will be most valuable for supporting the definition of the vf-OS Architecture (WP2).

- **Software Requirements** refer to all technical specificities that derive from the User and System requirements. Where system requirements roughly represented high-level modules, software requirements detail this specification, using mainly software developer language. They represent a low-level analysis down to the detail of implementation of the functionalities and constraints that are needed for the five
pilots, specified in a language that is close to the software developers. These form the foundation for the vf-OS software development tasks.

These levels of requirements should evolve with a proper traceability, where each higher-level requirement should trace to at least one requirement of the next lower level. There are two types of exceptions:

- Some requirements of a lower level cannot be traced upwards to a higher level because they are specific to that level of detail and it is not appropriate to “invent” a higher-level requirement just for that.
- Some higher-level details are already specified in a manner where they can be interpreted as lower-level requirements, and it is useless to have duplicate requirements just to maintain the traceability.

1.3.4 Requirements Post-processing and Change Management

More than just eliciting the requirements, the whole process needs to be defined regarding the correct storage, maintenance, and change management for each requirement. This includes:

- **Establishing unambiguous terminology**: The vf-OS project glossary will be used for clarifying the ambiguities found in the requirements terminology
- **Developing the requirements**: This will be the actual development of the list of requirements
- **Identifying inconsistencies, duplicates and contradictory requirements**: After the list of requirements is captured, there is the need to analyse it searching for problems
- **Analysing and validating the requirements**: The final list is then compiled and validated by the respective customer. This means the customer has an opportunity at each elicitation phase to analyse the requirements text, checking if the customer agrees with it and if it is compliant to its pilot’s constraints and needs
- **Managing changes**: All changes to any requirement must be performed with the approval of its accountable person and changes must be marked with a version change for each changed requirement

1.4 Literature analysis

To achieve the results on this deliverable, the team made use of several best-practices, mostly from:

- Software Engineering Institute/Carnegie-Mellon University’s CMMI® [CKS11]
- Project Management Institute’s Body of Knowledge PMBoK® [PMI13]
- European Space Agency’s Guide for Software Requirements Definition [ESA95]
- Papers and research in the area [DFi03, DFi07, IHo93]

1.5 Main Results and Outputs

The main results and outputs of this document are specified on the list of requirements stated in Annex C, which includes:

- The vf-OS functional requirements
- The vf-OS non-functional requirements
• A traceability list connecting requirements
2 Elicitation Activities

The main activity of the requirements development process is about eliciting the requirements. For this purpose, the methodology defined in Section 1.3 was used to obtain the specification of requirements, using activities such as user and technical meetings, direct and indirect observation, analysis of documentation, business, user needs, and the existing products on the market.

Several meetings were conducted to establish a consensus view and to point out the conflicts in defining and measuring functionalities. These meetings helped the customers and elicitation team to express their views on vf-OS. Additionally, the team developed questionnaires that were integrated on D8.1 [RPo17] and analysed the documented use-cases, stories, and metrics developed on D1.2 [JPA17] and D8.1.

2.1 Strategy

The strategy that was taken for the elicitation was defined during the kick-off meeting of the project, and included the definition of five “customer” pilots that were responsible for providing their vision of the vf-OS project, as can be seen in Figure 1.

Figure 1: vf-OS Requirements Elicitation Strategy
2.1.1 User Requirements

The elicitation of the User Requirements started on the second vf-OS project face-to-face plenary, where the elicitation team was split into groups of two people each. These groups were specifically created to achieve the most heterogeneous set of results. The groups paired experienced engineers with young ones, people from academia with people from industry, and so on. In addition to have a mixed expertise, the idea was to get people from multiple partners to become more familiar with each other, with the intention to help in future project Tasks. After reading the documentation for a specific pilot, the groups had a period of half an hour to elicit user requirements for that specific pilot. These were stated on post-it notes such as the ones shown in Figure 2. After that half-hour period, each group had an interview session with the pilot representative, creating an opportunity to validate the elicited requirements directly or correct them.

Figure 2: Post-its for the User Requirements Elicitation

The result of this exercise consisted of 130 user requirements. All the validated post-its were individually photographed and then put into a first user requirements spreadsheet. The spreadsheets were then validated once more by the “customers”, which had the opportunity to finally see a consolidated list (from all eliciting groups) of the needs that were raised over their pilot. They also had the opportunity to delete the needs that were duplicate or redundant, and to sort out which were contradicting or not following their vision. Finally, they were able to check on completeness, ie the chance to add their own user needs, as the final target was to ensure the list of user needs would be complete. The result set of validated user requirements listed 190 statements.

2.1.2 System Requirements

As the team showed interest and satisfaction in the applied methodology, a similar approach was taken during the system requirements elicitation phase and following phases. The approach was to have two teams doing brainstorming sessions individually for each pilot in order to be able to derive a set of system requirements from the above user requirements. For the selection of these teams, one partner was selected as the part of the team supporting the corresponding pilot (ie IKERLAN for Pilot1, KBZ for Pilot2 and LYON2 for Pilot3, CMS for the vf-OS Vision and ICE for the Generic Pilot), and another partner was selected for their experience in developing architectures and technological solutions, such as ASC and ALM. Each partner made their contribution, deriving a set of
system requirements from the list of user requirements for each pilot. The resulting set consisted of 430 system requirements, which were individually validated by the “customers”.

2.1.3 Software Requirements

Finally, from the list of user and system requirements, a new set of teams of two partners were asked to elicit the software requirements for each pilot. Again, the achieved objective was to walk through all pilots over all partners so that everybody would get better acquainted with the various target pilots. The set of statements were then validated by the “customers” and the resulting set raised 724 new software requirements.

The final accounts on pure elicitation of requirements and validation from customers had a total output of 1,344 requirements.

2.2 Guidelines on how to Perform the Requirements Elicitation

Despite the above strategy and resulting numbers of requirements collected, it was necessary to establish a set of rules and guidelines prior to performing the elicitation, to help the teams to perform their work in order to achieve the best results possible with the least effort.

2.2.1 Analysis of the Scope of Requirement Types

The analysis of a business needs to take into account multiple disciplines and views over that business, in order to be able to correctly capture all the needs that are required. These views on the business include [ESA95] the following major types of requirements:

- **Functional Requirements**: A function is a ‘defined objective or characteristic action of a system or component' and a functional requirement 'specifies a function that a system or system component must be able to perform'
- **Non-Functional Requirements**: Are all requirements that do not directly describe a functionality of the system, but instead describe concepts which are related to characteristics of that system. These include:
  - **Performance Requirements**: Numerical values for measurable variables used to define a function (eg rate, frequency, capacity, speed, or accuracy)
  - **Interface requirements**: Hardware, software, or database elements that the system (or system component) must interact or communicate with
  - **Operational requirements**: How the system will run (ie when it is to be operated) and how it will communicate with human operators. Operational requirements may describe physical aspects of the user interface. Descriptions of the dialogue, screen layouts, and command language styles are all types of operational requirements
  - **Resource requirements**: The upper limits on physical resources such as processing power, main memory, disk space etc. They may describe any requirement that the development or operational environment place upon the software. A resource requirement should state the facts about the resources, and not constrain how they are deployed
  - **Verification requirements**: Constrain the design of the product. They may do this by requiring features that facilitate verification of system functions, or by saying how the product is to be verified
• **Security requirements**: Requirement for securing the system against threats to confidentiality, integrity, and availability. They should describe the level and frequency of access allowed to authorised users of the software. If prevention against unauthorised use is required, the type of unauthorised user should be described

• **Portability requirements**: How easy it should be to move the software from one environment to another. Possible computer and operating systems, other than those of the target system, should be stated

• **Quality requirements**: The attributes of the software that make it fit for its purpose. The major quality attributes of reliability, maintainability, and safety should always be stated separately. Where appropriate, software quality attributes should be specified in measurable terms (ie with the use of metrics)

• **Software reliability**: The ability of a system or component to perform its required functions under stated conditions for a specified period of time

• **Safety requirements**: Specify any requirements to reduce the possibility of damage that can follow from software failure. Safety requirements may identify critical functions whose failure may be hazardous to people or property

2.2.2 Requirements Elicitation Best-Practices

In order for the vf-OS platform and components to be effectively designed, implemented, and measured, the requirements must be specific, unambiguous, and clear. To achieve that, the elicitation teams got numerous indications regarding how to act, including:

• Write simple, clear, and unambiguous statements
• Apply proper language according to the requirements level
• Testable (verification evidence should be stated)
• State what vf-OS should have and what it should NOT have
• State dependencies between requirements (different from traceability)
• Apply versioning
• State source (see section 1.3.2)
• Prioritise

A set of presentations were held right before each elicitation session, first in F2F meetings, then via e-mail. These presentations tended to show to the eliciting teams the purpose of each eliciting phase and how to do it, presenting “good” examples on “how to do it” and “bad” examples on “how not to do it”. Namely, a strong concern from the start was how to harmonise the structure of a requirements statement. As requirements tend to get complex and subject to interpretations, the idea was to define at least a standard structure for requirement statement, as can be seen in Figure 3.
Requirement Definition: Explicit Format

\[ \text{<actor>} \text{ <priority>} \text{ <action [AND action2...]>} \text{ <what>} \text{ <why>} \]

- **Actor**: Who is required to do the action
- **Priority**: Use words like “must” or “shall” instead of “may” or “will”
- **Action**: The action the Actor is required to do. It can consist of more than one closely related action (joined by “and”, but not by “or”).
- **What**: Context
- **Why**: Context

“The vf-OS system must define and validate the set of tools developed by the vf-OS consortium in order to support the specified and real-world scenarios and their exploitation potential.”

Figure 3: Explicit format definition and examples for vf-OS Requirements

This was the formal requirement statement structure taken into account on all requirements that were elicited.

2.3 Requirements Scope

The definition of the scope of the vf-OS framework will be determined by the list defined on Annex C. Hence, only the explicitly stated items are considered in-scope and everything else is considered out-of-scope.

Additionally, even from this specified list, the field “Requirement Specification Status” determines the scope analysis performed over the requirement specification. This can have the values:

- **Proposed**: States that someone proposed the requirement statement, but it hasn’t been validated by the customer
- **Rejected**: States that the customer did not agree with the statement
- **Validated**: States that the customer agrees and validated the statement

Thus, all requirements that do not have this field with the value “Validated” are considered out-of-scope.

Another definition that needed to be ensured was whether a requirement was feasible according to the vision of vf-OS and the effort, cost and other constraints of the project implementation itself. This was performed by creating a field named “Requirement Implementation Status”, which is the result of the analysis performed by the vf-OS team over the requirement. This field may have the values:

- **Proposed**: States that someone proposed the requirement for implementation, but it has not been yet validated by the vf-OS implementation team
- **Accepted**: States that the requirement was accepted for implementation, but it has not been implemented yet
- **Rejected**: States that although the customer wanted it, the vf-OS implementation team believes that the requirement is out of the scope of the project and it will not be implemented.
- **Being implemented**: States that the requirement was accepted and is already started to be implemented but is not fulfilled yet.
- **Implemented**: States that the requirement was accepted and was completely implemented, however it has not been yet validated by the customer.
- **Validated**: States that the customer agrees and validated with the implementation of the requirement.

Hence, a requirement is considered out of scope if the “Requirement Implementation Status” field has the value “Rejected”, although it is considered that the vf-OS team and the customer should review all “Rejected” requirements.

### 2.4 Analysis and Post-Processing

After the elicitation phase, an analysis was performed over the large set of requirements for the vf-OS framework. The idea was to analyse all requirements, looking for redundant information, erroneous statements, and contradictory requirements. To accomplish this task, the requirements were exported from a spreadsheet to a database.

Since the number of generated requirements was large, data processing and data mining tools was found to be useful for processing them. Hence, all the requirements from the different levels have been gathered into a dataset and the RapidMiner\(^1\) open source data science platform has been used to process the data, as shown in Figure 4. More specifically, RapidMiner has been used in the requirements consolidation process (see Section 1.3.4) to automatically identify duplicate requirements, identify actors, and find keywords by processing the requirements description. It is expected that this tool can be useful in later stages of the project as well.

![Figure 4: The RapidMiner data mining tool](https://rapidminer.com/)

---

\(^1\) [https://rapidminer.com/](https://rapidminer.com/)
3 Requirements Specification

The output of the requirements elicitation process after the post-processing and optimisation using RapidMiner (see Section 2.4) was again exported in the form of a spreadsheet, documented in Annex C. This document was split into two worksheets: one for the requirements concerning the vf-OS system and the other for requirements concerning the vApps that were needed for the pilots.

As stated before, there was a major concern of the team to make the requirements as standard as possible in their format. If section 2.2.2 targets the structure of each requirement statement (ie its text), this section describes the structure of the whole requirement record. That is, besides the requirement statement, each requirement comprises a set of fields (which can be seen on Annex C) which are needed for providing additional information to the requirement. Figure 5 shows the meaning of each of the fields (ie columns) that can be found on this spreadsheet.

**Requirements Table Layout**

|-----|-------------|------------------|-------------|---------|--------|----------------|------------------|------------|

Legend:

- **RID** Requirement Identifier:  RQ_<xxxx>  (eg RQ_0001)
  - xxxx: Sequential number with format “0001” onwards
- **Description** Specific and according to Section 2.2.2
- **[Non] Functional** Requirement is Functional or Non-Functional
- **Accountable** Who can edit the requirement?
- **Version** 1, 2, ... - Changed every time the requirement changes
- **Priority** 1 – Essential, 2 – Important, 3 – Nice-to-have, 4 – Cosmetic
- **Parent Req** RIDs for the backwards traceability

Figure 5: vf-OS Requirements spreadsheet structure

Additional columns were added for ensuring the completeness of the requirements life cycle, namely:

- **Priority**: Instead of considering a single Priority field, two fields were specified: One for the Customer Priority and another for the vf-OS Priority. These will be important to distinguish between what is important to the customer and what is important to the vf-OS project
• **Phase**: Which can be “User Requirements”, “System Requirements”, or “Software Requirements”
• **Pilot**: This field may include information about the applicability of the requirement regarding which of the five defined pilots effectively originated the requirement
• **Dependencies**: Determines if there are any dependencies for this requirement (e.g., other requirements connected or images or documents)
• **Actor**: The element of the vf-OS framework which the requirement targets
• **Req Filter**: Filter between the requirements targeted to the vf-OS platform and the ones for the vApps
• **Elicited By**: States the person who actually elicited the requirement, may be useful for clarifying questions about meaning of requirements
• **Requirement Specification Status**: Determines the status of the elicited requirement, which can be:
  • **Proposed**: Initial proposal for requirement
  • **Validated**: Requirement elicited proposal for functionality was validated by the “customer” (see Section 2.3)

• **Requirement Implementation Status**: Determines the status of the elicited requirement with respect to its implementation in the vf-OS system, which can be:
  • **Rejected**: Requirement will not be implemented
  • **Accepted**: Requirement will be considered for implementation
  • **Being Implemented**: Requirement is in the process of being implemented
  • **Implemented**: Requirement has been implemented
  • **Validated**: Requirement implemented functionality was validated by the “customer”
4 Conclusions and Future Work

This task was primarily to extract requirements from the end users in order for technical requirements to be realised. This document will be used as a primary input to the specifications defined in WP2 (Architecture), in order to help with determining the core feature set that needs to be delivered. In addition, it will clarify how the architecture will interface with identified external systems and meet the performance criteria of those interfaces. From the perspective of the forthcoming deliverables, this will help the definition of needs, of how functionalities are supposed to work, and whom to contact in case a developer does not understand a certain functionality.

Furthermore, there will be a significant input from these requirements to the use case validation deliverables in WP8, as the User Requirements provide a good indication of what functionality should be implemented. Subsequently, the Requirements Spreadsheet will be used to record the validation results against each of the requirements, particularly those that are defined as Essential for the Customer, which implies vf-OS must deliver the function. As such, the Requirements Spreadsheet that accompanies this document will act as a working document that will be updated through the life of the project and can provide focus on development, implementation, and validation efforts.
## Annex A: History

### Document History

<table>
<thead>
<tr>
<th>Versions</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1: First Draft Version</td>
<td>CMS: Carlos Coutinho, Adina Cretan, and Vasco Silva</td>
</tr>
<tr>
<td>V0.2: Second Draft Version</td>
<td>UPV: Francisco Fraile, Victor Anaya, and Héctor Navarro</td>
</tr>
<tr>
<td>V0.3: Third Draft Version</td>
<td>MASS: Jon Altube</td>
</tr>
<tr>
<td>V0.4: First review Version</td>
<td>CON: Fernando Monteiro</td>
</tr>
<tr>
<td>V0.5: Second review Version</td>
<td>KBZ: Raquel Melo</td>
</tr>
<tr>
<td>V0.6: Further Changes</td>
<td>ICE: Óscar Garcia</td>
</tr>
<tr>
<td>V1.0: Third review Version</td>
<td>IKERLAN: Óscar Saiz</td>
</tr>
<tr>
<td>V1.0.1: EU Version</td>
<td>APR: Benjamin Menghini</td>
</tr>
<tr>
<td></td>
<td>TARDY: Ibrahim Benali</td>
</tr>
<tr>
<td></td>
<td>UNINOVA: João Sarraipa and Sudeep Ghimire</td>
</tr>
<tr>
<td></td>
<td>ASC: Danny Pape and Tobias Hinz</td>
</tr>
<tr>
<td></td>
<td>ALM: Andries Stam</td>
</tr>
<tr>
<td></td>
<td>LYON2: Néjib Moalla</td>
</tr>
</tbody>
</table>
Annex B: References


[DFi03] Donald Firesmith (2003), Specifying Good Requirements, Software Engineering Institute


[ESA95] European Space Agency Board for Software Standardisation and Control (1995), Guide to the Software Requirements Definition Phase, ESA PSS-05-03 Issue 1 Revision 1

[IHo93] Ivy Hooks (1993), Writing Good Requirements, Proceedings of the Third International Symposium of the NCOSE - Volume 2

Annex C: List of vf-OS Requirements

The list of vf-OS Requirements is handed as a separate file, in the form of a spreadsheet located in the same folder as this document, with the name “D1.7-Requirements Specification-Annex C”, with the appropriate version. This separation was performed so that the document readers may experience an improved handling over the requirements, being able to perform actions over them such as sorting and filtering. Although the spreadsheet includes a single table for all vf-OS framework requirements, two sheets are presented in the document, one filtered with the requirements for the vf-OS Platform and the other filtered with the requirements for the proposed vApps to be developed in the scope of this project.