vf-OS: virtual factory Operating System

WP1: Vision, Scenario and Requirements

D1.3: Providers Scenarios Characterisation - vs: 1.0.1

Deliverable Lead and Editor: Danny Pape, Ascora

Contributing Partners: ASC, ALM, CMS, ICE, IKER

Date: 2017-02

Dissemination: Public

Status: EU Approved

Short Abstract

This document provides a high-level characterisation and classification of vf-OS technical providers' scenarios. It complements D1.2 User Scenarios Characterisation, which focuses on the user scenarios. Particular attention is given to technology providers and software developers, their functional needs, technical concerns and business models.

Grant Agreement: 723710

Document Status

Deliverable Lead	Danny Pape, Ascora
Internal Reviewer 1	Jean-Phillippe, MASS
Internal Reviewer 2	Fernando Monteiro, CON
Internal Reviewer 3	Stuart Campbell, ICE
Type	Deliverable
Work Package	WP1: Vision, Scenario and Requirements
ID	D1.3: Providers Scenarios Characterisation
Due Date	2017-01
Delivery Date	2017-02
Status	For EU Approval

History

See Annex A.

Status

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

Further Information

www.vf-OS.eu

Disclaimer

Project Partners:

Executive Summary

This document,"D1.3 Providers Scenarios Characterisation", aims to strengthen a common understanding of the different technology providers in vf-OS. The parallel deliverables, "D1.1 Vision Consensus" and "D1.2 User Scenarios Characterisation" complement this deliverable. The common understanding for vf-OS, and especially the combination of this deliverable and "D1.2 User Scenarios Characterisation", completes the description of all technology providers and the point of view of the users.

This deliverable consists of several sections that define and describe the different technology provider roles and their general activities. Based on these general activities of the technology provider generic scenarios are described, which brings the activities into a vf-OS context. These then lead to vf-OS specific scenario descriptions, where the general activity scenarios are specialised and described in further steps. In addition, a connection to the user scenarios from "D1.2 User Scenarios Characterisation" was taken, to map the user requirements with the intended work of the technology providers. At the end, the revenue streams are introduced to reward the providers for their work. The document is, thus, organised around the following pillars:

- **Provider Definition:** Technology providers are essential for vf-OS to extend and individualise the vf-Platform (vf-P). Therefore, vf-OS supports different provider roles for different use cases. The main roles are Software Developers, External Service Providers, Manufacturing and Logistics Solution Providers, and Platform Providers. Provider roles have, of course, overlapping activities.
- **Provider Scenarios:** Several provider orientated scenarios are established to help extract important requirements vf-OS has to fulfil from the provider perspective. This includes the different stages of providers such as Preparation, Implementation, Publishing, and Marketing/Management.
- **Adaption to User Scenarios:** A liaison with both providers and users are essential to build a functional ecosystem in a multisided market. This is as true for vf-OS as it is for iOS from Apple or the PlayStation Network from Sony. Derived from the user scenarios different requirements must be considered in order to develop a satisfactory vf-OS.
- **Revenue Streams for Providers:** Revenue models are essential to support providers for their vf-OS Assets. Attractive revenue models lead to a higher growth of high quality vf-OS Assets in the vf-Store. Additionally, the applicability of the revenue models in the real world are checked by adapting them to the user scenario pilot apps from 'D1.2 User Scenarios Characterisation'.

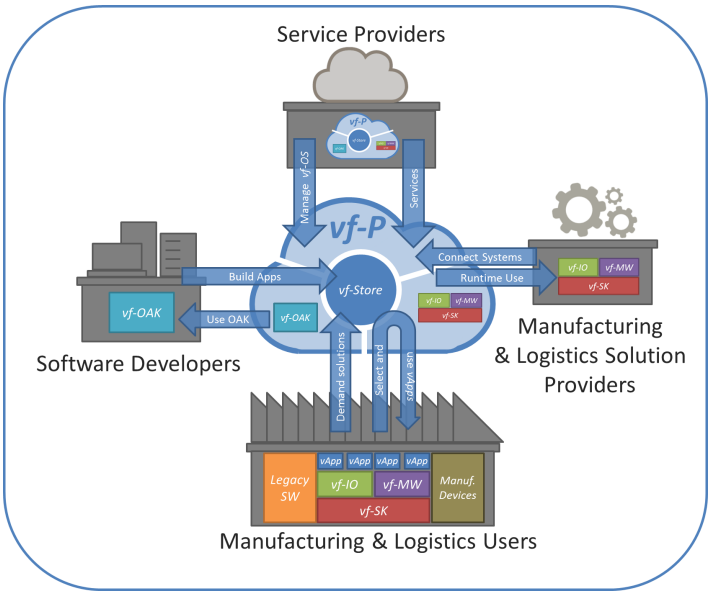
Table of Contents

Document Status II
History II
Status II
Further Information II
Disclaimer II
Project Partners: III
Executive Summary IV
Table of Contents V
0 Introduction 1
0.1 vf-OS Project Overview 1
0.2 Deliverable Purpose and Scope 2
0.3 Target Audience 2
0.4 Deliverable Context 2
0.5 Document Structure 2
0.6 Document Status 3
0.7 Document Dependencies 3
0.8 Glossary and Abbreviations 3
0.9 External Annexes and Supporting Documents 3
0.10 Reading Notes 3
1 Provider Definition 4
1.1 Provider Characterisation 4
1.2 Provider Activities and Artefacts 5
1.3 Vf-OS Tools 7
1.3.1 vf-OS Provider Interfaces 7
1.3.2 vf-OS Development Tools 7
1.3.3 vf-OS Support 8
1.3.4 vf-OS Asset Management 8
2 Provider Scenarios 10
2.1 Generic Provider Scenarios 10
2.1.1 Common Provider Needs 10
2.1.2 Platform Providers 11
2.1.3 Software Developers 13
2.1.4 External Service Providers 14
2.1.5 Manufacturing and Logistics Solution Providers 16
2.2 vf-OS Provider Scenarios 17
2.2.1 Platform Providers 17
2.2.2 Software Developers 19
2.2.3 External Service Providers 21
2.2.4 Manufacturing and Logistics Solution Providers 24
3 Adaption to User Scenarios 27
3.1 User Scenario 1: Manufacturing and Logistics – Automation 27
3.1.1 Pilot vApp: vfFailurePrevention (P1.1) 27
3.1.2 Pilot vApp: vfFailureManager (P1.2) 28
3.1.3 Pilot vApp: vfStockPolicies (P1.3) 28
3.1.4 Pilot vApp: vfProductionFollowUp (P1.4) 29
3.1.5 Pilot vApp: vfMaintenanceCalendar (P1.5) 29

3.1.6 User Scenario Analysis....30
3.2 User Scenario 2: Construction / Industrialisation....30
3.2.1 Pilot vApp: vfDocumentPortal (P2.1)....31
3.2.2 Pilot vApp: vfSteelValidation (P2.2)....31
3.2.3 Pilot vApp: vfOnSiteManager (P2.3)....32
3.2.4 Pilot vApp: vfConcreteFeedback (P2.4)....33
3.2.5 Pilot vApp: vfProductValidation (P2.5)....33
3.2.6 User Scenario Analysis....34
3.3 User Scenario 3: Manufacturing Assembly: Collaboration....34
3.3.1 Pilot vApp: vfCollaborationAnalyser (P3.1)....35
3.3.2 Pilot vApp: vfIndusEnabler (P3.2)....35
3.3.3 Pilot vApp: vfProductionPlanner (P3.3)....36
3.3.4 Pilot vApp: vfQualityInsurance (P3.4)....36
3.3.5 Pilot vApp: vfProductionTracker (P3.5)....37
3.3.6 User Scenario Analysis....38
3.4 User Scenarios Conclusions....38
4 Revenue Streams for Providers....42
4.1 Revenue Stream Models....42
4.1.1 Subscription....44
4.1.2 Transaction-based (pay per use)....45
4.1.3 License....45
4.1.4 Commission....46
4.2 Pilot Providers Revenue Stream Scenarios....46
4.2.1 Pilot 1: Manufacturing and Logistics – Automation....47
4.2.2 Pilot 2: Construction – Industrialisation....49
4.2.3 Pilot 3: Manufacturing Assembly: Collaboration....50
5 Conclusion....52
Annex A: History....53

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 Logistic Users** which will explore the marketplace for already created solutions, ready to be run on the vf-OS
- **Manufacturing and Logistics Solutions 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. 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 manufacturing stages 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 component of the system software in a real factory system where all factory application programs will run.

0.2 Deliverable Purpose and Scope

The purpose of this document “D1.3 Providers Scenarios Characterisation” is to define and describe provider roles and scenarios, which enables the extension of vf-OS to individualise the environment regarding specific customer needs. For those roles, interfaces must be created to allow the integration of software modules into the vf-OS environment. To achieve this goal, this document provides information about provider roles and the connection to user scenarios described in “D1.2 User Scenario Characterisation” to create coherent scenarios aligned to targeted app solutions from the different vf-OS pilots. Furthermore, this document describes what vf-OS needs to provide in order to enable a comfortable development and smooth software integration. Finally, define optional income streams for different provider roles to reward the developer for their efforts. In short, this deliverable will close the gap of scenarios to individualise a vf-OS environment.

0.3 Target Audience

Whilst the document is primarily aimed at the project partners, this public deliverable can be useful for the wider scientific and industrial community. This includes other publicly funded projects, which may be interested in collaboration activities.

0.4 Deliverable Context

The strengths of vf-OS will be the fulfilment of use cases mostly through individual software artefacts (eg Apps) that extend vf-OS. Therefore, it is necessary to create scenarios, where those use cases are described, and which this document can take advantage of. Its relationship to other documents is as follows:

- **D1.1 Vision Consensus:** The predefined generic scenarios from this deliverable act as a foundation for most scenarios treated in this deliverable
- **D1.2 User Scenarios Characterisation:** Provides the counterpart of provider scenarios, namely the user scenarios, where requirements can be extracted that affect the providers, including transitive dependent requirements

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1: Provider Definition:** Includes all information about different technology provider roles and their general activities

- **Section 2: Provider Scenarios:** This section describes at first generic scenarios of the different technology providers, and second specific vf-OS scenarios of all technology providers
- **Section 3: Adaption to User Scenarios:** Defines and describes the technology provider requirements extracted from the user scenarios of 'D1.2 User Scenario Characterisation'; These defined requirements bridge the gap of providing and consuming vf-OS Assets
- **Section 4: Revenue Streams for Providers:** Defines specific business models only for providers; It describes the possibilities to reward providers for their efforts
- **Section 5: Conclusions:** Provides the conclusions of the document

Annexes:

- **Annex A:** Document History
- **Annex B:** Reference

0.6 Document Status

This document is listed in the Description of Action as "public" since it provides general information about the technology provider and related scenarios of vf-OS and can therefore be used by external parties in order to receive insight into the project activities.

0.7 Document Dependencies

This document provides additional information, which will be reused in "D1.5 Requirements Specifications" and "D2.1 Global Architecture Definition".

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

Annexes and Supporting Documents:

None

0.10 Reading Notes

None

1 Provider Definition

The following section builds a theoretical foundation for the deliverable. To achieve this, first the different types of providers in vf-OS are characterised in Section 1.1. Then the general development activities of these providers are described in Section 1.2. And finally, in Section 1.3 the tools and components provided by vf-OS are introduced.

1.1 Provider Characterisation

vf-OS aims to bring together two different groups:

- Users: Manufacturing and Logistics Users
- Technology Providers: Software Developers, External Service Providers, Manufacturing/ Logistics Solution Providers, and Platform Providers

The focus of this deliverable is on the technology providers (hereinafter “provider”), their ways of working, their needs, and their business models. The findings of this deliverable are based on the vision consensus laid down in “D1.1”, and taking into consideration the user scenarios as specified in D1.2. The different groups of providers recognised in vf-OS and derived from the image above (Figure 1) are the following:

- **Platform Providers:** Have to integrate, install, and manage the vf-OS Platform, possibly using the customer’s IT infrastructure and also connecting it to current/legacy software/hardware
- **Software Developers:** Can access a new and high-growth potential market for vf-OS Manufacturing Applications (vApps). These applications will be developed using an Open Applications Development Kit (vf-OAK) to quickly build applications running on the vf-OS System Kernel (vf-SK) and using Devices Drivers, API Connectors, and the vf-OS Middleware (vf-MW)
- **Manufacturing and Logistics Solution Providers (MLSP):** Expose their ICT interfaces and manufacturing connections to the Manufacturing Applications. They will also be able to contribute to the development of Manufacturing Applications that may be added and commercialised in the vf-Store
- **External Service Providers (ESP):** Will provide external services (hosting, storage, connected cloud services, etc.) including those based on developed solutions. These services can be provided by vf-OS partners or third parties. The Platform Providers are a specialised form of ESP’s

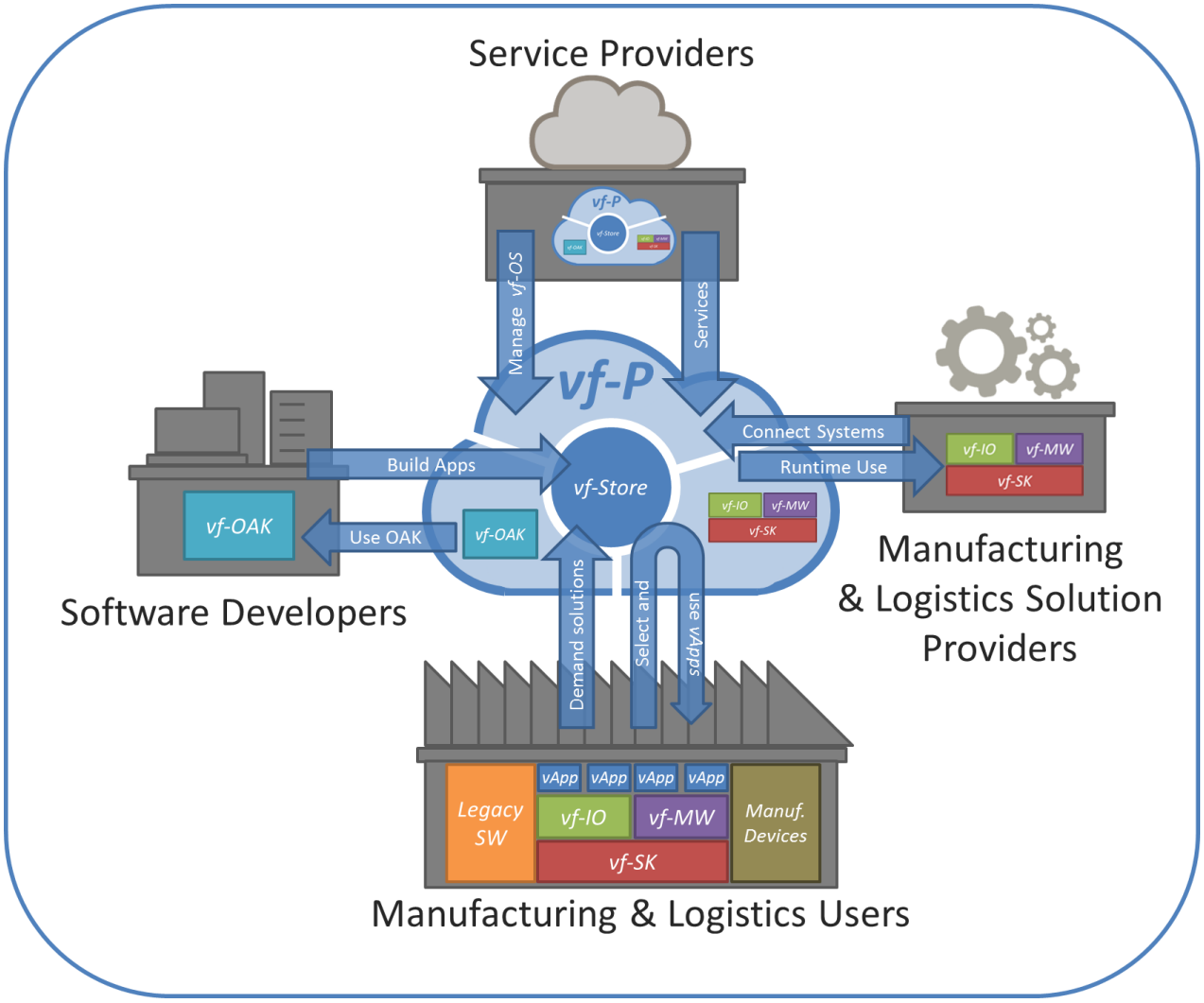


Figure 1: vf-OS Interaction Model

1.2 Provider Activities and Artefacts

To provide any vf-OS Asset, the providers have to develop them. Therefore, this section describes general development activities each provider has to deal with in vf-OS.

The development activities start with software planning and end with software provisioning. The common task of the vf-OS provider is to extend and support vf-OS with vf-OS Assets (new Manufacturing Applications, adapters, and services) and connections to legacy systems using the various interfaces, software development, and support tools provided by vf-OS.

Figure 2 describes default activities of providers who develop and integrate vf-OS Assets. These activities represent a full development lifecycle ranging from planning to publishing, through to support. Furthermore, this figure provides a foundation for the provider scenarios in Section 2.

Activity	Description

Requirement Analysis	The goal of the requirement analysis is to determine the needs or conditions to meet for newly developed assets. This goal must be achieved by considering the possibly conflicting requirements of the various stakeholders.
Design and Implementation	The design and implementation phase takes the results of the requirement analysis and creates specifications for a new asset. Based on these specifications the software is built.
Documentation	The target audience of a documentation of a software project can vary. On the one hand there are developers who get information on how the software itself is built (documentation of code, algorithms, interfaces, etc.). On the other hand, there can be a deep interest in the given functionalities by the marketing on how to market the product.
Testing	The goal of testing is to ensure the fulfilment of the defined specifications. This is performed by executing a program or application with the intent of finding possible errors and to verify that the software product is fit for use.
Deployment	Software deployment is the process of bringing software into its designated working environment and thus making it available for usage.
System Integration	System integration is the act to bring different components together to form a working system. It is based on physical connections and installation of different software components.
Maintenance and Error fixing	After the development phase, there is always a need for later changes in software. This can be either caused by errors in the software, improvement of performance, or new features.
Providing Updates	New software versions are the direct result of maintenance and error fixing. Updated software versions can be either published on a regular basis or in special circumstances.
Hosting	Hosting in relationship to software means to provide a platform that enables end users to use their system with as less individual intervention as possible. In recent years Software as a Service (SaaS) is a well-known example for this approach. The host cares for the environment and offers a working system to the end user.
Release Management	Release management is the process of managing, planning, scheduling, and controlling software build through different stages and environments – it includes testing and deploying software releases.
Configuration Management	Configuration management is a process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life cycle.
Support	End users may need assistance in using a system. The goal of support is to eliminate any occurring problems and to offer further assistance.
Security Auditing	Security auditing is a way of testing software with focus on security aspects. It not only contains the testing of the software itself but also interviewing staff, performing security vulnerability scans, reviewing application and operating system access controls, and analysing physical access to the systems.

Figure 2: Provider Activities

All the activities above relate directly or indirectly to various software and hardware artefacts. In the context of vf-OS, the following vf-OS Assets are distinguished:

- **vApps:** End user solutions, running on desktop or mobile clients, cloud or on-site servers, or a combination thereof, possibly connected to / dependent on vf-IO modules and services
- **Enablers:** Software Artefacts that are located in the vf-OS IO Toolkit and are aligned to fulfil specific tasks, which help to solve specific problems; In the case of vf-OS, they are not part of vApps, but enables to extend vf-OS itself, eg connecting to information systems through API Connectors and enables those information systems to exchange data with vf-OS.

- **External services:** External software services are hosted services outside of vf-OS but connected to it through a provided API. Those services are often running in the cloud (but not necessarily). Well known service functionalities are hosting or providing computing power

1.3 Vf-OS Tools

vf-OS provides different means to support providers to make vf-OS more customisable and therefore individual. On the one hand, it provides interfaces to integrate external software artefacts into the vf-P and on the other hand developer tools and support means to facilitate and accelerate the development process of vf-OS Assets. These are described in the following subsections:

- vf-OS Provider Interfaces
- vf-OS Development Tools
- vf-OS Support
- vf-OS Asset Management

1.3.1 vf-OS Provider Interfaces

vf-OS Provider Interfaces are access points for providers to register their vf-OS Assets (ie vApps, Enablers, External Services) in vf-OS. Those provider interfaces are differentiated between the different provider roles. Software Developers for vApps register their vf-OS Assets in the vf-Store, whereas External Service Providers register their services by means of the registration methods offered by the External Service Provision component or through Device Driver/API Connectors in the vf-IO. The interfaces have to fulfil the following provider related tasks:

- **Software Developers:** the vf-Store provides an interface for Software Developers to publish vApps directly from the vf-Studio. After Software Developers finish their development the developers can use the functionality through the vf-Studio to publish a vApp. Before the vApp is accessible in the vf-Store, an approval process carried out by administrators must be passed through.
- **External Service Providers:** vf-OS provides an External Service Provision component for the External Service Providers, which is able to help the integration of external services in vf-OS. The External Service Provision component provides standardised ways to access services as well as generic locator and registration interfaces. This facilitates the integration and usage of external services in vApps.
- **Manufacturing and Logistics Solution Providers**: vf-OS provides means to integrate custom Device Drivers and API Connectors to connect manufacturing enablers. The Device Drivers are responsible for collecting data from, or sending data / commands to physical peripheral resources (such as machines). Whereas the API Connectors are made to receive data from or send data to virtual peripheral resources such as ERPs, CRMs, etc.

1.3.2 vf-OS Development Tools

Development tools provided by vf-OS itself form a strong support pillar for developers to facilitate and accelerate the development process. The providers, mainly the Software Developers, benefit from the means provided by vf-OS. Development tools include the vf-Studio, vf-OAK, and Front End Environment.

The vf-Studio is tailored exactly to the needs of providers to build high-quality vf-OS Assets. Only optimised development tools for vf-OS are provided to create new vf-OS Assets. Through the vf-Studio the providers get access to the vf-OAK and a set of default front-end elements with integrated programming logic from the Front End Environment. The vf-OAK acts as an independent SDK, comparable to the Android Java SDK. It provides all functionalities, which vf-OS offers. To create well designed UIs and default behaviour templates for different scenarios (eg error reports, registration/login, request new passwords), the Front End Environment is designed to help the developers so that they can focus on the real vApp functionalities.

To integrate existing external services, vf-OS provides the Service Provision Framework that includes all information and interface descriptions of external services. These external services then can be easily used by Software Developers to integrate them in their vApps.

1.3.3 vf-OS Support

As a further support pillar, vf-OS provides different support capabilities to providers. As a first basic line of support, an API documentation is provided, which helps developers to use the provided functionality correctly. The API documentation describes the purpose of functionalities and also indicates types of expected parameters, return values, etc. As a second line of basic support, the Developer Engagement Hub is provided, where developers can ask questions regarding specific issues in the development process and to get in contact with other developers in order to exchange information (similar to StackOverflow). This approach aims to allow Software Developers to interact at a central point and build a vf-OS developer community. It is expected that the most discussions in the Developer Engagement Hub are about alternative solutions and experience reports. As a side effect, errors can be found in the vf-Studio, vf-OAK, vf-Front End Environment, or System Dashboard that then can be reported.

Finally, the Education and Training forum of vf-OS will guide developers with manuals, tutorials in form of different media types, such as text, images, and videos. This approach aims to avoid time consuming searches in the internet for solutions, and to present the most important tutorials at the source directly.

1.3.4 vf-OS Asset Management

The outcomes of the various providers will be available in the vf-Store. The vf-Store provides several options to add new vf-OS Assets to the collection. On the one hand, through the vf-Studio, which makes use of an API that enables an upload of an asset and on the other hand via a provider user interface of the vf-Store that provides UI elements to select an asset and upload it manually.

In addition to the executables, a manifest file has to be uploaded that contains all Meta information which is then shown in the vf-Store to describe the asset. This can also be performed through the vf-Store by uploading a manifest file with all necessary information, but also via the vf-Studio in terms of a predefined form with the detection of optional and mandatory content.

In contrast with the vf-OS Assets produced by Software Developers and Manufacturing and Logistics Solution Providers external services are located externally and not under vf-OS control. Thus an External Service Provision Framework within vf-OS will provide tools to manage the use of these external services. The Service Provision Framework offers functionality to/from those services for registration, lookup, and access of those external services, as well as dedicated APIs and method wrappers for often used services or

multiple commonly used services with comparable functionality. However, it does not itself provide the services. It also means that external services need to be compatible with (ie apply) the framework to be used/recognised in vf-OS.

2 Provider Scenarios

In contrast to “D1.1 User Scenarios Characterisation”, the various scenarios of each provider are written from the provider’s point of view. It articulates the way in which the various types of providers use, extend, and support vf-OS.

This section distinguishes between generic scenarios in Section 2.1 and scenario descriptions in Section 2.2. Generic scenarios are about default activities of provider businesses and extracted needs that should be covered in vf-OS. Scenario descriptions in Section 2.2 are directly related to vf-OS and the different vf-OS provider roles.

2.1 Generic Provider Scenarios

This section describes generic provider scenarios where common activities (outside of vf-OS) of different provider roles are described. Section 0 summarises common provider needs that are required by each provider role in vf-OS, excluding Platform Providers (Platform Provider needs are separately described in Section 0). After that, generic scenarios for each provider role are described. The description of those generic scenarios leads to a table of specific provider needs (inside vf-OS) of the different provider roles (Software Developer, External Service Provider, Manufacturing and Logistics Solution Provider, and Platform Provider).

2.1.1 Common Provider Needs

Overlapping and general needs for all provider roles, except Platform Providers, are listed in the following figure. Since the Platform Providers do not provide additional vf-OS Assets to vf-OS itself, it does not derive the following needs.

Activity	Description
Standardised requirements methodologies	Get defined requirements from a central location – from customers in case of a demanded application announcement. As an initiative development, requirements have to be written down at the same location, but should be highlighted as such.
Notification for requirement updates	Notifying the developer when requirements have been changed to avoid misunderstandings.
Download options for development tools	Providing development resources of vf-OS including a testing environment as a download bundle and individual downloadable components.
Testing	A testing environment to develop and test vf-OS Assets under real vf-OS circumstances.
SDK	A package which enables development of vf-OS Assets and includes all public vf-OS functionalities.
Documentations and tutorials	Providing support for developers in the vf-OS development environment. This includes templates, first steps tutorials, documentations, etc.
IDE tools	Providing an IDE to develop vf-OS Assets comfortably, such as code-completion, different developer views. Optimally with capabilities to extend the IDE in form of add-ons.
Manifest editor	An editor to create or update descriptions for vf-OS Assets, such as vApps, services, enablers, etc.
Usage statistics and billing	A dashboard and API that captures and provides information about the usage of published vf-OS Assets (calls, processing time, etc) and which can be used for eg client billing.

Error dashboard	A portal to view application errors in detail to be able to provide an update that fixes those errors.
Software quality inspection	Submitted vf-OS Assets should be reviewed by an administrative instance to check the asset for security, quality, usability, etc.
Market research	To allow users and developers to browse existing vf-OS Assets available on the vf-Store to understand their feature, use, complexity, etc.
Security framework	Any developed asset needs to be conformant with an appropriate security framework identified by vf-OS. This should cover content control, privacy, access control, etc.
Developer training	Besides documentation and tutorials, training for developers is provided by vf-OS to guarantee a high-quality development of vf-OS Assets.
Marketing	Ability to upload commercial details and further information about the asset.
Hosting capabilities	A platform where the vf-OS Assets can be hosted.

Figure 3: Common Provider Needs

2.1.2 Platform Providers

Platform Providers carry out all activities related to the actual provisioning of a software platform. In the case of vf-OS it acts as a holistic vf-OS environment, which provides an infrastructure for vf-OS components and Assets that enables them to run on and connect to the vf-OS. Platform Providers are not involved in the actual software development, but merely in their deployment and configuration. The needs of Platform Providers therefore primarily focus on these aspects.

The following figure indicates the Platform Provider's needs:

Activity	Description
Installation and uninstallation	All runtime components of vf-OS must provide an easy-to-use installation script or an executable installer. With this, the vf-P is able to provide automated installation procedures on a server or VMs, ie without any human intervention (configuration might need human intervention, though).
Multitenancy	All components, which provide a web-based front-end to end users or a generic API for use by vf-OS Assets need to be usable by multiple tenants at the same time.
Versioning and dependencies	All runtime components need to apply clear versioning standards and provide a standardised way to determine the version of the components they depend on.
Startable and stoppable	All runtime components need to be startable in an automated manner. All runtime components need to be stoppable in an automated manner without introducing (state) faults.
Authorisation and authentication	All runtime components need to be compatible with the authorisation and authentication mechanisms provided by the vf-P. Furthermore vf-OS accessibility is limited via a mandatory single sign-on.
Logging and accounting	All runtime components need to apply standardised non-functional logging mechanisms, but are free to use proprietary functional logging and accounting mechanisms.
Network requirements	All runtime components need to be compatible with the network interface demands of common virtualisation technology.

Figure 4: Platform Provider Needs

Platform Providers in vf-OS may host and provision the vf-P in several different ways, depending on the needs of their customers. For example, in a minimal setup, a Platform

Provider configures a single platform server to host a single runtime instance of the vf-P, which provides access to one or more of the following components:

- **A runtime instance of the System Dashboard**, for providing information about the system to vf-OS Assets and vf-OS components. This also requires that the server runs an instance of the vf-SK (if needed configured with modules from vf-IO and/or vf-MW)
- **Runtime instances of various other services developed within vf-OS**, such as Process Execution, Data Analytics, runtime access libraries from the External Service Provision component, Process Designer, Data Mapping, runtime subcomponents of the Front-End Environment component, Data Storage and Developer Engagement Hub

In comparison to the minimal instance, a large-scale setup provides the following components additionally:

- **A runtime instance of the vf-Store**, for publishing and buying vf-OS Assets, as well as for downloading binaries / sources from vf-OS components (vf-OAK, vf-P, vf-SK, vf-Store, vf-MW, vf-IO, all other services). The vf-Store is not only local but also federated and thus available from the outside.
- **Access to External Services** for integrating additional resources that are applicable for vf-OS, eg hosting or computation services
- **Configuration** of the system itself and also of the installed vf-OS Assets. It is conceived to have the same approach as modern operating systems, eg activate or deactivate vf-OS Assets, uninstall vf-OS Assets, clear the cache of a vf-OS Asset, or reset to default settings.
- **Administration and Monitoring** of a server pool, in which the servers act as hosts for virtual machines running one or more vf-OS components, tools and/or vf-OS Assets.

Platform Providers are completely free in choosing which of the components mentioned above they provide. For example, they could simply host a vf-P with a runtime instance of the vf-Store but without any runtime environment for vf-OS Assets.

The primary tasks of Platform Providers are described in the following.

- **Platform Hosting and Provisioning:** This includes the management of server pools the platform is running on. Part of this is the setup and configuration process based on the customer's needs. As modern software environments and platforms are based on virtual systems, the creation and hosting of virtual machines (VM) is also implemented in the vf-P.
- **vf-OS Asset Configuration and Provisioning:** Once the set-up of the platform itself is completed, the Platform Provider has to provide the requested software framework. This includes the deployment of runtime components, tools and/or applications on servers or VMs, and the realisation and maintenance of (preconfigured) VM images.
- **Maintenance:** It is indispensable to have secure and reliable authentication and authorisation methods for the vf-OS system. If the maintenance user is authenticated and authorised, the user is able to manage the server or virtual machine. These hosts can be started and stopped, eg for update reasons. Furthermore, a migration of vf-OS Assets, can be performed by an administrator and with help of the System Dashboard analysed to maintain all resources.

- **Monitoring:** Monitoring of any system is extremely important, because of issue solving and runtime improvements. It is needed to have a continuous overview of the system capacities and performance indicators. Furthermore, utilisation management must be integrated to be able to react quickly for disturbances.
- **Support:** Support will be needed to help users and providers with setup and configuration problems as well as runtime issues, such as freezing or non-responsive VMs. Therefore, a user is able to send a support ticket to an administrator, who takes care of the issue. To handle those support requests the administrator can perform several administrative actions with the help of system functionalities. These administrative actions are further described in Section 2.2.1.

2.1.3 Software Developers

As a developer, there is a need make the development process of software as easy as possible. At first the elicitation of requirements is a process worked out together with a customer. In case a development request is initiated by a client, it would be helpful if all requirements are formulated and accessible at a default location (eg website, cloud storage, etc). Furthermore, requirement updates should be highlighted and communicated to the developer. In the case of initiation by the developer, the developer must decide itself how to handle self-made requirements. But optimally the same methodology is applicable as from demanded requirements.

There are several steps described which are necessary to provide vf-OS Assets:

- **Development Tools Installation:** Software Developers have to familiarise themselves with the development environment. Developers expect to download all necessary resources which are needed to develop an application in a bundle. In general, such a bundle includes an IDE, the SDK itself with default UI elements, a documentation to support the developer during the implementation, and finally a testing environment (eg emulator) to test the created software artefact in a simulated environment. Furthermore, for experts, all resources should be downloadable individually.
- **Develop New Asset:** After the installation and configuration of all required components is completed, a tutorial with first steps (ie “Hello World”-program) helps developers enormously to get familiar with the development environment. Furthermore, additional tutorials, such as how to use the SDK, or how to integrate UI elements are helpful although not mandatory. Default properties of an IDE, such as code-completion, add-ons, or customisable views will help to reach the fulfilment of comfort for developers. As most developers avoid unnecessary efforts, not only code-completion is expected but also an application template to avoid developing a software artefact from scratch. This template should include default behaviour of the environment (eg an error report is sent to the backend automatically if an error occurs). During the implementation, it is common that developers are searching for solutions in the internet with the help of search engines (eg Google, Bing, etc.), relevant forums, and portals (eg StackOverflow). Therefore, a special need for application developers is to find a place for collaboration support in vf-OS. The exchange of information is incredibly important to build a well running community of developers.
- **Software Testing**: Before finally publishing an application in an app store, it should be tested in a typical use case environment. For this, an emulator (similar to Android emulators in Android Studio) should be provided. This emulator should always be in

line with the current version of the operating system itself. This means, that the emulator has to be updated along with it as such.

- **Software Deployment:** After finalising the development and testing an application, the developer needs to submit the final version of the software artefact to the app store. This includes the setup binary and in addition a manifest file, which contains Meta information about the vf-OS Asset. This process should hide the complexity and make it as easy as possible for the developer. The deployment process consists of the following activities:
  - uploading the vf-OS Asset
  - going through the validation process
  - validation notification if the uploaded vf-OS Asset is accepted or rejected

  In case of a rejected vf-OS Asset, it must be improved regarding the attached reasons from the validation notification. In case of an accepted vf-OS Asset deployment, it is now accessible in the app store for public use.
- **Providing Updates:** As soon as a software artefact is published and ready to use, the developer wants to have information about the usage. This includes error reporting, monetisation, popularity, and ratings of the software artefact. To have this all in one glance, the developer needs a portal to get all information. Based on this information the developer is able to provide updates to face errors, requested functionalities, and other feature requests.

2.1.4 External Service Providers

External Service Providers provide services to the vf-OS community which are either not available on the vf-P itself or typically may offer enhancements to the core services. Those services can easily be integrated into vf-OS to provide additional functionalities, such as hosting, computing, etc. Healthy competition is allowable (since the vf-P will also take a commission on sold services) and the enhancement may not only be technical but can be on service quality or price. Such services may relate to:

- **vf-Store** to be visible to the Software Developers and Manufacturing Users (detailed below) so they can consider when building new applications or enhancing/updating existing ones with new features
- **Software developers** to become part of the developer's applications, which are then sold to users. For example, a Software Developer may need a service not offered intrinsically on the vf-P and which is not core for them to develop themselves – this could include advanced hosting, computation, or algorithm libraries. Typically, the ESP would then either ask from the Software Developer one-off license fees or pay-per-use revenues as a percentage share of the end user's revenue
- **Manufacturing Users** to become part of the user applications which they utilise themselves. The same comment regarding rational and pricing types are valid as with Software Developers except there would be no application intermediary and since the service would be for restricted use in one user companies only, the prices would be considerably lower
- **vf-P** to become part of the platform. For example, the platform may wish to extend its set of core services to Software Developers and end users. Typically, the ESP would receive similar license fees to those mentioned for developers

There are several steps described, which are necessary to provide external services:

- **Assess vf-OS Environment:** The first step for the ESP is to understand the vf-OS environment itself: Technically, commercially, operationally, and legally. This would include registering the ESP profile, as a company and individual, on the system and exploring its usage both as an ESP but also from the perspective of other stakeholders such as Software Developers and users. As a developing organisation, the technical part will be particularly important so the interfaces and technical nuances of the platform are known. But also, since the goal would be to ultimately sell services to users, it will be necessary to understand the commercial/legal environment including pricing options. Naturally the ESP will also wish to see, as with any marketplace, what the competitive application range is in terms of number, features, and pricing. During this, the ESP would invariably find it beneficial to ask questions and/or view answers on any developer forum and similarly for any user forum so they can understand the system better and fine tune their offering to maximise their customers/revenues.
- **Create External Services:** Once the environment is understood and the ESP feels attracted to it and competent to develop in it, they then need to put themselves in a position to produce services, which can be used by others. This use could be by the Software Developers, users, or the platform as mentioned above. The first step will be to install the vf-OS development environment including both features for designing, developing, and testing applications/services and a developer edition of the vf-P to develop/test against. Once developers feel familiar, and armed with a suitable idea/design, the next step will be to build their idea in earnest. This could involve building a service from scratch using the development environment or simply taking an existing service and framing it in the context of vf-OS expectations. However, at the very least, the service must be compatible with the vf-OS External Service Provision component (which in turn will be based on the vf-IO framework) and any security (eg access control) mechanism dictated by vf-OS. After the service is completed it will naturally be tested against the features of the installed developer platform environment.
- **Add Services on vf-Store:** The completed and tested service then needs to be advertised and made available to others. This would be performed by the vf-Store and first it is necessary to upgrade the ESP account into a seller account on the vf-Store. This would typically involve the provision of additional commercial information, formal administration details, payment mechanisms, etc. Once (newly) registered, they may be requested (dependent on the platform) to take a knowledge and skills test to ensure that they have understood the vf-P and so helping ensure the confidence of any service they develop. Once passed, a service can be added but first it must be suitably described both technically and commercially according to atomised templates provided by the platform. Once described, the service is added to the vf-Store, although at this stage it is still invisible to users since services (especially for new users) will be tested by the vf-P operator using a similar test suite to the local one mentioned for the ESP.
- **Make Available Service:** In parallel with the Platform Provider service approval, the ESP will need to make the service operational so that when it is purchased/used the service is ready and functional. The deployment will most likely not be on the platform but in the environment of the ESP (even if this environment itself is hosted by another party). They will also need to make sure that access/usage of the service is communicated back to the platform (via suitable APIs) since this will be used for logging, security control, and billing purposes. Of course all of this will need to be tested as well. At some point as the service is marketed, Software Developers or

Service Providers will request information about it to explore potential opportunities and thus the ESP will need to communicate with those parties which it will do via the Developer Engagement Hub. Then, for example, a Software Developer would typically implement the ESP's service and integrate it either through direct coding or the Process Designer. Once the asset is complete and available via the vf-Store it can then be purchased and used and in which case the ESP service is called – for example through a process orchestration – with secured access taking place. Since the service has been called it then communicates information back to the platform, which in turn can use this data for, for example, billing and usage statistics.

- **Receive Revenue:** Once the usage information is communicated back to the platform the platform can use it, if relevant, to calculate the use against a client's usage profile (eg pay per use, bill monthly) and then bill against it typically to the manufacturing user. The user may then check this bill and if there are issues raise a dispute and if not pay the bill. A billing service will then allocate received payments according to agreements and receive a percentage share of the payments themselves before splitting the remainder to other relevant parties connected to the service usage according to predefined agreements

2.1.5 Manufacturing and Logistics Solution Providers

Manufacturing and Logistics Solutions Providers (MLSP) deliver specific services to vf-OS and the vf-OS community that permit the capture of data, knowledge, and any other activities related to a production environment. A MSLP service development process does not differ much from a typical software development as it is composed of the typical phases: requirements, specifications, development, test, deployment, maintenance, and publication. Once those services are developed and applied in a manufacturing environment, they can be connected to the vf-OS IO Toolkit (API Connectors or Device Drivers, depending on virtual or physical information).

There are several steps, which are necessary to provide MLSP services, such as:

- **Analyse Business Requirements:** The first step for the MLSP is to understand the needs and pains of its potential customers, the businesses being supported. For that purpose, there is a need to determine which requirements can be satisfied by the use of a service to get virtual and physical manufacturing information.
- **Assess Environment:** MLSPs need to understand the vf-OS environment itself, to understand its role in the ecosystem and how its services can be used best and integrated with the existing environment to reduce the difficulties in its adoption and maximise its services benefits. This includes the understanding and definition of clear APIs that cope with the architecture and ESP, and guidelines towards the integration with the platform and all its underlying services, dashboards, alerts and indicators, reports, and added-value services to again maximise its impact towards an increased acceptance of the customers.
- **Service Development:** Developers are used to work with a familiar IDE, where the (selected) SDK and all external resources, especially those focusing at the virtual and physical service development, are linked together with templates, documentation and tutorials. It should include connectivity to management environments for task planning and error management and also connected to forums (eg StackOverflow) where developers from different domains help each other and share experiences and doubts. A testing environment (eg emulator) to try out the created service, with direct connection to the equipment, in a simulated environment is also usually packed in these IDEs. A further activity of the developer is to familiarise themselves with the

development environment. After the installation and configuration of the IDE a tutorial with first steps, eg an “Easy-to-Connect-to-Machine”-like service, helps developers to get familiar with this IDE.

- **Service Testing:** Before finally deploying the service in the vf-Store, the developer normally wants to test it in a simulated environment. This environment is set up by the Quality Assurance (QA) developer. The QA developer is in charge of developing the test plans and defining the conformance KPIs, which will help to approve the developments carried out. For this, an emulator should be provided by the IDE selected together with a feature for configuring the equipment.
- **Service Deployment:** After finalising the development and testing of the service, the developer, together with the IT Manager deploys the final version of the service within the virtual and physical information. This way, when a manufacturer wants to access the services and the data offered by the equipment, it will be as easy as making use of the service just deployed.
- **Service Publication:** In addition to the deployment, the service can be published in a centralised store so the MLSP can increase its revenues by offering added value services. This central store should allow other manufacturers, who have installed the very same equipment within their plants, to acquire, download, install, and make use of the service with a secondary goal of providing feedback to the developer through the store. This feedback should include error reporting, monetisation, popularity, and ratings of the software artefact. In the publication, setting discounts and other marketing activities to increase the usage and attractiveness of the software artefact should be possible.

2.2 vf-OS Provider Scenarios

In the following sections, different scenarios for each provider role are described, which are based on the scenario activities (see Figure 2) and also from the generic provider scenarios (see Section 2.1) in a logical sequence. The following provider roles are reflected in its subsections:

- Section 2.2.1: Platform Providers
- Section 2.2.2: Software Providers
- Section 2.2.3: External Service Providers
- Section 2.2.4: Manufacturing and Logistics Solution Providers

2.2.1 Platform Providers

Based on an analysis of the various activities several chosen scenarios are presented in the following figure.

Activity Step	Description	Stakeholder involved
Scenario 1: Platform Hosting and Provisioning		
Set up server or virtual machine(s)	The vf-P Operator arranges a server or VM (hereinafter referred to as machine) for the hosting of the vf-P and other runtime components.	vf-P Operator
Configure server or virtual machine(s)	The vf-P Operator makes sure that the machine is configured with the correct operating system with current updates, supporting tools (eg web server), and libraries.	vf-P Operator

Download vf-P and components	The vf-P Operator downloads the vf-P and other runtime components, either as binary or as source code, from the public vf-OS repository.	vf-P Operator
Install vf-P and components	The vf-P Operator installs all components on the machine, starting with the vf-P. All components provide an easy-to-use installation script or an executable installer.	vf-P Operator
Configure vf-P and components	The vf-P Operator configures the components such that the appropriate runtime instances are created and configured to run as portlets in the web portal of the vf-P. Furthermore, the vf-P Operator configures the first administration and user accounts. The other components integrate seamlessly with the vf-P and support single sign-on for their users.	vf-P Operator
Run vf-P and components	The vf-P Operator starts the runtime components and checks whether the portal interface is (publicly) accessible via the web. The vf-P is now provisioned to the outside world.	vf-P Operator
Scenario 2: vf-OS Asset Configuration and Provisioning		
Set up server or virtual machine	The vf-P Operator arranges a server or VM for the hosting of one or more runtime instances of vf-OS Assets.	vf-P Operator
Configure server or virtual machine	The vf-P Operator makes sure that the machine is configured with the correct operating system, supporting tools (eg web server) and libraries, as well as with the correct version of the vf-SK and (if needed) modules from the vf-IO and/or vf-MW. The vf-SK is registered as a recognised instance in the System Dashboard accessible via the vf-P.	vf-P Operator
Install vf-OS Assets	The vf-P Operator downloads additional vf-OS Assets from the vf-Store or receives them directly from a third party software provider. The vf-P Operator installs the downloaded software from the vf-Store by means of the install script / executable installer provided with the software. The installation tools make sure that dependencies are taken into account and are installed on the machine as well.	vf-P Operator
Create vf-OS virtual machine instances	The entire setup can now be easily replicated if needed, eg for use by different tenants or for scalability or redundancy purposes. Management and monitoring of all runtime instances of all vf-OS Assets in the platform takes place via the System Dashboard installed alongside with the vf-P.	vf-P Operator
If applicable, create and publish an image from vf-OS virtual machine	The vf-P Operator can choose to create a standard image of the machine (in case of a VM) by taking a snapshot and providing a manifest.	vf-P Operator
Scenario 3: Maintenance		
Shutdown runtime instances, VMs or servers	In case of maintenance, the vf-P Operator performs a shutdown of vf-OS Assets in a standardised manner via the System Dashboard, accessible via the vf-P. The vf-OS Assets themselves make sure that stopping does not introduce faults. To provide transparency to all users a broadcast is sent out to inform users about the shutdown. After shutdown of the vf-OS Assets, VMs and/or servers may be stopped as well, depending on the type of maintenance or migration activities.	vf-P Operator
Perform Upgrades of software and/or hardware	The vf-P Operator upgrades software inside VMs, whole VM images, parts of the virtualisation technology, and/or server hardware.	vf-P Operator
Configure Backups	If needed, the vf-P Operator (re) configures several different types of backup (eg file system, databases, image snapshots, etc.) before significant updates).	vf-P Operator

Clean up file systems	Periodically the vf-P Operator performs clean-ups of technical logs in order to prevent file systems from filling up. The same is true for unused or outdated VM images.	vf-P Operator
Restart runtime instances, VMs or servers	After completion of the maintenance activities, the vf-P Operator restarts the servers, the active VMs, and/or the vf-OS Assets.	vf-P Operator
Scenario 4: Monitoring		
Capacity and performance monitoring	A configurable monitoring system (such as Nagios[1] or Zabbix[2]) enables vf-P Operators to look into the status of the running VMs and servers. The System Dashboard provides insight into the status (active or inactive) of all deployed vf-OS Assets.	vf-P Operator
Problem resolution	In case of a problem related to the system, the vf-P Operator informs the affected end user (main contact) and discusses a way to solve the problem by eg restarting, allocating more resources, or migration to newer versions.	vf-P Operator, end user
Security monitoring	Specific platform-independent network and traffic monitoring tools are used by the vf-P Operator to monitor the security status of the vf-P. The System Dashboard helps in this process by providing insight into the connections between vf-OS Assets running the cloud platform and those running outside the platform, ie vf-OS Assets running locally at an end user factory site, services running on the cloud infrastructure of Manufacturing and Logistics Solution Providers, or other 3rd-party external services.	vf-P Operator
Scenario 5: Support		
Ticketing	The end user can request a support ticket at the vf-P Operator by means of a dedicated provider-specific ticketing portlet within the vf-P. The vf-P Operator accepts the ticket and handles the support request.	End user, vf-P Operator
Inspection	The vf-P Operator can shut down and restart vf-OS Assets, VMs or servers, check non-functional logs, provide access to non-functional or functional logs to end users, and perform maintenance activities, all with the aim to understand and solve the problem reported by the end user.	vf-P Operator
Closing	As soon as a request has been handled, the ticket can be closed and the end user receives an official closing message via the provider-specific ticketing portlet.	End user, vf-P Operator

Figure 5: Platform Provider Scenarios

2.2.2 Software Developers

Based on an analysis of the various activities several chosen scenarios are presented coherently in the following figure.

Activity Step	Description	Stakeholder involved
Scenario 1: Development Tools Installation		

[1] https://www.nagios.org/
[2] http://www.zabbix.com/

Download tools	To develop vf-OS Assets, the developer has to get all development-related artefacts, which are provided by vf-OS. This includes tools for development and also for testing purposes. Therefore, the developer visits the vf-OS website to download the latest versions of the vf-P, vf-SK, and the vf-OAK.	Software Developer
Choose version of developer tools	Then the developer can choose between two versions – a bundled version that includes everything that is needed and an expert version where the developer can pick precisely, which component and tool will be downloaded.	Software Developer
Install on local machine	The developer installs the chosen version on a local machine (Windows, Linux or macOS). The vf-OS development tools, called vf-OAK, include the vf-Studio and provide a basic coding environment, debug facilities, and executable examples; the System Dashboard, a stand-alone vApp configuration environment for local setups similar to what the vf-P offers for (distributed) production environments; the Front End Environment, which contains default vf-OS UI elements and behaviour templates to accelerate the development; the SDK with all provided public interfaces of vf-OS to be used in vApps.	Software Developer
Configure Test Environment	The final step is to configure the development environment. This is performed via an easy-to-use wizard and its goal is the creation of different runtime environments that run the vf-OS Assets. In this various environment particular components of the vf-P can be activated.	Software Developer
Scenario 2: Develop New Asset		
Getting started	To create new vf-OS Assets for the vf-OS environment, the vf-Studio is used. Once the developer has familiarised themselves with the usage, the development can start by choosing what is the ambitioned outcome – a vApp or a vf-IO module (enabler).	Software Developer
Skeleton is provided to developer	Once the development process has started, the vf-Studio prepares an executable skeleton so the developer does not have to start on a greenfield site. Included in this template are distinct patterns like the handling of errors in association with the error report or a unified way of providing registration processes.	Software Developer
Support the developer	While in the development process itself, the developer can encounter difficulties. In this case, on the one hand the vf-Studio itself provides assistance with features such as code completion, and on the other hand the developer can draw on the help of other developers through the Developer Engagement Hub.	Software Developer
Final steps	Once the development process comes to an end, the developer must care for the correct functionality of all components. Therefore, it is required to test the software.	Software Developer
Scenario 3: Software Testing		
Configuration	The developer configures a new vf-OS Asset, such that it interacts with the appropriate desktop or mobile client simulators (if applicable).	Software Developer
Configure Contextual vApps	Testing can require the installation of additional existing vApps or vf-IO modules (the last ones possible together with the related testing APIs or simulators). Developers can recive these by downloading them via the vf-Store.	Software Developer
Testing	The developer tests the newly built vf-OS Asset by running the whole setup (the configuration steps only need to be executed	Software Developer

	once), using the simulators to provide input and check output, and using the debugger in the vf-Studio to check the internal functioning of the code.	
Scenario 4: Software Deployment		
Provide necessary information	Once the development and testing is completed, the developer can prepare everything to make the vf-OS Asset available in the vf-Store. Therefore basic information must be provided, eg the name of the vf-OS Asset.	Software Developer
Upload asset for review	The upload for the review process (performed by vf-OS Administrators) should be as easy and straightforward as possible. In the review process, the vf-OS App is tested against performance, security, etc. Therefore, this process follows the general upload process of the Apple App Store and Google Play. To achieve this, the upload process will be included in the vf-Studio.	Software Developer
Provide further information	Further information about the vf-OS Asset or asset can be provided via a web interface. This information contains screenshots, the requested prices, etc.	Software Developer
Asset is being reviewed and accepted	The uploaded asset will be reviewed by the vf-Store Administrator. This review includes a check for possible security issues and other restricted techniques. After the review is completed the asset will be available for customers in the vf-Store.	vf-P Operator
Scenario 5: Providing Updates		
Error becomes known	The developer logs in to the vf-Store web interface. There is an overview that shows all errors that have occurred and the developer is able to react to those errors.	Software Developer
Error gets fixed	The developer searches for the error in the code, fixes it, and tests the software afterwards to ensure that the fix did not lead to any side effects.	Software Developer
Update version in vf-Store	The developer changes the version number and uploads the new version to the vf-Store for review. This process is consistent to the software deployment process.	Software Developer

Figure 6: Software Developer Scenarios

2.2.3 External Service Providers

Based on an analysis of the various activities several scenarios are presented in the following figure.

Activity Step	Description	Stakeholder involved
Scenario 1: Assess vf-OS Environment		
Register on vf-OS	The ESP finds the vf-OS environment as a possible way to sell their algorithmic services such that Manufacturing Users can post data to their service, run specific algorithms, which they would pay for on use, and then process results would be returned. To fully explore how vf-OS operates they register their profile details, both company and individual, on the vf-P.	ESP vf-P Operator
Browse commercial and technical documentation	This then enables them to explore the wealth of commercial and technical data about the platform. They see that there are various commercial models (eg free, pay-per-use, per subscriber) and believe the pay-per-use model would be perfect for them. Technically, they read about the different	ESP vf-P Operator

	components and access the API information they need for design, operation, and billing. They also access the QoS (Quality of Service) characteristics which they need to be registered as an external service.	
Ask vf-OS Developer Community questions	Although there is plenty of material in the public domain about vf-OS, they have a few technical questions and join the vf-OS developer community via its Developer Engagement Hub.	ESP vf-OS Developer Community
Examine existing vf-OS Assets	They also want to see what existing services are their now – they want to ensure either their services are more enhanced than their competitors or not already available. The vf-Store lets them browse the services by category (eg algorithms), date (currentness), reputation (based on feedback), supplier (competitive analysis), and price. They can also see precise details and mock-ups about each service. They believe there are few competitive services in their class and can offer a better price/functionality for those that are already there.	ESP vf-P Operator
Engage with user community to find help refine service requirements	After discussing internally the ESP comes up with some options which they believe would make a really good commercial offer. However, to test these theories they interact with the vf-OS user community via the platform. They collect some feedback but at first pass the thrust of the commercial and technical offer is correct and it just needs some fine tuning.	ESP vf-OS User Community
Scenario 2: Create External Services		
Design (Fine tune) existing service to expose	With the fine-tuning in hand, received via the vf-OS user community, as well as the commercial/technical understanding of vf-OS, the ESP is ready to start designing their service at both levels. They think their service is ripe to be offered to users directly via a vApp they will build later, but think, as a first step they will expose their algorithm service via existing/new vApp to developers as an external service provider. They want to charge the end users based on a per-use model but will give the Software Developers the ability to integrate the APIs of their service for free and will publish them on the vf-Store,	ESP Software Developer
Install Developer Environment	In parallel with the designing element above, the ESP developers download the vf-OAK, which includes the vf-Studio and the designers of the core services such as the process designer. The vf-Studio is available for free and is easily installed.	ESP
Experiment with Studio	Each ESP developer 'plays' with vf-Studio following some simple service building examples and with some interaction with the Developer Engagement Hub and the development community.	ESP
Develop external service connector	They then start to try to integrate their own service using the vf-Studio and in particular the process designer. Initially, as a service aimed at Software Developers, there is no UI. However, they program the process designer such that once service APIs are called it triggers a series of internal services connected with their existing algorithm services. To do this they also install the vf-OS Process Execution Engine on their own servers although they could have used the engine on the vf-P itself but this would create too much interaction. The service is developed and tested.	ESP
Link to security/Access control component	The ESP also explores the security and access control services of vf-OS. They need to ensure that the service is integrated by validated Software Developers and is callable by validated users. Thus the vf-Studio is also used to link to the	ESP Software Developer

	platform security API.	
Scenario 3: Add Services on vf-Store		
Create/Refine vf-Store account	The ESP has an application, that wishes to publish on the vf-Store and which is conformant with the Security, QoS, and other vf-P criteria. To do this they upgrade their vf-OS account to a provider account which requires them to input further information on their business credentials (to ensure they are a valid supplier) and the nature of their expected offerings.	ESP Platform Provider
Take vf-OS Provider test	Providers are also expected to take and pass an online vf-OS exam to ensure they have fully understood the QoS, Technical, Commercial, Security, and other requirements of the platform. They pass this test but still all uploaded services will be closely examined by the vf-P Operator.	ESP Platform Provider
Create vf-Store manifest for service	Providers start to make the commercial and technical manifest for their first services. It includes obvious information such as provider details, API information, logos, images, help file, etc.	ESP
Register service on vf-Store	Providers then upload this manifest file to the vf-Store to register their first service but this still will need service authorisation.	ESP Platform Provider
vf-Store Approves service	The vf-P Operators always have to approve each application published to ensure it is conformant with the vf-OS expectations of such services, is free of malware, etc. Normally they do not perform technical testing (except if issues) but for their first services published by a new provider they perform a full test which they charge a fixed amount to the service provider based on the number of APIs to be tested and the service complexity. Once approved the Platform Provider communicates this to the ESP who, when ready can then switch on the service and open it to all relevant parties on the vf-Store.	Platform Provider vf-P Operator
Scenario 4: Make Service Available		
Make ready service (operate it)	The ESP has just had a vf-OS conformant service developed and passed the vf-OS provide process. Thus, they are waiting for their first purchases from, in this instance, the application developer community. However, for this they need to make sure the service is fully deployed and of course tested operationally. Since the vf-OS service is based on an existing one, which has been operational for several years they have no qualms about this. Largely it is the final service connection, which is checked along with the relevant calls to/from the vf-P including commercially related calls for, for example, billing and payment. This particular service does utilise the vf-OS process engine but this is deployed locally at the ESP so this is an additional test, which takes place.	ESP
Check new vApp developers using the new service	After some days, a Software Developer contacts the ESP, via the vf-P and the Developer Engagement Hub, to request some additional information about the services (the ESP subsequently upgrade the manifest information).	Platform Provider ESP Software Developer
Application developer integrates external service	After the exchanges, the Software Developer integrates the services through the process designer and as per the manifest it is at no cost making it largely a commercial risk free process for them. They perform some tests with the ESP and integrate the security parts of the manifest, as well as the commercial billing parts to ensure the ESP is suitably compensated. Once created and published on the vf-Store (along with all suitable processes to control this and as similar to the above).	Application Developer ESP Software Developer

Run application which uses service	A Manufacturing user purchases an application, which contains the ESPs algorithmic service. In the configuration the user selects the option to use the said function. This request is communicated via the vf-P from the application to the ESP who approves this and the user and platform are notified via a security token.	Manufacturing User
Validate service request access characteristics	With the approval the Manufacturing user starts to use the application and their sensor data is pushed to the algorithmic service. It is accompanied by the service token, which is used for the security/access validation as well as allowing the vf-P to know 'how much' of the service is being accessed. The service itself is triggered through an orchestrated process.	ESP
Once used communicate usage information to platform	Correspondingly the ESP application communicates usage information back to the vf-P – for example how much processing timing it has taken the vf-P to process the data.	Platform Provider
Scenario 5: Receive Revenue		
vf-P calculates usage against subscription profile	Once an ESP service has been called, both the caller and the receiver will communicate usage back to the vf-P and payment information back to the vf-Store. If the two agree (and both match the same original criteria) then this signifies to the vf-P that the amounts are valid.	Platform Provider
vf-Store bills client	On the next monthly billing cycle the Manufacturing user should be billed and the vf-Store will be automatically paid by direct debit arrangement, which the user signed up to when they registered to use the application. Of course there is a dispute process (similar to Alibaba) if there are any service usage disputes.	Platform Provider Manufacturing User
Client pays vf-P	The client does not issue any dispute within 3 days of the itemised bill and thus by contract the service is deemed correctly received and the payment is received by the vf-Store.	Manufacturing User Platform Provider
Platform Pays External Service Provider	The vf-Store takes a percentage commission on the payment and relays the remainder, along with usage information to the ESP. The ESP notes the payment and the success of their first exploration into vf-OS and will look to develop further services.	Platform Provider ESP

Figure 7: External Service Provider Scenarios

2.2.4 Manufacturing and Logistics Solution Providers

Based on an analysis of the various activities several chosen scenarios are presented in the following figure.

Activity Step	Description	Stakeholder involved
Scenario 1: Analyse Business Requirements		
Analyse pains and needs	A business user/area defines a document or set of needs that can be satisfied by an information system. This definition includes functionalities, data and knowledge, and expected results.	Business
Determine a set of requirements	A Business Analyst determines with the business customers more detailed features and constraints, identifies processes,	Business Analyst

	activities, sensors, and actuators, and suggests generic enablers and results to be achieved.	
Define SLAs, KPIs and other metrics	The Business Analyst determines non-functional requirements with respect to capacity, responsiveness, metrics and integration needs.	Developer + Business Analyst
Information Specifications	The Business Analyst specifies the data and information models, the expected integrations with the business, and other needs such as when and how the business data is being ingested by the information system, defining the procedures and rules to do it.	Business Analyst
Scenario 2: Assess Environment		
Get Information from vf-OS	Analyse the specifications of the existing vf-OS environment, including specifications on how to develop vf-OS Assets, and how to integrate them and services into vf-OS.	Developer Engagement Hub
Define connection APIs	Integrate the developed vf-OS Assets and the Information System using vf-IO and ESP.	vf-IO, ESP
Integration with vf-OS	Determine guidelines on how to integrate properly with vf-P and other resources such as the System Dashboard and Front End Environment, and ask directions on how to integrate with vf-OS and its Development Community.	vf-OS Developer Community
Scenario 3: Service Development		
Getting started	To create a service the vf-Studio will be used, which includes the vf-OAK. Once the developer has familiarised itself with the usage of the vf-Studio, the development of the service can start.	Software Developer
Usage of service templates	To ease the development of services, the vf-Studio provides templates with built-in functionalities such as hardware configuration, default API Connectors, Device Drivers, pre-defined manifest content or handling of errors.	Software Developer
Helping the developer	Nearly all developers are stubborn and prefer to sort out by themselves the problems they may encounter. However, the vf-OAK comes with a Developer Engagement Hub providing assistance to the developer in those moments that all viable solutions have been tried out and have failed.	Software Developer
Test of development	Once the development phase is reaching its end, the developer has to verify the correct functionality of the service before sending it to the production environment.	Software Developer
Scenario 4: Service Testing		
Development of a test UI	The QA developer creates a test UI that will communicate with the enabler so all data gathered will be examined to identify errors in the service development.	QA Developer
Configuration of the Test Environment	Both the software developer and the QA developer use a configuration panel inside the vf-Studio to configure the new service. This requires wiring connections, ie by connecting interfaces between the equipment, the service and the previous UI.	Software Developer, QA Developer
Configure Dependencies	All dependencies that are included in the manifest file must be installed or should be installed during the installation process of the newly created service so it runs without any problems.	Software Developer, QA Developer
Testing	The QA developer creates the testing routines and tests the newly built service by running the whole setup, using the simulators and/or the equipment directly to provide input and check output in the specific test UI.	QA Developer

Scenario 5: Service Deployment		
Preparing the deployment	Once the development and testing is successfully completed, the developer can prepare everything to deploy the service in the equipment production environment. A wizard within the vf-Studio will guide the developer through the process of creating the installation packages.	Software Developer
Installation in target systems	The IT Manager together with the Software Developer deploys the enabler in the target systems (app store).	Software Developer, IT Manager
Scenario 6: Service Deployment		
Provide necessary information	The developer can prepare the service for being accessible, ie purchased, in the vf-Store. To allow the publication of the enabler basic information has to be provided (eg what the equipment is developed for, the price, the sensors it reads) in the dialog for publishing a vf-OS Asset to the vf-Store.	Software Developer
Upload service for review	A review process is necessary to ensure that the service is reliable, performs as expected, and is free of “hidden” material by the vf-Store Administrator.	Software Developer
Service is being reviewed and accepted	The uploaded service will be reviewed by the vf-Store Administrator. This review includes a fundamental check for possible security issues and other restricted techniques, similar to other online stores, such as Apple App Store or Google Play. After the review is completed and satisfactory the service will be available for customers in the vf-Store. In the event that further information is requested by the vf-Store owner, the developer has to satisfy with these requirements else the service will not be published.	vf-Store Owner, Software Developer

Figure 8: Information Provider Scenarios

3 Adaption to User Scenarios

After the generic and vf-OS specific provider scenarios have been defined and described in Section 2, additional activities related to vf-OS user scenarios are taken into account by further analysis of the defined vf-OS user scenarios in Sections 3.1, 3.2, and 3.3. This ensures that vf-OS will fulfil the required needs, which arise from these scenarios. Finally, the extracted needs and results are described and outlined in Section 3.4.

3.1 User Scenario 1: Manufacturing and Logistics – Automation

This pilot is elicited from MASS and ViaSolis and takes place in a factory environment. A detailed description of this use case can be found in “D2.1 User Scenarios Characterisation”.

In the following narrative, each pilot vApps from this use case is described, analysed, and the particularities are extracted to filter requirements for the provider. This use case consists of the following planned vApps:

- vfFailurePrevention
- vfFailureManager
- vfStockPolicies
- vfProductionFollowUp
  vfMaintenanceCalendar

Common technical foundations are summarised in Section 3.1.6.

3.1.1 Pilot vApp: vfFailurePrevention (P1.1)

Short Scenario Description

This vApp aims to automatically detect equipment maintenance or failure events using data analysis algorithms and to trigger the appropriate monitoring process by sending notification alarms to the corresponding actors. PLC data will be stored in a way that the data analysis algorithm can use it. When the system detects an event, it notifies maintenance managers in both MASS and VS through monitoring alarms. Upon receiving an alarm, maintenance managers analyse the information and decide if any maintenance process is required.

Analysis

As this vApp deals with users from different organisations a role-based access control (RBAC) is needed to take into account the different perspectives of the users. Data from the equipment should be analysed for the vApp with the help of the Data Analytics component to get the needed information. A subscription mechanism of events should be in place so the user is warned about possible issues in the equipment. The information presented in the vApp is then analysed by MASS and VS maintenance managers.

The extraction of technical foundations follows:

- **Security:** Sensitive data must be protected against unauthorised use, so a user manager has to regulate user permissions in order to guarantee maximum security
- **Integration of Equipment:** vf-OS Enablers and an API should allow the vApp to access information from the equipment

- **Data Transmission:** The data should be transferred from the source to the processing component and finally to the user in a transparent and efficient way
- **Subscription:** The user, via the vApp should be able to subscribe to different events either triggered by the vApp including based on the data analysed or equipment errors
- **Exploitation of Information:** The user should be presented with the raw data, the information (ie trends and forecast) calculated from the data, to be able to extract the knowledge by themselves

3.1.2 Pilot vApp: vfFailureManager (P1.2)

Short Scenario Description

This vApp performs the automatic registration of failures of the Tabber Stringer machine in a production line by using data from the PLC and notifying the users with an alarm when a failure is detected. The equipment owner can also create an alarm manually if a fault is noticed.

To support the search for the detected failure solution, both the owner and the supplier of the equipment may view the history of registered faults and the solutions applied.

Analysis

This vApp deals with users from different organisations, thus a RBAC (Role Based Access Control) is needed. Automatic alarms should be displayed in the vApp UI so the user can react accordingly. The alarms, together with their metadata, should be able to be manipulated and stored. An acknowledgement mechanism of the different messages exchanged between the different actors should be also in place to allow the processing of the faults. To solve the failures, both actors may need to get access to the documentation of the equipment along with the history of failure's logs and a set of possible interventions as per previous experiences but also containing the full range of solutions.

The extraction of technical foundations follows:

- **Security:** Sensitive data must be protected for unauthorised use, so a user manager has to regulate user permissions in order to guarantee a maximum of security
- **Integration of Equipment:** vf-OS Enablers and an API should allow the vApp to access critical information from the equipment
- **Data Transmission:** The data should be transferred from the source to the processing component and finally to the user in a transparent and efficient way
- **Subscription:** The user, via the vApp, should be able to subscribe to alarms from the equipment, ie based on errors
- **Data Access and Manipulation:** Metadata from the alarms should be read and updated in the data storage. In addition, documentation from the equipment, historical data and potential solutions should also be available to the user

3.1.3 Pilot vApp: vfStockPolicies (P1.3)

Short Scenario Description

The main goal of this vApp is to manage spare-parts that each company needs, monitor their stocks, and alert when the stock is lower than a minimum defined for each spare-part.

In order to meet this goal, it is necessary to feed "Spare-part Data Storage" with spare-part data either from each company's information systems or by manually importing. Each

imported spare-part is identified in the equipment supplier's system with a unique code and this code must be in the equipment owner data storage. All the information will be visible continuously and when the algorithm detects that there is a shortage in some spare-part the application will alert the maintenance manager of the company.

Analysis

This vApp deals with users from different organisations, thus RBAC (Role based access control) is needed. Data management for product data is crucial to get the right spare-part to be replaced. Spare-parts replacement orders are directly placed on the suppliers via MASS' ERP system. When the order has been acknowledged by the recipient MASS' ERP system is updated with the new spare-parts quantity accordingly.

From a 'tracking-order' perspective, the data should be accessible from VS' ERP system via an external link to the vApp. The extraction of technical foundations is:

- **User Management / Permissions:** Sensitive data must be protected for unauthorised use, so a user management have to regulate user permissions in order to guarantee a maximum of security
- **Integration with ERP Systems:** An API should allow the integration with ERP systems guaranteeing the reading of spare-parts, placing orders, and tracking shipments

3.1.4 Pilot vApp: vfProductionFollowUp (P1.4)

Short Scenario Description

This vApp intends to collect selected production PLC data from a Tabber Stringer machine in real time. The actions include to: store this data, and calculate the production KPIs as productivity, availability, performance, or quality using collected values to monitor these results and follow-up the production. Production issues will be detected, if a calculated indicator will be too low compared to its preconfigured value and an alarm will warn the production line manager.

Analysis

Data is sent from the equipment to the vApp that calculates certain KPIs. These calculated KPIs should be displayed in the vApp UI so the user can analyse them. An alarm bound to a KPI should be triggered after this KPI is below a predefined threshold. The alarm is communicated to the user so the user can act.

The extraction of technical foundations is as follows:

- **Integration of Equipment:** vf-OS Enablers and an API should allow the transmission of critical information from the equipment to the vApp
- **Data Transmission:** The data should be transferred from the source to the processing component and finally to the user in a transparent and efficient way
- **Subscription:** The user should be able to subscribe to alarms from the equipment, ie based on errors
- **Exploitation of Information:** The user should be presented with the raw data, the KPIs, and potentially information (eg trends and forecast), calculated from the data, to be able to extract the knowledge by themselves

3.1.5 Pilot vApp: vfMaintenanceCalendar (P1.5)

Short Scenario Description

This vApp's aim is to schedule a maintenance calendar for a Tabber Stringer machine and to handle the communication in terms of maintenance operations between MASS and VS. MASS will define preventive maintenance operations needed by the Tabber Stringer and these operations will be recorded in the data storage. Depending on the annual calendar configured by VS and the operations registered by MASS, an algorithm should create a scheduled maintenance calendar that can be modified automatically when new operations are added or when unscheduled maintenance tasks took place.

Depending on spare-part orders, time limits, etc., the application must notify the maintenance managers to establish the intervention in advance. When the intervention is completed the maintenance managers can add information and the last maintenance date will be modified and recalculated automatically.

Analysis

Calendar information is shared across teams. The information stored in the ERP system regarding spare-parts and maintenance interventions is used for the calendar calculations by the vApp. These updates are communicated to the subscribers of the calendar. The extraction of technical foundations is as follows:

- **Information distribution:** The data coming from the calendar should be shared only to subscribers regardless their roles and their affiliation
- **Data Transmission:** The data should be transferred from the source to the processing component and finally to the user in a transparent and efficient way
- **Subscription:** The user, via the vApp, should be able to subscribe to the different calendars
- **Data Access and Manipulation:** The metadata of the interventions and the maintenance operations from the machine should be read and updated from the data storage
- **Integration with ERP Systems:** An API should allow the integration with ERP systems guaranteeing to keep track of latest spare-parts stock
- **Exploitation of Information:** The user should be able to see the maintenance operations and the interventions scheduled

3.1.6 User Scenario Analysis

One main points of this user scenario is the sharing of information across the supply chain which is crucial for executing daily activities and for planning future actuations. It has to be noted that in this case two different users from the two different actors may need to access the same piece of information yet they should have different accessing roles to guarantee privacy and prevent information loss. This can be achieved by a dedicated user management and permissions system. Furthermore, it can provide functionalities to protect sensitive information from unauthorised use and guarantee a maximum of security.

Additionally, a topic-based data storage has to be available to execute the different proposed vApps. This data storage must implement the same user management and permissions system described above to provide access to it.

3.2 User Scenario 2: Construction / Industrialisation

This pilot is elicited from Consulgal and takes place at constructions sites. A detailed description of this use case can be found in "D2.1 User Scenarios Characterisation".

In the following sections each pilot vApp from this use case is described, analysed, and filtered to provide requirements for the providers to be covered by vf-OS. This use case consists of the following planned vApps:

- vfDocumentPortal
- vfSteelValidation
- vfOnSiteManager
- vfConcreteFeedback
- vfProductValidation

At the end, common technical foundations are summarised in Section 3.2.6.

3.2.1 Pilot vApp: vfDocumentPortal (P2.1)

Short Scenario Description

This vApp enables a party to access, edit, and store all necessary documents regarding construction site operations digitally through a web based solution through a construction sites lifetime. The documents are differentiated between templates and external documents. Templates can be edited directly from the vApp, but external documents can only be uploaded as images or text documents and then be accessed in a read-only mode. In addition, this vApp shall be used by various users with different permissions so as to only access authorized documents.

Analysis

Different types of users at a construction site require different roles for the vApp, which leads to a user management with permissions for different access rights. Also the data management is very important, since the integrity of the documents must be ensured. Templates are obtained and must be downloaded from a database. Optimally, the most used templates are provided locally. External data should be uploaded as an image or text document. Therefore, for example, a camera must be installed on a mobile device, which can take pictures of sufficient quality. The data should be accessible from a browser and a mobile device, therefore a web based solution is preferred. The extraction of technical foundations are:

- **User Management / Permissions:** Sensitive data must be protected for unauthorised use, so a user manager has to regulate user permissions in order to guarantee a maximum of security
- **Upload of images or text documents:** A messaging format must be used, which is able to transmit documents without any data loss
- **Mobile device including a camera:** This scenario allows for external documents to be photographed, uploaded and includes access from the software with the camera of the device

3.2.2 Pilot vApp: vfSteelValidation (P2.2)

Short Scenario Description

This vApp identifies steel bars and is able to validate them regarding to their ID markings. The process starts by taking a picture of a transport document and enclosed ID tags. After that, pictures of the ID markings on steel bars are taken to compare it with the ID marking from the expected steel bar. For this it must be ensured that the quality of the images is good. Finally, the comparison of the ID markings is performed by the vApp, which leads to a report for approval.

Analysis

To take photos from documents and steel bars, the used device (typically a mobile one) must have access to a camera with sufficient characteristics (eg resolution, depth of field). The resulting images must be processed and recognized to enable a comparison against images of other steel bars including validation of accordance. The resulting report should not only be sent to the user directly, it can also be provided to a remote user, which takes a messaging functionality for granted (either via client-to-client notification or email). Furthermore, the reports shall be saved in a central storage to gain access for other users. The extraction of the technical foundations is:

- **Device including camera access:** This scenario provides functionalities to take photos of documents and steel bar markings; Therefore, the developer has to use an internal API of the host system to connect the software with the camera of the device
- **Image Processing:** Taken images must be prepared for the later image recognition; Therefore, the images must be processed with help of image processing techniques
- **Image Recognition:** Image recognition describes the process of the comparison of feature extractions of images to detect matches; For this APIs or external online services can be used
- **Validation Reporting:** The validation report shall be sent to remote users, for this a message infrastructure (Publish and Subscribe, app to app, push notifications, email, etc) must be used

3.2.3 Pilot vApp: vfOnSiteManager (P2.3)

Short Scenario Description

This vApp concerns delay handling in daily processes at a construction site. After a delay is reported to the vApp (eg from a truck driver), the vApp forwards the notification to responsible users. After the responsible person receives the notification and confirms it, the vApp automatically alerts the construction supervisor. The vApp forecasts the impact and lets the supervisor analyse the delay. If the impact influences other processes and surpasses a predefined threshold the personal work plan of the affected worker is shown to let the user edit changes. These changes are forwarded to the affected persons.

Analysis

The reporting functionality among each user implies a need for either a notification mechanism to report messages from user to user, or the reports will be sent via e-mails attached with an URL with the delay information. After confirmation from the final responsible person the vApp provides a monitor and alert handling to send warnings to other responsible persons automatically. The vApp must also be aware of all processes regarding time, actuality, and personal work plans. This data must be integratable into the vApp either by adding this to the vApp manually or importing the data from another source. Due to changes in personal work plans a synchronisation with the data storage must be implemented to keep all information up to date. According to this, a database must be installed to have a common data source for all clients. The extraction of the technical foundations is:

- **Monitor and Alerting:** Values of the construction site (eg arrival times of materials) must be observed and if a threshold is exceeded an alarm is triggered, which makes use of a messaging infrastructure to notify responsible persons

- **Import/Export:** Personal work plans should be entered into the vApp; Since a manual operation is too complex an import/export function is needed to ease this process and save time
- **Messaging:** Either email communication or a app-to-app communication should be enabled to exchange notifications

3.2.4 Pilot vApp: vfConcreteFeedback (P2.4)

Short Scenario Description

This vApp gives the feedback of the test results of newly delivered parts from the construction site to a plant, which is responsible for the delivered parts. During this process an external application is used to generate test values and reports them to responsible persons. These test values must be integrated into the vApp to compare the values with predefined thresholds. While comparing the values, a monitoring and alerting mechanism warns the user if test values are below the threshold.

Analysis

To integrate data from an external application, an import functionality is necessary to integrate test values in a standardised way and optimised for the comparison process. Alternatively, a form must be provided to guide the user through the process. Before comparing test values with thresholds, the thresholds must be defined and imported from another source. The comparison itself must be equipped with a monitoring and alerting mechanism that uses a messaging technology to send the complete report to a remote person. The extraction of technical foundations is:

- **Import:** Test data must be integrated into the vApp; Since a manual operation is too complex an import/export function is welcome to ease this process and save time
- **Alternative Import:** To have a backup solution a form should be provided to enter test values manually
- **Monitor and Alerting:** Test values must be compared and if a threshold is exceeded an alarm is triggered, which makes use of a messaging infrastructure to send warnings to responsible persons
- **Messaging:** Other users must be notified about the reporting of the comparison of the test values; therefore a message infrastructure must be selected

3.2.5 Pilot vApp: vfProductValidation (P2.5)

Short Scenario Description

The vApp manages the process of ordering and receiving products at a construction site. Two different approvals are necessary to validate the entire process. An ordering process is initiated by a person who needs to have some new products to continue at the construction site. Therefore, a “Product Approval Request” form must be sent to the responsible person who is in charge of approving or rejecting this. The “Product Approval Request” contains all products to be purchased. After the responsible person receives the request, the person has the option to approve or reject the order. If the order is approved, the process can be continued and the order can be submitted. Otherwise, the order must be updated according to predefined rules and the “Product Approval Request” must be submitted again with the updated data. After the products relating to the order arrive at the construction site, they must be compared with the shipping manifest. A second form is filled in (Reception of Materials and Equipment) and attached with the shipping manifest in the vApp. This will then be sent to another person who validates the data and repeats the

validation process by checking for conformity of the product. Again, this form can be approved or refused. If approved, the products can be stored at the production site, else the product returns to the supplier and a compliant replacement will be requested.

Analysis

The processes of approval for a product request and reception of materials and equipment can be broken down and realised through two templates. These templates must be downloaded or deposited in the vApp as a default. The vApp has to be able to accompany the entire process. Once an order arrives at the construction site, the vApp needs an upload functionality to attach a shipping manifest, which comes with the respective delivery. If a shipment is rejected, the responsible person for checking the shipment conformity must be informed immediately – there real-time communication must be ensured. Finally, the reports should be logged in the data storage. The extraction of technical foundations is:

- **Templates:** Up-to-date templates for “Product Approval Request” and “Reception of Materials and Equipment” should be downloadable or at least preconfigured in the vApp as default forms
- **Data Exchange:** Standardised forms that can be sent to another party needs a defined data format for exchange (eg JSON, XML)
- **Upload:** Shipping manifest file must be attached to a filled template form for validation
- **Messaging:** A client-to-client messaging infrastructure is necessary to enable the push notifications
- **Real-Time:** Depending on the replacement process, a real-time communication must be ensured between the parties to return the products with the same truck that delivered the products

3.2.6 User Scenario Analysis

One additional main point of this user scenario is the internet connection, which cannot be guaranteed on construction sites. Of course it depends on the location, but to be prepared for special cases (eg a construction site without any internet connection) a vApp should be runnable ‘no internet’ in mind. Another side effect of the handling with confidential documents (ie contracts) is to ensure data integrity and security, especially during the transmission. Therefore a synchronisation mechanism has to be available, which ensures all documents are available on each device and which keeps track of possible collisions when documents have been changed on multiple systems. This mechanism has to be aware of the above mentioned lack if a stable internet connection as well.

The documents themselves must be stored in a central data or a cloud storage component that can be even provided by an external service.

3.3 User Scenario 3: Manufacturing Assembly: Collaboration

This pilot is elicited from APR and Tardy and takes place between factories. A detailed description of this use case can be found in “D2.1 User Scenarios Characterisation”.

In the following each pilot vApp from this use case is described, analysed, and the particularities are extracted to filter requirements for the providers to be covered by vf-OS. This use case consists of the following planned vApps:

- vfCollaborationAnalyser
- vfIndusEnabler

- vfProductionPlanner
- vfQualityInsurance
- vfProductionTracker

At the end, common technical foundations are summarised in Section 3.3.6.

3.3.1 Pilot vApp: vfCollaborationAnalyser (P3.1)

Short Scenario Description

This vApp has the objective to evaluate the ability of a group of parties (in this case, made up of the industrial partners APR and TARDY) to jointly industrialise and produce a common customer project. The targeted application integrates as input a STEP file and generate as output a classification of the business opportunity: Mandatory, High ROI, nice to have, or not relevant. With this information the vApp will classify the consortium's capability.

Analysis

The process is to determine the partners' capability. For this purpose, the vApp starts by importing a STEP file. Access to data storage is needed, as well as access to the features under analysis, such as contracting details: delay, quantity, collaboration conditions, quality control process, etc. The vApp shall then assist to decompose the project features to be able to make the classification for each feature. To fulfil this the vApp connects to the partner's IT system to aggregate more information about the respective feature. With this information and the sales rules from the partners (which are input into the vApp and the result of contracts between the partners), the vApp generates classification reports of the business opportunity and identifies a decision regarding the consortium capability to satisfy the implementation of the customer project, providing additionally reports for the industrial engineering managers and a report for the customer.

This should be a web-based solution with access to:

- **User Management / Permissions:** Data must be kept safe from unauthorised use, so the system must feature an accessibility management tool to ensure only authorised accesses have the rights to fetch the business sensitive information
- **Processing of STEP files and getting information from Analytics:** Key for the correct functioning of the vApp is the ability to assist in the decomposing of STEP files of any complexity; Furthermore, the partner's provided analytical information to make the classification for each feature
- **Data Storage of STEP files and other information and evidences:** A messaging format must be defined, which is able to transmit documents without any loose of information
- **Reporting capability:** This vApp needs to be able to generate reports over the analysed scenarios and information

3.3.2 Pilot vApp: vfIndusEnabler (P3.2)

Short Scenario Description

The vfIndusEnabler vApp's objective is to provide a support to the industrialisation teams from a group of organisations (in this case, of the industrial partners APR and TARDY) in order to allow STEP files to become real manufactured products in the most efficient way by managing the corresponding communication. The support is via user-friendly

communication channels between industrials partners and the automated syntheses of the project features based on collected data

Analysis

To satisfy this scenario, the vApp needs to collect information about quotations so the customer can create orders based on them. Included in the order is common information like the chosen products, quantity, or other comments. The industrial and methods engineering manager receive the information of the placed order to find out which products need to be produced. To support this process the vfIndusEnabler vApp analyses the technical requirements and production detail information for the production process. This information is stored in the ERP system.

This vApp thus needs to have access to:

- **User Management / Permissions:** Sensitive data must be protected for unauthorised use, so a user management have to regulate user permissions in order to guarantee maximum security
- **External Interfaces (ESP):** To get the customer quotations and access to the industrial partners' communication channels
- **Internal Interfaces and drivers:** The vApp must have access to the production detail information provided by the vf-MW
- **Output Integration with ERP/Notification system:** This vApp needs access to the notification environment so that the industrial teams can be notified to proceed with the defined features decomposition report, and to an Information System (ERP) for storing the project outcomes

3.3.3 Pilot vApp: vfProductionPlanner (P3.3)

Short Scenario Description

This vApp has the objective to coordinate the production sequence among two industrial plants and ensure coherence on the production lines. Thus the vfProductionPlanner brings a first key step in the collaborative process control.

Analysis

By receiving the technical data from the consortium's ERP systems, this vApp generates a production sequence and report for the consideration of the production teams to approve or reject. Upon approval, the vApp notifies the customer that the process is about to implement the sent order, and validates the orders in the ERP system. The vApp then proceeds to check the ongoing productions according to the approved rules and specifications.

To achieve this, the vApp must have access to:

- **ERP Information System:** Access to the technical data stored in the ERP of both industrial partners
- **Messaging:** The scenario needs to have a messaging infrastructure to send notifications to other users

3.3.4 Pilot vApp: vfQualityInsurance (P3.4)

Short Scenario Description

This vApp has the main objective to anticipate quality divergences during ongoing production in order to respect customer commitments. In order to reach this resources will

be deployed such as integrated CPS system on the production line, alert reporting systems, or decision making support tools (eg trend, graphics)

Analysis

This vApp fetches events of any anomaly in terms of product quality, resources, etc is detected. As soon as an anomaly is detected the impact of those events will be analysed and the vApp generates recommendations and notifications to the production manager. If there is a long-term impact, the vApp issues a separated alert report to the production consortium, so that necessary actions can take place to fix that issue.

To ensure the vApp fulfils its purpose, it must have access to:

- **The CPS inventory:** This is essential so that the app can get the anomaly events generated by this module
- **Analytics:** Eventual anomalies result from analytic calculations and therefore, it is necessary to use algorithms that reflect this use case
- **The actuators for the production control:** Access to manufacturer and logistic information must be provided to get manufacturing information, such as orders and production line information
- **Messaging:** This vApp must be able to generate notifications in a multicast manner; A corresponding infrastructure must be provided for this

3.3.5 Pilot vApp: vfProductionTracker (P3.5)

Short Scenario Description

This vApp has the objective to analyse the ongoing production, make a production forecast and after that compares it to the planned goals. Thus production managers from APR and TARDY can be alerted to react and anticipate deviances of the delivery date. Based on their decisions, the vfProductionTracker will suggest actions that can be accepted by the user to mitigate unfavourable forecasts.

Analysis

This vApp must fetch orders (both valid and initialised) and information about the current production line. After that it compares the gathered information and makes some statistical and trend analysis over the collected data. Then, this vApp concludes the projected forecast for the production process. If this forecast does falls behind the initial estimations, the vApp generates alerts to the production manager. Upon receiving this alert, the manager will use this vApp to adjust manufacturing orders.

To ensure the vApp fulfils its purpose, it must have access to:

- **The orders repository (data storage):** This is essential so that the vApp can get the production manufacturing orders
- **User Management / Permissions:** Data must be kept safe from unauthorised use, so the system must feature an accessibility management tool to ensure only authorised accesses have the rights to fetch the business sensitive information
- **Access to the production environment:** The scenario needs to verify and validate the actual production environment for determining the conformity of the produced goods
- **Messaging:** This vApp needs access the messaging infrastructure so that the industrial teams are able send and receive messages/notifications
- **Analytics:** Statistical analysis are needed to forecast a production process

- **Reporting System:** This scenario should be able to generate local or global reports

3.3.6 User Scenario Analysis

The most important feature for this user scenario is the access to the APR and TARDY repositories, and the handling of sensitive (confidential contracts and production orders) information. Therefore a data storage is needed, which can be either a central database or a cloud storage solution accessible through the External Service Provision Framework. Furthermore, almost all vApps proposed need access to ERPs of the consortium, as well as to the production orders and the vf-OS Data Storage.

There is also some care taken to generate reports and notifications to the consortium. This must be available to the respective business users.

3.4 User Scenarios Conclusions

The evaluation of all user scenario pilots revealed that each provider [Software Developers (SD), Manufacturing and Logistics Solution Providers (MLSP), and External Service Providers (ESP)] is covered. This provides the opportunity to show the possibilities of vf-OS through the pilots. The following figure shows the exact assignment of all provider roles in the User Scenarios.

User Scenario	SD	ESP	MLSP
User Scenario 1: Manufacturing and Logistics – Automation			
vfFailurePrevention (P1.1)	✓		✓
vfFailureManager (P1.2)	✓		✓
vfStockPolicies (P1.3)	✓		
vfProductionFollowUp (P1.4)	✓		✓
vfMaintenanceCalendar (P1.5)	✓		✓
User Scenario 2: Construction / Industrialisation			
vfDocumentPortal (P2.1)	✓	✓	
vfSteelValidation (P2.2)	✓	✓	
vfOnSiteManager (P2.3)	✓		
vfConcreteFeedback (P2.4)	✓		
vfProductValidation (P2.5)	✓	✓	
User Scenario 3: Manufacturing Assembly: Collaboration			
vfCollaborationAnalyser (P3.1)	✓		
vfIndusEnabler (P3.2)	✓	✓	✓
vfProductionPlanner (P3.3)	✓		✓
vfQualityInsurance (P3.4)	✓		✓
vfProductionTracker (P3.5)	✓		✓

Figure 9: User Scenario Assignment of Provider Roles

The commonalities in each user scenario are already evaluated and described in Sections 3.1.6, 3.2.6, and 3.3.6. But there are also some overlapping requirements within the user scenarios, which requires more accurate attention. These are:

- **Messaging:** Several pilots show different messaging variants, such as app to app, Publish and Subscribe, email, etc. An infrastructure should be provided in vf-OS to enable message transmission
- **Resource Management:** A data storage is needed to save data centrally. Furthermore, the data storage must support different types of data

- **Access to Manufacturing data:** The access to virtual and physical information from the manufacturing environment are often addressed and should be available through API Connectors and Device Drivers in the vf-OS IO Toolkit.

The following figure identifies main challenges from the analysis (see Sections 3.1.6, 3.2.6, and 3.3.6) of the different user scenarios plotted against each scenario.

Challenge	User Scenario 1	User Scenario 2	User Scenario 3
User Management: The user's aim is to use vApps with different user roles (eg administrator, manager, and worker). Therefore, it is necessary to provide vApps with different behaviours related to user profiles. This includes authorisation and authentication mechanisms that should be provided in a way that this functionality can be adapted by providers within vApps with ease.	✓	✓	✓
Security: Secure data transfers are necessary to guarantee data integrity and security for sensitive data exchanges between vApps. Especially for the exchange of contract information between companies should be protected against misuse. Therefore encryption and decryption mechanisms should be provided to allow providers to use those standardised security mechanisms.	✓	✓	✓
Connectivity: Internet connectivity should not be considered as default, since construction sites might be out of reach of any connectivity. Therefore a synchronisation framework should be provided to synchronise gathered offline data with online data.		✓	
vf-OS Accessibility: Well defined interfaces allow providers to integrate a communication module in a vApp to use vf-OS online functionality, such as analytics.	✓		✓
Resource Management: Data access should be reliable, robust, and always accessible. Furthermore, an alternative data storage should be configurable to provide a backup solution. Data storages can be used from vf-OS directly or external services can also be used. Furthermore the data storages must be flexible and support multiple data types, which are content neutral	✓	✓	✓
Messaging: Sharing information between several parties should be provided to allow an instant messaging for time critical activities, eg approval for order requests. The challenge is to configure the messaging infrastructure of vf-OS or to find another reliable external service for this, which also provides sufficient security.	✓	✓	✓
Enablers: An easy integration of enablers into vApps is indispensable. For this, various interfaces should be made public.	✓		✓

Figure 10: Requirements derived from User Scenarios

The challenges from Figure 10 influence the requirements in “D1.5 Requirements Specifications” and interactions between components and providers in “D2.1 Global Architecture Definition”. The impact on D1.5 Requirements Specifications results in additional effort to cover requirements not only from the user’s but also from the provider perspective.

Another impact is the focus on provider requirements in “D2.1 Global Architecture Definition”. Figure 10 lists several technical challenges that vf-OS has to consider during the design of the global architecture, eg provider interfaces (see Section 1.3.1) and developer tools (see Section 1.3.2). vf-OS should also provide means to integrate resource management, security, and collaboration solutions that common challenges are performed consistently and can be integrated easily in vf-OS Assets.

Finally, the challenges from Figure 10 are prioritised as Must/Should/Could, which will be considered in “D1.5 Requirements Specifications”. Based on the frequency of the allocations in the vApp descriptions and in Figure 10 the resulting priorities for providing solutions in vf-OS are depicted in Figure 11.

Property	Pilots	Must	Should	Could
Messaging	P1.1, P1.2, P1.4, P1.5 P2.1, P2.3, P2.4, P2.5 P3.3, P3.4, P3.5	✓		
Resource Management	P1.2, P1.5 P2.1, P2.5 P3.1, P3.5, P3.5	✓		
Access to virtual and physical information	P1.1, P1.2, P1.3, P1.4, P1.5 P3.1, P3.2, P3.3, P3.4, P3.5	✓		
Event-based notifications (Publish/Subscribe)	P1.1, P1.2, P1.4, P1.5	✓		
Data Visualisation	P1.1, P1.2, P1.3, P1.4, P1.5 P2.1, P2.2, P2.3, P2.4, P2.5 P3.1, P3.2, P3.3, P3.4, P3.5	✓		
Access to external services	P2.1, P2.2 P3.1, P3.2		✓	
User management	P1.3 P2.1 P3.1, P3.2, P3.5		✓	
Analytics	P3.4, P3.5	✓		
Security	P1.1, P1.2 P2.1 P3.1, P3.2, P3.5		✓	
Hardware Access in vf-OS Assets	P2.1, P2.2			✓

Figure 11: Prioritised Provider Requirements

In Figure 11 can be seen that nearly all pilot apps are dependent on basic infrastructure functionalities such as “messaging”, “resource management” and “event-based notifications (Publish/Subscribe)”. Therefore, these are prioritised as a Must-feature. In addition to that, “access to virtual and/or physical information”, “data visualisation”, and “analytics” are also ranked as a Must-feature due its massive use in the pilot apps and also as functionalities that are core functionalities of several apps. Furthermore, the features “access to external services”, “user management” and “security” are ranked as should, because of several incidents and exchangeable functionalities but nevertheless

useful and required by some pilot apps. Moreover, a complex user management is not part of vf-OS, but nevertheless a concept in combination with security is developed. The only Could-feature is for “accessing hardware modules” of mobile devices in vf-OS Assets. This is not part of vf-OS itself but for developing vf-OS Assets it could be ensured, that development technologies (programming language) are selected to support this.

4 Revenue Streams for Providers

This section concerns the revenue streams for the providers. First, revenue stream models are described in common in Section 4.1 including the in more detail the models with the most potential in terms of vf-OS. Second, in Section 4.2, the user scenarios are analysed and for each provider role a recommendation for specific revenue stream models to be used is given.

4.1 Revenue Stream Models

A revenue model is a framework for monetisation purposes. It identifies which revenue source to pursue (revenue stream), what value to offer, how to price the value, and who to bill for the service. It is a key component of a company's business model. It primarily identifies what product or service will be created in order to generate revenues and the ways in which the product or service will be sold.

An organisation may have multiple revenue streams, which refer to the money an organisation generates from each customer segment. In establishing the revenue model of a business, the combinations of revenue streams are particularly important because they enable the company to take advantage of all positive effects of the involved revenue streams. Figure 12 lists several different revenue stream models that can be taken into account.

Free (or nearly free) for the user: • Early exit strategy • Pay-what-you-want (PWYW) • Tip jar/donation • Freemium model • Barter or Swapping for services • Barter or Swapping for products	**Broker/Matchmaking** • Commission-based model • Auction model
Paid (direct sales business model): • Subscription model • Premium model • Pay-per-use model • Add-ons/In-app purchases • License fees • Single purchase model • Pay-as-you-go model (PAYG)	**Mixed business model:** • Razor and blade model • Crowdfunding • Open source model • No frills model (discount or budget model)
Third Party options: • Advertisement (Ad-based) model • Affiliate/Referral fee • Get-one-give-one model (G1G1) • Franchise model	

Source: https://www.boardofinnovation.com/business-revenue-model-examples/

Figure 12: Revenue Stream Types

With a focus on multisided marketplaces/platforms, it can be said that free or nearly free revenue stream types like freemium model are relevant. Freemium is a combination of the words free and premium. It describes a business model in which you give a core product

away for free to a large group of users and sell premium products to a smaller fraction of this user base. A known example of a freemium business model is Skype which provides free computer to computer calling and sells premium products in the form of voicemail, conference calls and worldwide connection to landlines and mobile phones.

Freemium business models play a key role in business today. They are widely use in a range of industries. An example of this is that apps with freemium models account for 98% of the revenue in googles app store and 95 % in apples app store.

One key objective of freemium is to attract new users. If this is not achieved, it probably means that free offerings are not compelling enough and/or there is a need to provide more or better features for free. If the free offering generates significant traffic, but few users are paying to upgrade, the free offerings are too rich, and should be cut back. Thus getting an adequate conversion rate that balances both objectives: ie attracting new users and generating upgraders.

In addition to this, an adequate pricing model must be designed according to the set of premium products and services. Usually a combination of some of the models presented in previous subsections, eg subscription, will be used to generate the target revenue of the freemium business model.

Multisided marketplaces/platforms providers' revenue streams are typically based upon subscription fees. A more and more significant portion is generated indirectly by claiming a revenue share from developers of third-party products and services.

The frequency or amount of revenue generation is another variable to take into account when talking about revenue streams. In this sense, revenues can be recurring (predictable revenue that can be expected to continue in the future) or non-recurring (eg a license fee, which is typically a one-time remuneration). More and more companies secure additional constant revenues by offering other services, such as continuous training and certification, along with the core value proposition. Both criteria (direct/indirect revenues and its frequency) must be taken into account when defining the appropriate model.

Open Source is also a key feature in vf-OS. It refers to something people can modify and share because its design is publicly accessible. In this sense, open source software is software with source code that anyone can inspect, modify, and enhance. Open Source promotes free redistribution of its source code, hence it is also called free software. Open sourcing commercial software poses many challenges, the biggest of which revolves around the meaning of 'FREE'. At the heart of the matter is the need to release code for free vs. protecting existing business interests, staying ahead of competition, and allowing customers to own their commercial deployments. As they do with proprietary software, users must accept the terms of a license when they use open source software. Open source licenses affect the way people can use, study, modify, and distribute software.

But Open Source doesn't just mean something is free of charge. Open source software programmers can charge money for the open source software they create or to which they contribute. But in some cases some programmers find that charging users money for software services and support (rather than for the software itself) is more lucrative. This way, their software remains free of charge, and they make money helping others install, use, and troubleshoot it. There are various ways to make money while developing Open Source software, including:

- Providing integration and support services
- Selling subscriptions to updates and support
- Selling proprietary components to segments of the user base

- Selling premium plug-ins, applications, services and themes
- Selling hosting services (ie Software as a Service model)
- Selling the software under a commercial license and releasing the code under an Open Source license simultaneously, aka Dual licensing (MySQL)

As Open Source include several revenue stream types based where the previous types, only subscription, transaction, license, and revenue sharing, are considered as main models in this deliverable.

In the context of a cloud-based platform developed with an Open Source philosophy, the revenue streams considered more appropriate for vf-OS providers are:

- Subscription
- Transaction-based (pay per use)
- License
- Revenue Sharing (commission)

4.1.1 Subscription

Subscriptions (also referred to as fees per term) are fixed instalments that, for example, a cloud-based platform consumer pays for getting access to, and using a certain services of the platform, for the duration of that subscription.

For both Platform as a Service (PaaS) consumers and providers, this approach provides a dependable basis for calculating costs and revenues and for planning service levels and capacity. In this model, the consumers do not have to make a repeated large purchase per year and the business does not have to solicit any orders from existing customers. This adds to the productivity of the business, since the time and effort required in order a generation is saved. The goods and services are delivered to the customer as and when they are required. The customer knows the payable amount in advance and this enables them to plan their budget well. The revenue predictability of the business also increases and the business can gauge where it’s headed. A subscription business model offers much higher payment safety for the business. It helps in realizing automatic online transactions to receive regular payments for the products and services sold. This ensures not just regularity of business but also the cash being generated from it.

An additional advantage is that the model gives an opportunity to get upgrades and increased revenue opportunities from the existing customers. Knowing the customers for a long period of time and solving their queries helps in coming up with solutions more suitable for the business. As a consequence, the business gets consistent opportunities in growing the revenue.

The subscription fee is a good choice if the provided value is high and a user will engage in several transactions since it also spreads the cost. Related to that is managing the customer relationship to ensure that the automated processes are satisfactory to customers given the price of the service. For subscriptions, the price is central to the customer, and each customer must think they are getting greater value than the recurring payment they make.

To succeed with the subscription business model, companies should not only focus on price, but also on cost. To remain profitable, it is critical that the company's costs remain low relative to price. Once subscription revenue exceeds costs, the company is generally free to grow unfettered. That is why one of the challenges with this model is that a mandatory payment might discourage users from signing up. One way to get around this

could be to offer heavy discounts for early adopters, or even lifting the fee completely to build the initial user base.

4.1.2 Transaction-based (pay per use)

Whereas subscriptions do not (or only partially) consider a consumer's platform usage, user-based transactions are typically based on the usage behaviour of the platform consumer. It establishes ties between usage and payment.

This model encourages more suppliers to join the platform and thus increases the liquidity of the marketplace's supply for two reasons: first, when the upfront fee is taken away it is easier to join, and second, if you only charge when a sale is made, you lower the supplier's risk of losing money. A transaction fee model also scales well: the more sales the platform generates, the more revenue it brings in.

Software pay per use is one of the contemporary trends to enter the market with the advent of cloud computing & Software as a Service (SaaS). The concept of cloud network implies the offering of a range of services by a third-party provider using the internet. The services offered includes SaaS, wherein the virtual server provides the usage of the software through the cloud. Combining these two notions, pay per use and SaaS, results in the offering of pay per use SaaS (Software as a Service), which essentially entails paying a small subscription fee on a per usage basis every time the software is being used.

In reference to the benefits of SaaS pay per use for Enterprises, it can be said that the model is primarily dependent upon certain market conditions, such as higher potential for piracy, lower inconvenience costs, majority of marginal users, and strong cloud network presence. Whereas perpetual licensing is important for heavy users, a market having the above-mentioned conditions will always benefit the SaaS Pay per Use model. So, while the developer finds advantages of an increased authorized user network, lower costs of marketing, enhanced customer reliability, and lesser impact of piracy etc; the rewards for users are even greater as they get to use the licensed full and updated versions for a small fee, even for minor everyday usage without incurring the huge expenditure on acquisition.

The capacity to provide full and updated functionality at an affordable per-use fee permits vendors to reach a wide spectrum of users leading to increased economic productivity. Thus, software pay per use model is a likely winner in this context provided the basic infrastructure for a strong network is in place.

4.1.3 License

A licensing agreement is a partnership between an Intellectual Property (IP) rights owner (licensor) and another who is authorized to use such rights (licensee) in exchange for an agreed payment (fee or royalty).

Licensing is most commonly applied to innovations that involve sophisticated technology protected by IP agreements. The innovation itself may not be a complete product, and may need to be integrated into a broader offering in order to create value for the end user. Very common among software companies, this model is however losing its sheen due to the emerging trend of SaaS subscription models.

Licensing could be for usage, which is the model for IP (patents, copyrights, trademarks). This type of license is usually limited by time, territory, types of products, volume, etc. The other kind is for certification, such as the McAfee SECURE trustmarks used for Internet websites.

Licensing revenues can be structured in different ways, with upfront payments by the licensee or with payments that are revenue-dependent. An important consideration in structuring licensing agreements is the portion of income derived from licensing revenue versus that deriving from royalties. Royalty revenue is dependent on the selling ability of the party integrating the licensed technology, and the size of the addressable market for the end-product.

Strategically, licensing may run the risk of exposing IP to the party integrating the technology into their products. It is therefore important to ensure that patents are defensible and that other IP is protected.

Another disadvantage to issuing a license is that it creates competition. In fact, the license places your competition on a level playing field because the competitor now has the right to use the same production processes you use. A licensing company may attempt to limit competition by limiting the scope of the license as much as possible. For example, the license may contain geographic, time or quantity restrictions that protect the market of the licensing company.

4.1.4 Commission

The most popular model for modern marketplaces is to charge a commission from each transaction. Thus, PaaS providers can request a commission or revenue share for placing and promoting an application that was developed by an individual software vendor onto the platform. The biggest benefit of this model is that providers are not charged anything before they get some value from the marketplace. This is really attractive for the providers.

The biggest challenge in getting the commission model to work is to provide enough value for both the customer and the provider especially at the start. Another challenge with the commission model is pricing strategy: How big should be the commission; it should be the same for all users or not; it should be charged the customer, the provider, or both; the first commission should be lower to get people to join the platform and raise it later on or not, etc.

The general recommendation is to use the commission model as the main revenue stream whenever feasible. There are scenarios in which it is not feasible for the marketplace to facilitate payment transaction. In these cases, the commission model does not work, for example, when:

- The size of the typical transaction is huge
- The marketplace has lots of different types of offerings
- The invoicing process is too complex for the marketplace to facilitate it. This is common in business-to-business (B2B) and some business-to-consumer (B2C) marketplaces
- Money is not exchanged at all in the marketplace. In these cases, you need a different type of business model

4.2 Pilot Providers Revenue Stream Scenarios

Different revenue streams can be applied to support different strategies. Thus, each provider focused on in this section (Platform Providers, Software Developers, Manufacturing and Logistics Solution Providers and External Service Providers) will have to evaluate and choose the optimal combination of revenue streams when defining their model.

The revenue model that potentially can apply to a provider type depends on its specific role in a use case scenario. In the following subsections, each vf-OS pilot use case is analysed regarding the providers' role and the before mentioned revenue stream models. Figure 11 presents the revenue models that apply to each provider type according to this analysis.

Revenue Stream Models	Platform Providers	Software Developers	Manufacturing and Logistics Solution Providers	External Service Providers
Subscription	✓	✓		✓
Transaction-based (pay per use)	✓	✓	✓	✓
License			✓	
Revenue Sharing (commission)			✓	

Figure 13: Revenue Stream and Provider Types in Pilot Use Cases

4.2.1 Pilot 1: Manufacturing and Logistics – Automation

The Manufacturing and Logistics pilot use case presents a Business to Business (B2B) collaborative approach involving also spare-parts suppliers as third party companies. The key issues for this pilot are:

- Safe inter-company communication
- Real time PLC data collection and storage in the platform
- Data storage service in the platform
- Specific purpose customised vApps
- Remote access to data

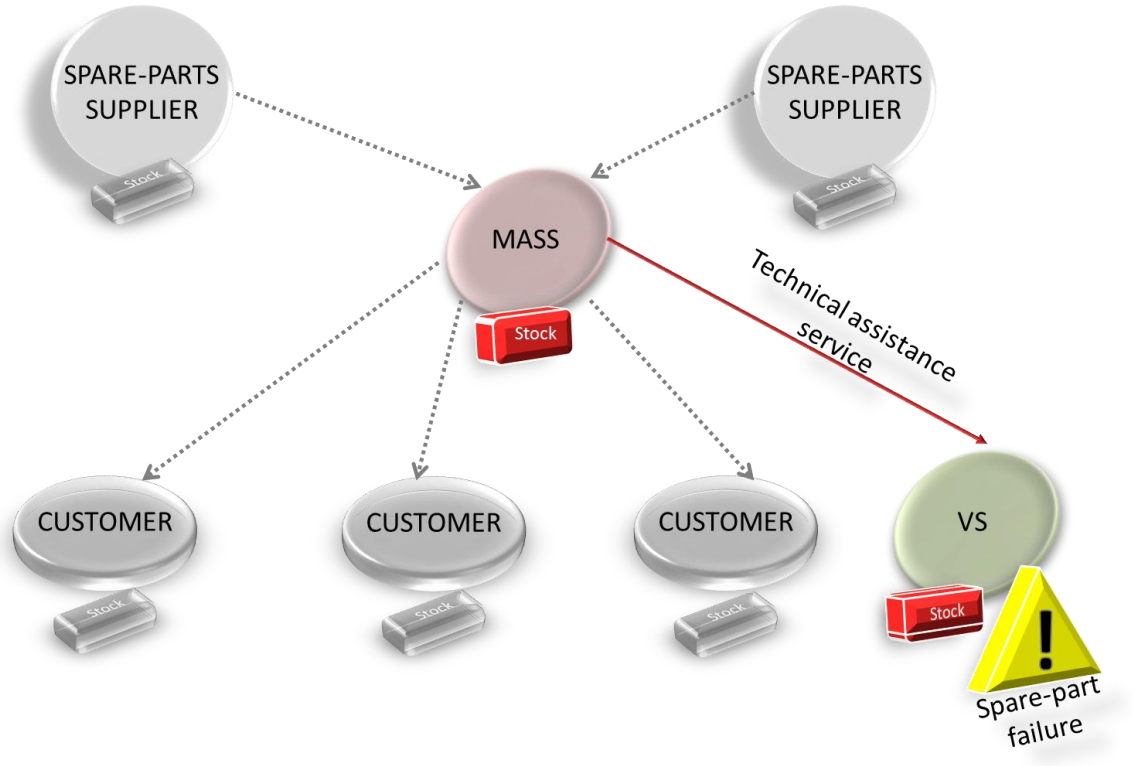


Figure 14: Manufacturing and Logistics Pilot Scheme

The main roles and revenue stream types of providers in this use case are:

- **Platform Providers:** Implement vf-OS environment at the use case customer's IT infrastructure and also connecting it to legacy software. Transaction based approach seems to be the first candidate revenue stream type, but subscription can also play a relevant role when upgrading becomes critical.
- **Software Developers:** To produce and to sell customised vApps required in the pilot use case. In this case, transaction based approach is the main option since it offers continuously reoccurring revenue although a license model could work well also. Also a recurring revenues scheme should be considered when upgrading becomes a relevant issue for the customer.
- **External Service Providers:** To provide services such as safe communications, hosting, storage, connected cloud services, etc. A combination of subscription and transaction based schemes should be considered. Subscription could be the base for recurring services such as safe communications and remote access, while transaction based could fit storage services depending on space requirements of the use case.
- **Manufacturing and Logistics Solution Providers:** provide their ICT interfaces and manufacturing connections to implement the customised vf-OS environment assuring connection to legacy systems. These providers are linked to Platform Providers because their "product" provides services to applications via the platform. A revenue sharing scheme or a licensing agreement with Platform Provider can apply in this case.

4.2.2 Pilot 2: Construction – Industrialisation

The construction pilot presents a use case where different actors of a value chain collaborate in a business process. The key issues for this pilot are:

- Safe document management
- On-line documents processing
- Document storage service in the platform
- Remote access to project documents
- Real time project monitoring and workflow management
- Real time quality control and feedback

Figure 15: Construction Pilot Scheme

The main roles and revenue streams types of providers in this use case are:

- **Platform Providers:** Implement vf-OS environment at project supervisor IT infrastructure and also connecting it to legacy software. At a first glance, a specific implementation is required in both companies. Transaction based approach seems to be the first candidate revenue stream type, but subscription can also play a relevant role when upgrading becomes critical because collaboration among companies will go beyond one specific project.
- **Software Developers:** To produce and to sell customised vApps required in the pilot use case. In this case, transaction based approach is the main option since it offers continuously reoccurring revenue although a license model could work well also. Also a recurring revenues scheme should be considered when upgrading becomes a relevant issue for the customer.
- **External Service Providers:** Provide services such as safe communications, hosting, storage, connected cloud services, etc. This provider profile is likely to play a major role in this type of one-off settings. A transaction based scheme should be considered as the most relevant.
- **Manufacturing and Logistics Solution Providers:** Provide their ICT interfaces and manufacturing connections to implement the customised vf-OS environment assuring connection to legacy systems. The role of these providers would very likely be linked

to the project's workflow management service provider who would require these ICT interfacing components to implement the solution. A revenue sharing scheme linked to a transaction based approach of the service provider can apply in this case. A licensing agreement with Platform Provider could also be appropriate.

4.2.3 Pilot 3: Manufacturing Assembly: Collaboration

This pilot presents a use case where two complementary manufacturing actors collaborate in industrialisation and production processes. In a first instance, project workflow management approach is required for industrialisation phase. Then a production workflow monitoring and supervision is a more appropriate scenario for the production phase. So, this pilot includes issues regarding both of the previous pilots:

- Safe document management
- On-line documents processing
- Remote access to project documents
- Safe inter-company communication
- Real time production data collection and storage in the platform
- Specific purpose collaborative production planning vApps

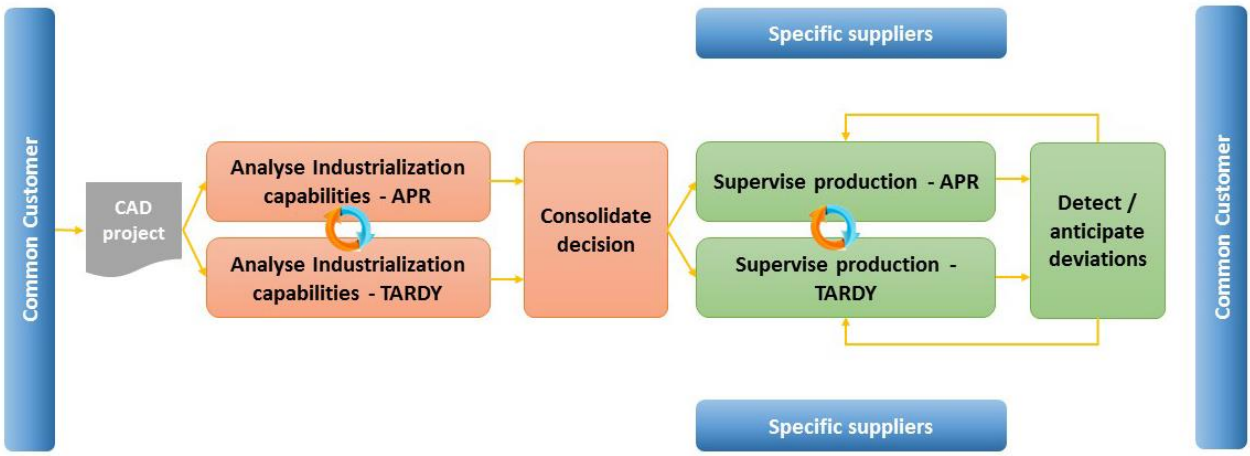


Figure 16: Inter-Company Synchronisation Pilot Scheme

The main roles and revenue streams provider type in this use case are:

- **Platform Providers:** Implement vf-OS environment at project supervisor IT infrastructure and also connecting it to legacy software. At a first glance, each project would require a specific implementation with its corresponding database and document templates. In consequence, a transaction based approach seems to be the best candidate revenue stream type.
- **Software Developers:** To produce and to sell customised vApps required in the pilot use case. In this case, transaction based approach is the main option since it offers continuously reoccurring revenue although a license model could work well also. Also a recurring revenues scheme should be considered when upgrading becomes a relevant issue for the customer.
- **External Service Providers:** Provide services such as safe communications, hosting, storage, connected cloud services, etc. As data storage is expected to be based on on-site ERP systems, subscription could be the base for recurring services such as safe communications and remote access. No major transaction based revenues are expected apart from some implementation and training services.

- **Manufacturing and Logistics Solution Providers:** Provide their ICT interfaces and manufacturing connections to implement the customised vf-OS environment assuring connection to legacy systems. These providers are linked to Platform Providers because their “product” is a platform component. A revenue sharing scheme or a licensing agreement with Platform Provider can apply in this case.

5 Conclusion

The vf-OS Provider Scenario Characterisation provides an overview about the different vf-OS providers and their activities. The providers are extracted from the previous DOA and “D1.1 Vision Consensus”. Each provider extends the vf-OS environment in a different way and vf-OS itself provides different interfaces for each provider. The provider roles in vf-OS are as follows:

- Platform Providers
- Software Developers
- External Service Providers
- Manufacturing and Logistics Solution Providers

The key point of the description of these provider roles are the mappings between the providers and the interfaces provided by vf-OS. It turned out that each provider has its own way of using and accessing vf-OS with various functionalities.

As a further result of the definition of the providers and their activities, scenarios are written in a narrative way directly related to vf-OS. Especially the defined provider scenarios are designed to extract requirements from the providers’ point of view. Furthermore, the providers are put in relation with the pilot scenarios of “D1.2 User Scenarios Characterisation” to cover not only vf-OS, but also the user point of view. This results in an analysis, where additional information is provided, that will be reflected D1.3 and D2.1.

Finally, the revenue streams of the providers are described via generic models and after that, those revenue stream models are applied to the different user scenarios test the revenue models for applicability in a real world.

This document will be used as a foundation for “D1.5 Requirements Specification” for the providers’ point of view and “D2.1 Global Architecture Definition” to arrange the different provider interfaces.

Annex A: History

Document History	
Versions	V0.1: First draft produced by ALM V0.2: Section 1 completed V0.3: Section 2 completed V0.4: Restructured document with new a new section arrangement V0.5 Editor provided content and examples to guide partners for remaining sections V0.6 Section 1, 2, 3 and 4 consolidated V0.7 Section 1 and 2 restructured, content adjusted V0.8 Document completed and ready for pre-review V0.9 Document completed and ready for review V1.0 3rd Reviewer version V1.0.1 3rd Reviewer version comments fixed
Contributions	ASC: • Danny Pape: Main editor, second draft, Input to all sections, and multiple comments • Tobias Hinz: Main editor, second-draft, Input to all sections ALM • Andries Stam: Input to Sections 1.1., 1.2, 2.1.1, and 2.2.1 ICE • Stuart Campbell: Input to Sections 2.1.3, 2.2.3, multiple comments, and peer review • Rebecca Campbell: Final English Review • Oscar Garcia: Input to Sections 2.1.5, 2.2.5, and multiple comments CMS • Carlos Coutinho: Input to Sections 2.1.4 and 2.2.4 IKER • Juan-Maria Goenaga: Input to Section 4 • Eduardo Saiz: Input to Section 4

Annex B: References

No references

www.vf-OS.eu