www.forgetit-project.eu

ForgetIT

Concise Preservation by Combining Managed Forgetting and Contextualized Remembering

Grant Agreement No. 600826

Deliverable D8.1

Work-package	WP8: PoF Reference Model and Framework
Deliverable	D8.1: Integration Plan and Architectural Approach (V2 - Amended Version)
Deliverable Leader	Francesco Gallo (EURIX)
Quality Assessor	Heiko Maus (DFKI)
Estimation of PM spent	9
Dissemination level	PU
Delivery date in Annex I	31 October 2013 (M9)
Actual delivery date	06 December 2014 (v1), 25 August 2014 (v2)
Version	v2
Revisions	10
Status	Final Release
Keywords:	ForgetIT Architecture, Preserve-or-Forget Framework, Integration Plan, Components

Disclaimer

Changelog for version v2

With respect to version v1 (originally delivered D8.1) the following editorial changes have been applied:

- the architecture layers have been reduced to three (Active Systems, Preserve-or-Forget (PoF) Middleware, Preservation System), the updated diagrams are in Section 4; the Preservation System now includes the Digital Repository (formerly referred to as the Archive) and the Cloud Storage Service, Section 2.3 has been renamed accordingly;
- Section 5 about the integration plan has been anticipated and placed after Section 4, a new Table in Section 5 summarizes the planned integration of the various components for each Preserve-or-Forget (PoF) Framework release;
- the adoption of OAIS in the ForgetIT architecture is discussed in Section 2.3.1;
- the evaluation of OAIS and middleware solutions is reported in the new Section 7: a Table containing a list of assessment criteria for OAIS platforms and the selection of the best candidate according to the proposed criteria have been added.

Changes above have been suggested after the first annual project review.

List of Authors

Partner Acronym	Authors
LUH	Nattiya Kanhabua, Kaweh Djafari-Naini, Claudia Niederée
LTU	Parvaneh Afrasiabi Rad, Ingemar Andersson, Göran Lindqvist, Jörgen Nilsson
IBM	Ealan Henis, Simona Rabinovici-Cohen
DFKI	Heiko Maus, Frank Steinmann
CERTH	Vasileios Mezaris, Olga Papadopoulou, Vasilis Solachidis
dkd	Olivier Dobberkau, Phuong Doan
USFD	Mark Greenwood
EURIX	Walter Allasia, Francesco Gallo, Jacopo Pellegrino

Contents

Executive summary

The primary objectives of WP8 are (a) to devise a reference model which comprises the concepts and processes for the managed forgetting, the contextualized remembering and the integrated information and preservation management (synergetic preservation), and (b) based on this reference model, to integrate components implementing these concepts into a technologically coherent framework, the Preserve-or-Forget (PoF) Framework.

This document describes the architecture of the PoF Framework with its main integrated components developed by the technical WPs (WP3-WP7). The integration plan for testing and validating the project results is described, too. The results presented in this document have been produced within Task 8.1.

Making use of Model Driven Architecture (MDA) approach and UML notation, under the lead of EURIX, all technical partners have worked together to establish and document a first version of the functional specifications of the ForgetIT architecture and the corresponding integration guidelines for the different components.

The architecture defined in the present document will be improved in an iterative approach, to be refined and extended during the project when new results and insight become available for integration and testing.

The defined architecture includes the Active Systems, the Preservation System and the Preserve-or-Forget (PoF) Middleware. The Active Systems represent the user applications, in this project two applications will be developed and tested: the Semantic Desktop (WP9) and TYPO3 (WP10). The Preservation System, responsible for the preservation of the content within the framework, is made up of a Digital Repository (WP8) and a Cloud Storage Service (WP7), with a Storlet Engine for executing specific tasks close to the data. Finally, the PoF Middleware provides the bridge between the Active Systems and the Preservation System and will integrate components developed by WP3-WP6, which implement the core ForgetIT principles.

Structural and behavioral diagrams for the overall architecture have been defined. Dynamic diagrams representing two priority workflows for basic managed forgetting and synergetic preservation are provided. Based on component diagrams and descriptions, an integration plan for the three PoF Framework releases has been outlined. The validation of the first framework prototype will be tested using the two priority workflows defined.

The implementation of the Digital Repository and the PoF Middleware will leverage, wherever possible, existing solutions and technologies, which will be adapted and customized within WP8. A preliminary assessment of OAIS solutions for the Digital Repository and of different technologies and platforms for the PoF Middleware has been included in this document.

1 Introduction

The primary objective of WP8 is to define - in collaboration with the other technical WPs as well as with the interdisciplinary components from WP2 - a reference model which comprises the concepts and processes for *managed forgetting*, *contextualized remembering* and *synergetic preservation*. As support for this so-called Preserve-or-Forget (PoF) Reference Model, the second objective is to integrate the components developed in the project into a technologically coherent framework implementing the ForgetIT model.

The present document describes the architecture of the PoF Framework and is the outcome of Task 8.1 - *Integration plan and ForgetIT architecture*, whose main objective is to ensure that (1) the components developed in WP3-WP7 work together and that (2) the conceptual architecture for the PoF framework will be specified, including component responsibilities, interface definitions and the foreseen interplay between components (protocols and processes). Task 8.1 also targets the plan for the integration of software components into the ForgetIT architecture in order to test and validate the project results.

The PoF Framework architecture discussed here currently includes the *Active Systems*, the *Preservation System* and the *PoF Middleware*. The overall architecture will be refined using an iterative approach during the project.

Two Active Systems are developed and tested in the project, namely the Semantic Desktop and TYPO3. The Preservation System includes a Digital Repository and a Cloud Storage Service (leveraging Storlet technology) for executing specific preservation tasks close to the data. The Preservation System implements several features which are typically provided by a digital archive and is based also on the OAIS model [1]. The PoF Middleware implements the ForgetIT intelligent preservation solution and provides the bridge between the Active Systems and the Preservation System.

Wherever possible, the implementation of the Preservation System and PoF Middleware will rely on existing solutions and technologies, which will be adapted and customized within WP8. The PoF Middleware will integrate components developed by WP3-WP6 for realizing the novel ForgetIT methods, while the cloud storage is developed in WP7.

Other project deliverables provide relevant input to the present document. In particular, D3.1 [2] discusses the importance of managed forgetting and provides useful guidelines for the PoF Framework. Other deliverables, such as D4.1 [3], D5.1 [4], D6.1 [5] and D7.1 [6], describe the foundations and the state of the art for relevant topics such as information extraction, synergetic preservation, contextualization and computational storage services. Based on the results provided by these documents, several components will be developed during the project lifetime, to be further integrated in the PoF Framework, as discussed in the following.

Three releases of the PoF Framework are planned: for each component in the architecture we provide a plan for its integration mapped to the three releases. For the first release two priority workflows have been identified: they will be used as a reference to validate the adopted approach with end-to-end processes, involving all components in the PoF

Framework. The reference workflows used as validation of early integration are based on D9.1 [7], which introduces the two main application use cases and scenarios, which have been taken into account when designing the architecture.

This document also provides input to other deliverables, not only the WP8 ones: D3.2 [8], D4.2 [9], D6.2 [10] and D7.2 [11] for the first prototypes of several components, D5.2 [12] for a workflow model for the transition between Active Systems and the Preservation System, based on the architecture described here.

After discussing the architecture and the integration approach, we anticipate the results of the preliminary assessment of candidate solutions which could be used for implementing the Middleware and the Digital Repository. The adopted solutions and their integration will be detailed in D8.3 [13].

For the PoF Middleware, we investigated different open source solutions: based on the results from WP5, a message oriented approach has been chosen, since this is a well established solution for enterprise application integration.

Concerning the Digital Repository, we started from popular open source platforms based on OAIS specification, since this is the reference in the digital preservation community. Nevertheless, Task 8.2 will provide the PoF Reference Model, which will extend the current OAIS specification to support ForgetIT principles (see deliverable D8.2 [14]).

Moreover, the PoF Framework will include the preservation aware storage developed by WP7, so the assessment of the OAIS platforms has been focused on digital repositories and preservation platforms which, in order to implement the whole Preservation System, could also be easily integrated with the cloud storage engine provided by WP7 .

The document is organized as follows: in Section 2 we provide an overview of the overall architecture, with the main components, described in Section 3; in Section 4, making use of UML notation, structural and behavioral diagrams for the PoF Framework are provided, as well as two priority workflows for early integration; in Section 5 we discuss the integration plan for the different framework releases; in Section 6 we describe the testbed environment for the integrated components; in Section 7 we first describe and evaluate candidate solutions compliant to OAIS for the Digital Repository (Section 7.1) and then provide a preliminary overview of candidate technologies for the PoF Middleware (Section 7.2); finally, we provide a list of acronyms and abbreviations used in the document.

2 ForgetIT Architecture

The architecture of the PoF Framework is the first outcome of WP8. The main purpose was to design a system to integrate all components developed in the project into a technologically coherent framework which could be used to implement the PoF Reference Model. In order to achieve this, the expected results of each technical WP have been evaluated and a collaborative approach has been adopted to design the overall architecture.

The method used to design the architecture is based on a Model Driven Architecture (MDA) approach, adopting UML as the standard modeling language for designing the architecture, from preliminary sketches to final representation, using an iterative approach.

The ForgetIT architecture is made up of three layers:

- *Active Systems*
- *Preserve-or-Forget Middleware*
- *Preservation System*

which are represented in Figure 1 and are shortly described in the following.

The first layer, the Active Systems, represents user applications. In the context of ForgetIT, two applications will be developed, integrated and tested: the Semantic Desktop (WP9) and the TYPO3 CMS (WP10). These two applications are related to the main scenarios, as already described in D9.1 [7].

The second layer is provided by the PoF Middleware, whose main purpose is to enable seamless transition from Active Systems to the Preservation System (and vice versa) for the synergetic preservation, and to provide the necessary functionality for supporting managed forgetting and contextualized remembering. The middleware provides the integration framework for all components developed in WP3-WP6.

The third layer is the Preservation System, composed by two sub-systems: a Digital Repository and a Cloud Storage Service. The Preservation System provides both content management and typical archive functionalities, inspired by the OAIS model [1]. In Section 2.3.1 the motivations for this model in the ForgetIT architecture are briefly discussed.

The Preservation System is responsible for the digital preservation of the contents created by the applications and must provide required functionalities for the synergetic preservation. The Preservation System provides not only ingest and access functionalities, but also data management of the archived content, data curation and content storage. The development and integration of the Digital Repository and of the Preservation System is under the responsibility of WP8, while the cloud storage is the main outcome of WP7.

The approach adopted in ForgetIT to implement the OAIS Preservation Planning and Archival Storage functionalities makes use of an advanced cloud storage system powered by a Storlet engine, a mechanism to execute resource consuming tasks close to the data.

The architecture components are discussed in more detail in the next Sections. The information reported for all architecture blocks clearly demonstrates that all technical WPs will contribute to the integrated framework and that the designed architecture can accommodate all components in a coherent way.

The integration mechanism when moving from one layer to the next has been defined in terms of interfaces and protocols (for some components this is still under development), as described in Section 3 and Section 4.

An overview of the PoF architecture is depicted in Figure 1, which provides a graphical representation of the main components with some descriptive information (components developed during the project are shown in green, components shown in blue or cyan belong to existing platforms which will be further developed and customized to fit with project purposes and for integration in the overall architecture). The UML diagrams of the architecture can be found in Section 4.

2.1 Active Systems

The PoF architecture includes two user applications which are used to validate the main scenarios. One is a personal preservation application based on the Semantic Desktop, developed in WP9, and the other one is an organizational preservation application based on the TYPO3 CMS, developed in WP10 (see D9.1 [7]). Both systems are complex and include several components, the internal details are already discussed elsewhere (see deliverables D9.2 [15] and D10.1 [16]). In this document we focus only on the features which are relevant for integration.

Two example workflows involving the two applications as well as functionalities of other ForgetIT components are described in Section 4.2. Concerning the interfaces and the integration with the other architecture components, the two applications will make use of REST APIs provided by the PoF Middleware to notify the system about content to be preserved and to retrieve updated content.

Both components will expose a CMIS [17] interface to allow the middleware components to retrieve the content. The communication with the middleware is provided by application specific adapters, which encapsulate application-specific extensions for interacting with the PoF Middleware. The separation of functionalities that will be part of the Middleware and those which will become part of the Active Systems is a challenging task. The details of this separation are still under discussion.

Further details about the two Active Systems can be found in Section 3.1, where the Semantic Desktop and TYPO3 are described in detail in Table 1 and Table 2, respectively.

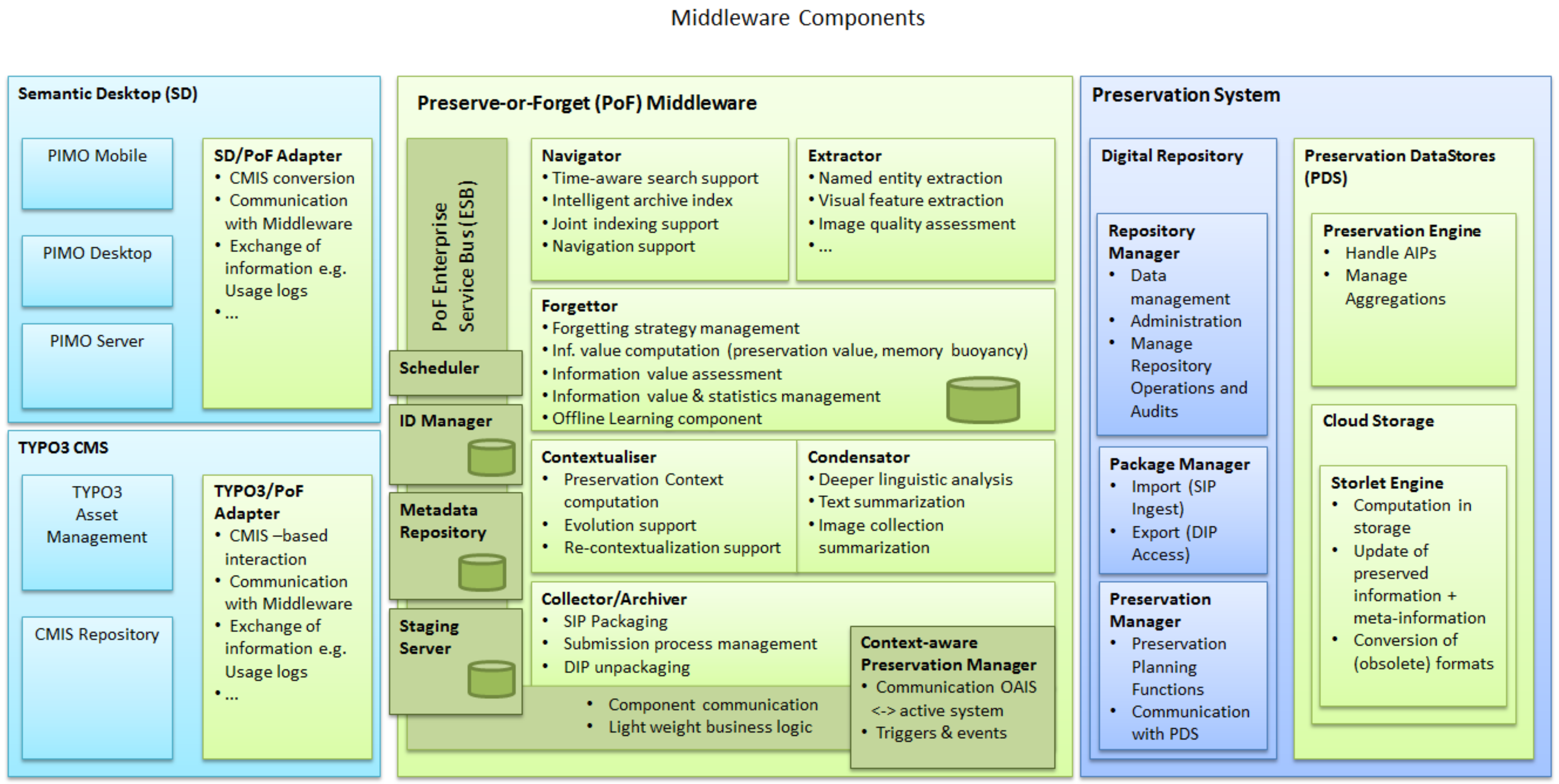
Middleware Components
Semantic Desktop (SD)
PIMO Mobile
PIMO Desktop
PIMO Server
SD/PoF Adapter
• CMIS conversion
• Communication with Middleware
• Exchange of information e.g. Usage logs
• ...
TYPO3 CMS
TYPO3 Asset Management
CMIS Repository
TYPO3/PoF Adapter
• CMIS –based interaction
• Communication with Middleware
• Exchange of information e.g. Usage logs
• ...
Preserve-or-Forget (PoF) Middleware
PoF Enterprise Service Bus (ESB)
Scheduler
ID Manager
Metadata Repository
Staging Server
Navigator
• Time-aware search support
• Intelligent archive index
• Joint indexing support
• Navigation support
Extractor
• Named entity extraction
• Visual feature extraction
• Image quality assessment
• ...
Forgettor
• Forgetting strategy management
• Inf. value computation (preservation value, memory buoyancy)
• Information value assessment
• Information value & statistics management
• Offline Learning component
Contextualiser
• Preservation Context computation
• Evolution support
• Re-contextualization support
Condensator
• Deeper linguistic analysis
• Text summarization
• Image collection summarization
Collector/Archiver
• SIP Packaging
• Submission process management
• DIP unpackaging
• Component communication
• Light weight business logic
Context-aware Preservation Manager
• Communication OAIS <-> active system
• Triggers & events
Preservation System
Digital Repository
Repository Manager
• Data management
• Administration
• Manage Repository Operations and Audits
Package Manager
• Import (SIP Ingest)
• Export (DIP Access)
Preservation Manager
• Preservation Planning Functions
• Communication with PDS
Preservation DataStores (PDS)
Preservation Engine
• Handle AIPs
• Manage Aggregations
Cloud Storage
Storlet Engine
• Computation in storage
• Update of preserved information + meta-information
• Conversion of (obsolete) formats

Figure 1: Overview of PoF architecture (*green*: components developed during the project; *blue*: existing components to be improved and customized).

2.2 PoF Middleware

The main purpose of the PoF Middleware is to enable the three main principles of ForgetIT, namely a seamless transition from active systems to the archive (Synergetic Preservation), a meaningful transition back to the active system (Contextualized Remembering), support for the processes such as information value assessment required for resource selection and forgetting (Managed Forgetting).

The PoF Middleware integrates components for feature extraction, contextualization, condensation and managed forgetting. Other components are associated to common tasks and managed through the middleware bus. Examples of shared components are the `Scheduler` and the `ID Manager`. The integration of all components into a middleware supporting project scenarios is the main challenge for future WP8 integration activities.

The PoF Middleware will expose REST APIs to be consumed by the user applications and will include a CMIS client to retrieve content. Furthermore, the PoF Middleware includes components for importing and exporting content into and from the Preservation System. The PoF Middleware will be able to properly package the content and associated metadata into a format which is compliant to what is expected by the Preservation System.

Several approaches and technologies are available in the literature and also in the IT marketplace to implement complex middleware infrastructures for enterprise applications. The analysis of the most appropriate approach for the middleware component and for the transition between the Active Systems and the Preservation System is reported in deliverable D5.2 [12]. Here we just point out that the PoF Middleware must satisfy different requirements, such as flexibility for the integration of heterogeneous components and the capability to ease the communication among the components, implementing the concept of middleware bus. In addition to this, the middleware should enable the implementation of complex workflows corresponding to the core ForgetIT principles. The implementation of a Message Oriented Middleware (MOM) [18] seems to be a promising and well established approach to fulfill such requirements and will be discussed in deliverable D5.2 [12]. We provide a quick overview of popular middleware solutions in Section 7.2, their evaluation is still in progress. The solution adopted for the first release of the PoF Framework will be described in deliverable D8.3 [13].

2.3 Preservation System

The Preservation System is responsible for the preservation of digital content created by the Active Systems and provides the required functionalities enabling the synergetic preservation, for the smooth transition of content from the archive to the applications. The Preservation System exposes different APIs for ingest and access, depending on the actual implementation and must support the PoF Reference Model.

The Preservation System is also referred to as Archival Information System (AIS) in other project documents, to point out the content archival functionalities, although, as already stated, in the specific context of ForgetIT we need to go beyond the usual definition of archive based on OAIS specification and adopted in other digital preservation communities (see deliverable D8.2 [14]).

The Preservation System is composed by two integrated systems, a Digital Repository and the Cloud Storage System developed within WP7, described in the following Sections after shortly discussing the role of OAIS in the PoF architecture.

2.3.1 OAIS model in ForgetIT architecture

OAIS is currently the most recognizable conceptualization of a digital preservation system. Several initiatives and projects have promoted the adoption of OAIS concepts and terminology, which is often used in the digital preservation community as a conceptual model for discussion and comparison and is widely accepted as the reference standard for implementing digital preservation platforms (all main preservation solutions available, both commercial and open source, claim their compliance to OAIS). It is worth noting that OAIS does not endorse or recommend a specific implementation on any level and the specification itself allows additional services beyond those required, as well as an extensible information model.

For these reasons we included OAIS (and an OAIS comliant system) into the PoF Framework architecture to represent the core presrevation functionality. OAIS provides a functional model and a information model, and both of them will be evaluated during the project to define a new reference model going beyond the boundaries of OAIS (see deliverable D8.2 [14]).

The OAIS functional model is depicted in Figure 2. OAIS functional entities include *Ingest*, *Access*, *Data Management*, *Administration*, *Preservation Planning* and *Archival Storage* [1]. In the specific context of ForgetIT, Archival Storage is implemented by the Cloud Storage Service. Preservation Planning is also partially implemented by the cloud storage, which is actually a preservation aware storage system. Ingest and Access provide additional features related to the specific ForgetIT core principles, e.g. contextualization.

In addition to defining the parties involved in the long-term preservation of digital materials, OAIS provides also an information model for managing the digital materials as they pass through the system. A significant component of this model is the Information Package (IP). Each IP consists of the digital object(s) to be preserved, the required metadata and the Packaging Information which relates content and metadata (see [1]).

OAIS outlines three types of Information Package: Submission Information Package (SIP), Archival Information Package (AIP) and Dissemination Information Package (DIP). SIP and DIP are external to the archive and refer to the producer and consumer, respectively. AIP is internal to the archive.

In ForgetIT, the Active Systems (via the PoF Middleware) act both as producers and as consumers. It is common practice to adopt the same representation for SIP, AIP, and DIP

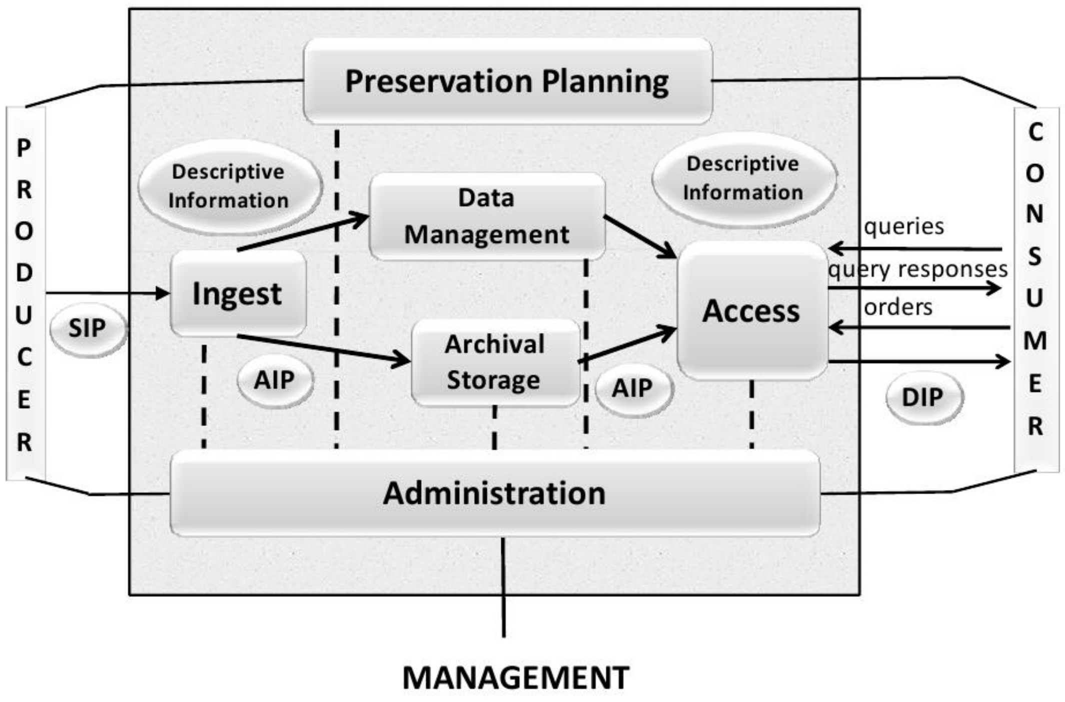

Figure 2: OAIS functional entities [1].

(e.g. several platforms use METS [19] as XML wrapper for the three of them, but this is not mandatory).

The SIP definition for ForgetIT is part of WP5 activities (see deliverables D5.1 [4] and D5.2 [12]), and has been taken into account for the implementation of PoF Middleware components responsible for content packaging.

2.3.2 Digital Repository

In the PoF Framework, the main role of the Digital Repository is to manage the content produced by the user applications, providing also additional archival features in cooperation with the Cloud Storage Service, exposing interfaces to import and retrieve content.

A Digital Repository is typically a software application for managing digital content and delivering the content to its consumer in convenient ways, it can be a single platform or even a set of applications. Several implementations exist, providing general purpose repository systems, supporting a variety of multimedia contents and usually embedding a suite of applications to support different needs of the users.

It is worth noting that the meaning of the term *digital repository* is widely debated and also the common understanding changed over time, from an initial focus on software systems to a wider and overall commitment to the stewardship of digital materials, which requires not just software and hardware, but also policies, processes, services, and people, as well as content and metadata. Moreover, the terms *digital repository* and *archive* are often used interchangeably. OAIS uses the term *archive* when referring to an organization that intends to preserve information for access and use by a so-called *Designated Community*, while ISO 16363:2012 standard - *Audit and certification of trustworthy digital repositories*, based on another CCSDS specification [20], prefers the term *digital repository* instead.

For the Digital Repository implementation, it is planned to rely on existing solutions. A list of candidate solutions is discussed in Section 7.1.

Further details about the Digital Repository functionalities and the role in the ForgetIT architecture can be found in Table 13.

2.3.3 Cloud Storage Service

Cloud storage services are growing in use and popularity, for both personal and business applications. Digital repositories often support cloud storage as one of the different storage options to backup and restore the digital contents. In the context of ForgetIT, the cloud storage system is used not only for storing digital contents, but also to perform specific tasks close to the data. This new paradigm is based on the concept of preservation aware storage and is discussed in deliverable D7.1 [6].

The Cloud Storage Service integrated in the ForgetIT platform is based on Preservation DataStores (PDS) and OpenStack Swift [21], discussed in deliverable D7.1 [6], provides storage resources for AIPs but also a Storlet engine for executing specific operations close to the data, i.e. future processing steps can be done in the archive without requiring to extract it to a server and put it back into the archive.

These tasks can include different content transformations, such as format migration, other resource consuming tasks, such as integrity checks as well as operations for enabling managed forgetting within the archive and operations for supporting context evolution for archived content.

Further information about the Cloud Storage Service can be found in Table 14, including details related to the internal components and the technologies adopted for the implementation.

3 Architecture Components

In this Section we describe the components of the overall architecture, split into three main blocks (Active Systems, PoF Middleware and Preservation System), as already discussed in Section 2. The overview of the PoF Framework is depicted in Figure 1, while the component diagram of the PoF Framework is shown in Figure 4.

The PoF Middleware components are divided into (a) shared components (performing common tasks) and (b) components implementing the core ForgetIT functionalities. Middleware components are managed through a Enterprise Service Bus (ESB) [18]: this is particularly relevant for components responsible for common tasks and is highlighted in Figure 1 and Figure 4. In the ForgetIT architecture, the ESB provides the communication layer for all components in the PoF Middleware, reducing the number of point-to-point connections between communicating applications [18].

For each component a fact sheet is provided, describing main expected functionalities and focusing on information which is relevant for the integration.

3.1 Active Systems: Semantic Desktop and TYPO3 CMS

The Semantic Desktop and the TYPO3 CMS are described in detail in WP9 and WP10 deliverables. Here we focus only on the internal components relevant to the integration along with the application-specific adapters. The main features of the two systems are summarized in Table 1 and Table 2.

3.2 Shared Components of the PoF Middleware

The PoF Middleware integrates several components. We identified four shared components, which are responsible for general tasks managed by the middleware bus:

- `Metadata Repository` (see Table 3)
- `ID Manager` (see Table 4)
- `Scheduler` (see Table 5)
- `Context-aware Preservation Manager` (see Table 6)

The components above can be used by other components for processes internal to the PoF Middleware or can provide functionalities used by the other layers: for example the `ID Manager` provides the mapping for all different identifiers associated to resources to be properly managed by the applications or by the Preservation System, while the `Scheduler` manages the different asynchronous activities which are executed in the PoF Middleware, activating specific workflows and tasks.

3.3 Middleware components supporting core ForgetIT functionality

In addition to the shared components, the PoF Middleware contains six other components, which implement core ForgetIT functionality (see Figure 1):

- `Forgettor` (see Table 7)
- `Extractor` (see Table 8)
- `Condensator` (see Table 9)
- `Contextualizer` (see Table 10)
- `Navigator` (see Table 11)
- `Collector/Archiver` (see Table 12)

Together with the Preservation System these components implement the three core ForgetIT principles of managed forgetting, contextualized remembering and synergetic preservation. For each component we describe the main features in a separate Table.

The integration of the components above leverages the ESB approach for the communication, adopting standardized interfaces and protocols for sharing information within the PoF Middleware. ESB provides several benefits, such as loose coupling among components, flexibility, scalability from point-solutions to enterprise-wide deployment, support for multi-protocol communication. For a detailed discussion see [18].

3.4 Preservation System: Digital Repository and Cloud Storage

The two sub-systems which constitute the Preservation System are described here separately. While the Cloud Storage Service will be developed within WP7, for the Digital Repository an existing solution will be adopted, since several open source implementations are already available and are supported by a large number of institutions and archives.

The Digital Repository, assumed to be compliant to the OAIS model but also supporting its extension in ForgetIT, is described in Table 13. The Cloud Storage System, providing a preservation aware storage, is described in Table 14. Together they implement the long term digital preservation of ForgetIT contents, empowered with specific functionalities to support smooth transition from and to the user applications through the PoF Middleware.

Component Name	**Semantic Desktop**
Partner Responsible	DFKI
Contributing Partners	
Work Packages	**WP9**, WP3, WP4, WP5, WP8
Reference Deliverables	D9.2, D9.3, D9.4
Current Status	Prototype is available.
Short description and role	The Semantic Desktop is a personal information management system with an underlying ontology semantically describing the user's mental model and the resources involved. This ontology is the Personal Information Model (PIMO). The Semantic Desktop infrastructure consists of a PIMO Server with a dedicated API so that any third party could use the PIMO for own purposes (e.g., using it as tagging vocabulary). In addition, a combination of plug-ins for (some) standard applications as well as dedicated components/UI for specific purposes (such as task management, photo collection) is provided. In ForgetIT the Semantic Desktop serves as a means to learn about user's resources, their usage over time, importance, interrelations, and context for each resource from the PIMO. Once the ForgetIT services are combined with the Semantic Desktop infrastructure, synergetic preservation is realized with nearly no additional effort. The PIMO will also provide context information for realizing contextual remembering as well as means for contributing to managed forgetting. The infrastructure will be enhanced with several ForgetIT services such as image quality assessment or text condensation.
Delivery Mode	Platform running in application server on dedicated virtual machine.
Subcomponents	PIMO Server, PIMO desktop clients, specific plug-ins for applications, HTML5 mobile client, User Observation Hub (UOH)
Main APIs, input and output formats	PIMO API description available in documentation, JSON RPC is used.
Plan for integration at M18	Platform connected to PoF Middleware; initial workflows are served.
Plan for integration at M27/M36	Iterative enhancement of interplay with PoF Middleware; concise preserving desktop client (M27); concise preserving mobile client (M36).
Language, runtime framework	Server: Java, JSP, Apache Tomcat, MySQL; Desktop: Java, HTML5 (JavaScript, CSS); Mobile App: HTML5
SW and HW Requirements	The Semantic Desktop can be deployed on a VM.
Dataset for testing	Stainer data set; 24/7 instance at DFKI (live usage); test instance for ForgetIT; cloning of PIMOs possible
License	BSD-compliant license for interfaces and PIMO model; implementation free use in ForgetIT
Notes	The prototype and some documentation can be found at `https://pimo.kl.dfki.de`.

Table 1: Semantic Desktop (Active System, WP9)

Component Name	**TYPO3 CMS**
Partner Responsible	dkd
Contributing Partners	
Work Packages	**WP10**, WP8, WP9
Reference Deliverables	D9.1, D10.1, D10.2, D10.3
Current Status	Design of pilot applications; evaluation of standards to be used; CMIS server for testing under development.
Short description and role	TYPO3 is an enterprise-class, open source CMS, used internationally to build and manage websites of all types, from small sites for non-profits to multilingual enterprise solutions for large corporations. TYPO3 is a user-friendly, intuitive tool for producing and maintaining web pages with just a few clicks of the mouse. Authors benefit from the full-featured rich-text editor that offers all of the formatting options they would need in a WYSIWYG tool with a familiar Word processor-like interface. Seamless integration of multimedia content and dynamic image manipulation are available right out of the box in TYPO3. In addition, an internal messaging and workflow system helps content creators and editors to collaborate in the administration back-end. TYPO3 provides an extremely detailed permissions system for implementing professional editing workflows for both users and groups. Administrators can even manage multiple websites in one TYPO3 installation and share users, extensions, and content between them.
Delivery Mode	Release in an agile approach, CMIS server published with Alfresco.
Subcomponents	CMIS, Semantic Annotations, ForgetIT TYPO3 Extensions (Content Dashboard, Metadata Directory, Semantic Layer, ForgetIT module, Feedback and Conflicts Module, Recycle and Inducing Module, CMIS).
Main APIs, input and output formats	CMIS, REST, OWL
Plan for integration at M18	Application connected to PoF Middleware; initial workflows are served; development according to evaluation plan (WP10).
Plan for integration at M27/M36	Release of the pilot application with the ForgetIT TYPO3 Extensions and the integrated components (CMIS Client/Server and Semantic Services).
Language, runtime framework	TYPO3 CMS 6.2 LTS
SW and HW Requirements	Requirements for SW and HW available in the project documentation (`http://typo3.org/about/typo3-the-cms/system-requirements`)
Dataset for testing	(1) Approved Spielwarenmesse Press Release (2) Testbed including multi domains in TYPO3 (3) Any other press releases and content consisting of text and media assets created in testbed.
License	GPL
Notes	Source code available on project repository [22]

Table 2: TYPO3 CMS (Active System, WP10)

Component Name	**Metadata Repository**
Partner Responsible	EURIX
Contributing Partners	LUH, USFD, CERTH, LTU
Work Packages	**WP8**, WP3, WP4, WP6, WP5
Reference Deliverables	D8.3, D8.4
Current Status	Started component design.
Short description and role	Component that manages metadata extracted or computed for individual documents and collections and makes them available for other components. This, for example, includes extracted entities, context information or memory buoyancy and preservation values. The Metadata Repository relies on the fact that all resources can be identified by an unique ID, which enables the retrieval of metadata stored in the repository for a resource. The repository might also include pointers to summaries for individual documents and/or document collections.
Delivery Mode	REST service
Subcomponents	Metadata storage, component for access/search support.
Main APIs, input and output formats	Main methods include storage of new metadata for a resource, deletion of metadata for a resource, access to specific types of metadata given a resource and search in the Metadata Repository.
Plan for integration at M18	Basic implementation according to the implemented scenarios, to support integration in the PoF Middleware.
Plan for integration at M27/M36	Improvements and extensions of the repository.
Language, runtime framework	Not decided yet, maybe re-use existing open source component and integrate it into the PoF Middleware.
SW and HW Requirements	Requires database for managing the metadata (indexing and search).
Dataset for testing	Some simple test cases should be sufficient for evaluating prototype.
License	Open source, if not otherwise implied by using an existing component.

Table 3: Metadata Repository (PoF Middleware, Shared Component, WP8)

Component Name	**ID Manager**
Partner Responsible	LUH
Contributing Partners	EURIX, dkd, DFKI
Work Packages	**WP8**, WP5, WP9 (ID formats), WP10 (ID formats)
Reference Deliverables	D8.3, D5.2
Current Status	Started implementation of first prototype.
Short description and role	This component mediates between the IDs used in the Preservation System (both in the Digital Repository and in the Cloud Storage Service) and the IDs used in the Active Systems. It might also be used to acquire new unique IDs.
Delivery Mode	REST service or standalone software library.
Subcomponents	ID Repository, ID Generator
Main APIs, input and output formats	Main methods include generation of new ID and retrieval of IDs from a repository. Different standards can be used, such as UUID.
Plan for integration at M18	Basic implementation according to the implemented scenarios, to support integration for the priority workflows.
Plan for integration at M27/M36	Maybe improvements and extensions if necessary, to support additional requirements.
Language, runtime framework	Not decided yet, maybe re-use existing open source component to be integrated in the PoF Middleware. Possible candidate is ObjectDB [23], native object database written in Java with embedded ID generator.
SW and HW Requirements	Requires database for managing the IDs.
Dataset for testing	Some simple test cases should be sufficient
License	Open source, if not otherwise implied by using an existing component

Table 4: ID Manager (PoF Middleware, Shared Component, WP8)

Component Name	**Scheduler**
Partner Responsible	EURIX
Contributing Partners	LUH, LTU
Work Packages	**WP8**, WP3 (scheduling of forgetting process), WP5 (scheduling of archiving process)
Reference Deliverables	D8.3, D8.4
Current Status	Started development of first prototype.
Short description and role	Component that starts processes such as the forgetting process or an archiving process based on a defined schedule (e.g. once a day) or based on events, for which it is listening, plus additional conditions.
Delivery Mode	Active component, triggers other components and processes. Depending on PoF Middleware implementation, the Scheduler should be a running process, listening for specific triggering events.
Subcomponents	Scheduling queue, event management, business logic for event processing.
Main APIs, input and output formats	API that allows the scheduling of processes based on time and events, API for requesting status information, API for deletion of scheduled events.
Plan for integration at M18	Basic implementation for processing of simple events or for basic time-based scheduling of processes and tasks.
Plan for integration at M27/M36	Improvements and extensions, integration with PoF Middleware communication layer (ESB) supporting complex workflows.
Language, runtime framework	Not decided yet, maybe re-use existing open source component to be integrated in the PoF Middleware.
SW and HW Requirements	Not yet specified, depending on the actual implementation.
Dataset for testing	Some scheduling test cases based on priority workflows.
License	Open source, if not otherwise implied by using an existing component.

Table 5: Scheduler (PoF Middleware, Shared Component, WP8)

Component Name	**Context-Aware Preservation Manager**
Partner Responsible	LTU
Contributing Partners	EURIX, dkd, DFKI, TT, IBM
Work Packages	**WP5**, WP8, WP6
Reference Deliverables	D5.3
Current Status	Started design of the component.
Short description and role	The function of the Preservation Planning entity, and to some extent the Administration entity, in the Preservation System) need to be *stretched out* to meet the Active Systems (and their owners). Some of it is available through other components in the PoF Middleware, but there still exists a need to handle changes on both sides of the PoF Middleware, which includes enabling communication of events and triggers relevant for both the Preservation System and the (owners of the) Active Systems. As an example, the Preservation System has internal preservation plans which might include transformation of classes of objects at ingest, if the objects are in unsuitable formats. These *default transformations* need to be communicated to the Active Systems. This may be communicated already at (or before) initial ingest of the first object of a specific type, since these plans are known beforehand. As another example, the Preservation System is responsible for preservation of the objects for long term, but the PoF Framework must be able to re-contextualize the objects into Active Systems. This means that when the Preservation System makes a decision to transform a class of objects, this must be communicated to the Active System and its owners. If this transformation would ruin the chances of re-contextualization, e.g. by deleting the original object, some actions need to be taken to ensure the possibility of re-contextualization (e.g. transformation to another format for re-contextualization).
Delivery Mode	REST service
Subcomponents	Event logger
Main APIs, input and output formats	REST APIs under definition, supporting JSON and XML formats.
Plan for integration at M18	Transmission of simple events between Active System and Preservation System, not targeted for first PoF Framework prototype.
Plan for integration at M27/M36	Development and integration of first prototype, communicate changes in Preservation System and Active Systems, regarding e.g. information structure.
Language, runtime framework	Java/J2EE, Java Application Server/Tomcat
SW and HW Requirements	Linux OS, other requirements not available yet.
Dataset for testing	Create ad-hoc dataset, e.g. descriptions of transformation events using JSON or XML, to be processed by this component.
License	Open source license

Table 6: Context-Aware Preservation Manager (PoF Middleware, Shared Component, WP5)

Component Name	**Forgettor**
Partner Responsible	LUH
Contributing Partners	DFKI
Work Packages	**WP3** , WP9 (requirements and APIs), WP10 (requirements and APIs)
Reference Deliverables	D3.2, D3.3, D3.4
Current Status	Preliminary prototype under development.
Short description and role	This component is managing the forgetting process. It computes the Preservation Value (PV) and Memory Buoyancy (MB) for resources based on information provided by the Active System for this computation such as usage information, context information and creator as well as based on the previous MB and PV values, statistics and defined strategies and rules. The results of the Forgettor component will be made accessible for the Active System via the Metadata Repository. It is planned that the Forgettor is activated on a regular basis. For this purpose it interacts with the Scheduler.
Delivery Mode	REST service
Subcomponents	(1) *Assessor*: the Assessor calculates values for current information value assessment of a resource, especially MB and PV. It takes into account different forgetting strategies as well as the previous values and statistics from the last computation. (2) *Strategy Manager* (database): different forgetting strategies and rules are managed here. (3) *Statistics/Value Repository* (database): storing all the values computed for information value assessment (including MB and PV) and statistics and using it for computing new values. (4) *Analyzer*: used for classifying resources based on a strategy and the values computed in Assessor and statistics.
Main APIs, input and output formats	Input: resource IDs and metadata associated with the resource (mainly information on resource usage and resource context). The metadata should be a flexible data-structure, e.g. key-value pairs, so that it can be extended for different cases. Output: computed MB and PV values together with classification in more high level classes (e.g. ”to be preserved” or ”low importance”); these will be used to update the respective information in the Metadata Repository to make the most current values available for the Active System. It is still under discussion if the Forgettor also informs the Scheduler or the Active System about the completion of the computation (e.g. via event listeners).
Plan for integration at M18	A first simple version of the Forgettor with a simple function for computing MB.
Plan for integration at M27/M36	Intermediate and final release of the overall Forgettor.
Language, runtime framework	Java
SW and HW Requirements	Database for storing historical values.
Dataset for testing	Usage logs and other usage information from the user applications.
License	Open source

Table 7: Forgettor (PoF Middleware, Core Component, WP3)

Component Name	**Extractor**
Partner Responsible	CERTH
Contributing Partners	USFD, TT
Work Packages	**WP4**, WP8
Deliverables	D4.2, D4.3, D4.4
Current Status	Preliminary prototype under development.
Short description and role	The Extractor takes as input the original media items (e.g. a text, a collection of texts, or a collection of images) and extracts information that is potentially useful not only for the subsequent execution of the Condensator, but also for other components or functionalities of the overall PoF Framework (e.g. for search). This extracted information constitutes the Extractor's output, and will be provided in simple text or XML files (to be decided, depending on what is most convenient for integration, there is some flexibility). We envisage the Extractor to include subcomponents that will perform the following tasks:(1) named entity extraction from text, (2) tokenization, (3) visual feature extraction from images, (4) concept detection in images, (5) image visual quality assessment. The Extractor could be either a command line tool or a REST service (both options seem feasible).
Delivery Mode	Command line tool or REST service.
Subcomponents	(1) Named entity extraction from text, (2) Tokenization, (3) Visual feature extraction from images, (4) Concept detection in images, (5) Image visual quality assessment.
Main APIs, input and output formats	Input: one text file or a collection of text files or a collection of images. Output: plain text or XML files with analysis results.
Plan for integration at M18	A first release of the overall Extractor, integrating most of its subcomponents, as part of D4.2 [9].
Plan for integration at M27/M36	Intermediate and final release of the overall Extractor.
Language, runtime framework	Matlab/Octave, C++ executables/binaries which require OpenCV libraries (.dll/.so).
SW and HW Requirements	Operating System: Windows or Linux OS required, Matlab/Octave, NVIDIA GPU desirable.
Dataset for testing	(1) Travel of two colleagues to Edinburgh 2013, (2) travel pictures from CostaRica 2013 dataset, (3) a dataset of about one thousand images has been assembled and experiments are being run for blur detection, (4) other external datasets (e.g. TRECVID).
License	BSD license (compliant with OpenCV), license for GPU_SURF and SURF implementation (an application of SURF algorithm is patented in the US), license for LIBLINEAR Project software.

Table 8: Extractor (PoF Middleware, Core Component, WP4)

Component Name	**Condensator**
Partner Responsible	CERTH
Contributing Partners	USFD, TT
Work Packages	**WP4**, WP8
Reference Deliverables	D4.2, D4.3, D4.4
Current Status	First prototype under development.
Short description and role	The Condensator will get as input the Extractor's output and possibly also the original media items that were processed in order to generate this output (or a subset of these media items). Based on this input, the Condensator will perform further text and image analysis tasks whose results are specific to the condensation process and thus of no need to other parts of the PoF Framework, and will use all the available analysis results for performing text and image collection condensation. No feedback loop from the Condensator back to the Extractor is foreseen (thus, the Condensator can only be called after the Extractor has been executed for the same data, and the Condensator's results will not be fed in any way back to the Extractor). The final output of the Condensator will be the condensed (i.e., summarized) media items or collections, or pointers to them (depending on media item modality and on what is more convenient for integration, to be decided at a later stage). Any other analysis results generated within the Condensator for the purpose of supporting the generation of the condensed media collections most probably will not be returned to the framework (since, by definition, these are only intermediate results useful for condensation; otherwise, their extraction would be part of the Extractor). We envisage the Condensator to include subcomponents that will perform the following tasks: (1) *deeper* linguistic analysis, (2) text summarization, (3) face detection and clustering, (4) image collection summarization. The Condensator could be either a command line tool or a REST service (both options seem feasible).
Delivery Mode	Command line tool or REST service.
Subcomponents	(1) *deeper* linguistic analysis, (2) text summarization, (3) face detection and clustering, (4) image collection summarization
Main APIs, input and output formats	Input: the output of the Extractor, which is plain text or XML files with analysis results, and the original text and image items, Output: text files, image files and plain text or XML files with analysis results
Plan for integration at M18	A first release of the overall Condensator, integrating most of its subcomponents, as part of deliverable D4.2 [9].
Plan for integration at M27/M36	Intermediate and final release of the overall Condensator.
Language, runtime framework	Matlab/Octave, C++ executables/binaries which require OpenCV libraries (.dll/.so).
SW and HW Requirements	Operating System: Windows or Linux, Matlab/Octave
Dataset for testing	(1) travel of two colleagues to Edinburgh 2013, (2) travel pictures from CostaRica 2013 dataset, (3) other external datasets (e.g. TRECVID)
License	BSD license (compliant with OpenCV)

Table 9: Condensator (PoF Middleware, Core Component, WP4)

Component Name	**Contextualizer**
Partner Responsible	USFD
Contributing Partners	LUH, CERTH
Work Packages	**WP6**, WP8
Deliverables	D6.1, D6.2, D8.3
Current Status	First prototype under development.
Short description and role	The Contextualizer will embrace different subfunctionalities including as its core functionality the equipment of information objects with sufficient context information for their long-term interpretation and use, taking as input the original media items (e.g. images, text, etc.) as well as the output of the Extractor and Condensator. Furthermore, it might also use external data sources for enriching the context information (e.g. Wikipedia, or other pictures for the same event). This information will be used to determine the context required to unambiguously describe the input media and is likely to be defined with reference to an ontology; either large public ontologies, such as DBpedia, Freebase etc., or private ontologies from the PIMO etc. Storing a complete copy of an ontology with each preservation package is likely to be highly inefficient, and so this component will also be responsible for determining the minimum context that can be stored without loss of information. It will also interact with the Collector/Archiver for preparing the context information for packaging. Contextualization will be triggered by the intend to archive an individual information object or a set of information objects. The exact nature of the Contextualizer is likely to differ dependent upon the media type. The exact formatting of the context has yet to be formalized although most tools will output XML encoded data. Furthermore, it will also be responsible for reflecting evolution in the Active System (and the world) into the stored context information, in order to keep the information objects as well as the context information useful and understandable on the long run. This requires mechanisms to get informed about major changes in the Active Systems (e.g. in the ontology). Furthermore, it has to be identified which stored contexts are effected by these changes and change has to be represented and propagated into the Preservation System. For bringing information objects back into active use, a mechanism is required to apply the encoded change to the context information and/or to use them for integrating the information object into the current context (re-contextualization). This might also require using external resources or organizational information, if the captured context information is not sufficient. A further type of change affecting understandability is terminology evolution, which has to be detected and reflected in the context, in order to keep things findable.
Delivery Mode	REST service, initially a command line tool.
Subcomponents	Different components for text versus images etc., components for contextualization and re-contextualization.
Main APIs, input and output formats	REST service, definition of interfaces and response formats is in progress.
Plan for integration at M18	Basic component (separate components for different media types).
Plan for integration at M27/M36	A more integrated component with advanced functionality.
Language, runtime framework	Java, C++
SW and HW Requirements	Linux OS required.
Dataset for testing	Not decided yet, probably user experiments.
License	Released by USFD under LGPL.

Table 10: Contextualizer (PoF Middleware, Core Component, WP6)

Component Name	**Navigator**
Partner Responsible	USFD
Contributing Partners	EURIX, LUH
Work Packages	**WP6**, WP8
Reference Deliverables	D6.2, D8.4
Current Status	Started component design
Short description and role	It is likely that the Navigator component will be highly integrated into the use case tools and as yet the form it will take has not been fully defined. The component will, however, be responsible for allowing users to see preserved items in their context (both the context at the time of preservation and at retrieval) and allow the navigation of the archived content in the Preservation System via links to other items via shared context links (i.e. a photo collection of a trip to Edinburgh in 2013 would share a context with a diary about a trip to Edinburgh in 2016). It is envisaged that an initial version of this component will focus on providing a search interface across both active and archived information, with later versions incorporating more ideas around context navigation. Such a component would require access to both active and preserved content and may need to be deployed within use case tools and the Preservation System (as a Storlet) as well as within the PoF Middleware to make navigation feasible within sensible time constraints. This requires support for a shared index or a method for combining the results from two indexes into a meaningful way. For the ranking, a time-aware search support is required, which favors information objects in active use over information objects from the Preservation System.
Delivery Mode	REST service
Subcomponents	Index management, time-aware search support, context navigation support.
Main APIs, input and output formats	REST service, definition of interfaces and response formats is in progress.
Plan for integration at M18	A basic component for early testing.
Plan for integration at M27/M36	Advanced features depending on implemented scenarios.
Language, runtime framework	Not decided yet, depends on actual implementation and deployment.
SW and HW Requirements	Linux OS required.
Dataset for testing	Not specified yet, probably user experiments.
License	Released by USFD under LGPL.

Table 11: Navigator (PoF Middleware, Core Component, WP6)

Component Name	**Collector/Archiver**
Partner Responsible	LTU
Contributing Partners	dkd, DFKI, EURIX
Work Packages	**WP5**, WP8 (SIP format), WP9 (CMIS client), WP10 (CMIS client)
Reference Deliverables	D5.2, D5.3, D5.4
Current Status	Preliminary prototype under development.
Short description and role	Triggered by an event (either generated in the PoF Middleware or after a request from user applications), this component contacts the Active Systems and collects data objects that should be preserved, then prepares and submits them to the Preservation System by packaging the data objects together with relevant metadata (into a SIP). The component receives a preservation request. Acting on that request, the Collector fetches the object and metadata from provided reference, via the CMIS interface of the Active System. The Collector notifies the ID Manager with the CMIS ID (GUID) of the object, and notifies the PoF Middleware bus that an object has been collected. Before a SIP can be created, other PoF Middleware components, such as the Extractor and the Condensator, need to process the object and extract relevant metadata and other characteristics needed for the forgetting process. This metadata is stored in the Metadata Repository, and on a trigger from the PoF Middleware bus, the Archiver fetches metadata and prepares the package and submits it to the Preservation System. There are at least two options here: (1) the Archiver sends a reference to where the Preservation System can fetch the package; (2) the Archiver sends the package to the ingest folder of the Preservation System. In response to the submission, the Archiver needs an ”archive ID” that should be sent to the ID Manager. The Collector/Archiver is also responsible for restructuring DIP into a package that the Active System can handle to get the information back into active use. As a response to a trigger, that can come from the Active System, or from PoF Middleware internal components (e.g. the Scheduler), a request is made to the Preservation System for a DIP. The DIP is then disassembled and restructured (if needed) for adoption in the Active System. This may include restructuring of metadata in order to facilitate ingest into the Active System. Transformation of content objects is not considered to be a part of this functional entity. Communication will chiefly be with Active Systems, and with the Preservation system.
Delivery Mode	Command line tool or REST service.
Subcomponents	Packager, Metadata extractor, Hash generator/checker, CMIS client.
Main APIs, input and output formats	REST APIs supporting different input/output formats and protocols: AtomPub, CMIS, custom ForgetIT XML schema, TAR package (SIP/DIP).
Plan for integration at M18	Ingest workflow in place with basic functionality; simple re-contextualization of DIP into Active System.
Plan for integration at M27/M36	Continued development of ingest workflow, and in particular re-contextualization. Final component available at M36.
Language, runtime framework	Java/J2EE, Java Application Server/Tomcat.
SW and HW Requirements	Linux OS required.
Dataset for testing	Example data from user applications, retrieved with CMIS protocol and used to prepare SIP.
License	Not yet available, possibly open source.
Notes	Dependencies: Condensator, Extractor, Contextualizer (i.e. Metadata Repository) for metadata. Communicates with PoF Framework adapters in the Active Systems and in the Preservation System.

Table 12: Collector/Archiver (PoF Middleware, Core Component, WP5)

Component Name	**Digital Repository**
Partner Responsible	EURIX
Contributing Partners	
Work Packages	**WP8**
Reference Deliverables	D8.3, D8.4, D8.6
Current Status	Stable solution available, to be customized for minor changes.
Short description and role	The Digital Repository is responsible for organizing digital content received from producers and for delivering such content to consumers in convenient ways. The Digital Repository has to manage the content produced by the Active Systems, providing also archival functionalities, exposing interfaces to import and retrieve ForgetIT content during synergetic preservation processes. The Digital Repository integrates with the Cloud Storage Service, making up the Preservation System. Since it acts also as an archive, it should be compliant to the OAIS functional model, implementing the main functional entities (*Ingest*, *Access*, *Preservation Planning*, *Data Management*, *Administration* and *Archival Storage*). The Digital Repository should support the packaging model selected in the project, exposing REST APIs for ingest and access, with internal workflow engine. Concerning access, basic search (using AIP identifiers or simple descriptive metadata) is required. Advanced search is provided by other PoF Middleware components. The customization of the archive will include the development of an adapter to integrate the PDS.
Delivery Mode	Platform running in application server on dedicated machine.
Subcomponents	Ingest, Access, Data Management, Preservation Management (Preservation Planning and Administration), Archival Storage (interface to Cloud Storage Service). OAIS functional entities will be extended to support ForgetIT principles (see deliverable D8.2 [14]): for example Ingest and Access also associated to de-contextualization and re-contextualization; deletion of archived content either permanently or by simply removing content from repository index could be required for managed forgetting. For Preservation Planning, processes are executed close to the data (e.g. fixity checks or format transformation), making use of Storlets. For Archival Storage, a module for storing AIPs, integrating with PDS, must be implemented. AIP versioning useful for specific ForgetIT processes. Integration using REST API, Web user interface is useful.
Main APIs, input and output formats	REST APIs for ingest and access are exposed, AIP packaging model should make use of METS [19] for metadata description, packaging format based on BagIt or derivatives. Supported protocols include OAI-PMH, an additional CMIS or JCR compliant interface must be implemented if required.
Plan for integration at M18	Fully implemented and integrated, adopt existing solution for early integration.
Plan for integration at M27/M36	Refinements due to possible changes in the components or to demonstrate additional scenarios.
Language, runtime framework	Java, running in application server.
SW and HW Requirements	Linux server, enough disk space for test resources and services.
Dataset for testing	Picture dataset as first test, then all content types from user applications.
License	Open source, GPL or BSD compliant.
Notes	Evaluation of different candidate solutions reported in Section 7.1.

Table 13: Digital Repository (Preservation System, WP8)

Component Name	**Cloud Storage Service**
Partner Responsible	IBM
Contributing Partners	
Work Packages	**WP7**
Reference Deliverables	D7.2, D7.3, D7.4
Current Status	Preservation DataStores and Storlet Engine is under development.
Short description and role	Preservation DataStores (PDS) is an OAIS-based preservation-aware storage that serves as an advanced Archival Storage and supports offloaded functionality. At the top, it provides an OAIS-based interface for operations on AIPs (e.g., ingest, access, delete), as well as an interface for preservation actions (e.g., check fixity, transform, add aggregation). At the bottom end, it utilizes various generic cloud storage and compute from different providers. In addition, the system includes a Storlet Engine that can be plugged into a private cloud or object storage to execute computation modules (called Storlets), close to the data (transformations or other resource consuming tasks executed on the archived content can be done directly in the Preservation System without requiring to extract it to a server and put it back into the Preservation System). The PDS interface specification can be found on the ENSURE project web site [24, 25]. The underlying cloud storage that will be used in PDS for ForgetIT is the OpenStack Swift, the Object Storage component of the open source OpenStack [21] framework, enhanced with a Storlet Engine to perform computations close to the data. Building the Storlet Engine and Storlets for ForgetIT data and use cases is the main focus of WP7. Examples of potential Storlets for ForgetIT: (1) Summarization and aggregation processes to enable managed forgetting in the archive, (2) Redundancy detection and deletion processes to support managed forgetting, (3) Multimedia analysis algorithms, (4) Integrity checks to make sure the data is not altered over time, (5) Format transformations.
Delivery Mode	REST service deployed in a virtual machine
Subcomponents	PDS, OpenStack Swift, Storlet Engine
Main APIs, input and output formats	PDS interfaces are documented in WP7 deliverables.
Plan for integration at M18	A first release of PDS with Storlet Engine and some Storlets for specific ForgetIT tasks.
Plan for integration at M27/M36	Intermediate and final release of PDS with Storlet Engine.
Language, runtime framework	Java, Python
SW and HW Requirements	Linux with OpenStack software and IBM extensions
Dataset for testing	Depends on ForgetIT use cases and Storlets, all content types provided by Active Systems and ingested in the Preservation System are supported, transformation with Storlets is applied to specific content types according to preservation rules.
License	Proprietary

Table 14: Cloud Storage Service (Preservation System, WP7)

4 Architecture Diagrams and Integrated Workflows

In this Section we present a first round of more in depth modeling of the PoF architecture and the related processes. For this purpose, we adopt Unified Modeling Language (UML) notation [26], creating static and dynamic diagrams of the architecture. We are currently using only a subset of the available diagrams, namely component diagrams among UML structure diagrams and activity diagrams among UML behavior diagrams. Further, more fine-granular diagrams will be exploited in the next round of modeling, for example class and object diagrams will be added to deliverable D8.3 [13].

In addition to deployment and component diagrams for the PoF Framework architecture, we also describe two initial priority workflows which will be used to drive and validate early integration of the main components in the PoF Middleware as well as their interaction with the Active Systems and the Preservation System. For those integrated workflows we provide UML activity diagrams.

4.1 Structure Diagrams

In Figure 3 a UML deployment diagram is shown, based on a possible configuration of systems and nodes which will be implemented for development and testing (see Section 6 for a description of the testbed environment).

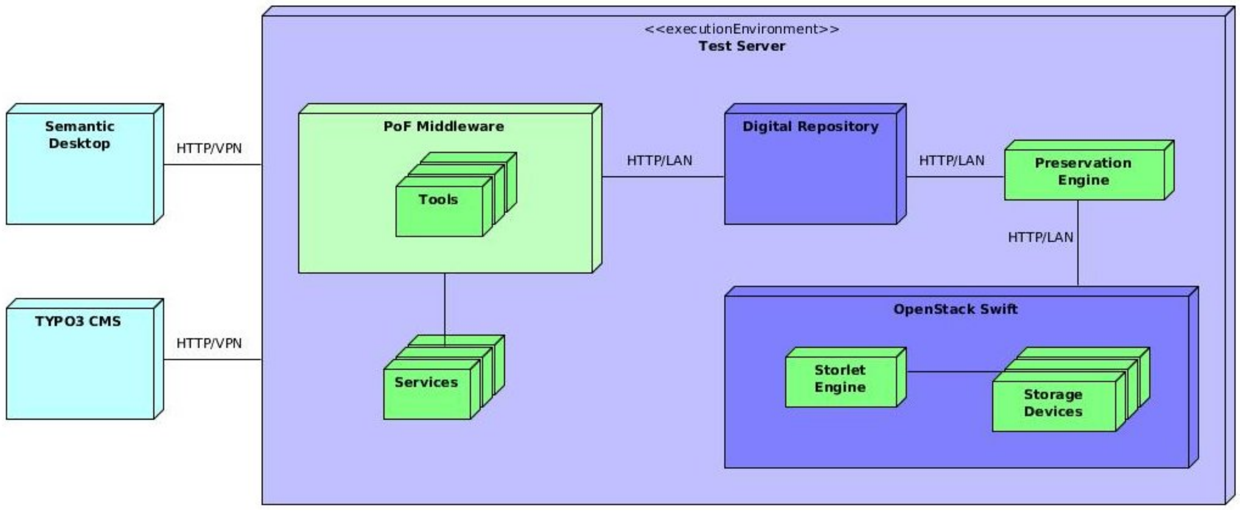

Figure 3: Deployment diagram of the PoF architecture.

The component diagram of the architecture is depicted in Figure 4. It is based on the architecture overview presented in Figure 1 and shows main interfaces and protocols connecting system components. The functionality of the individual components has already been described in Section 3.

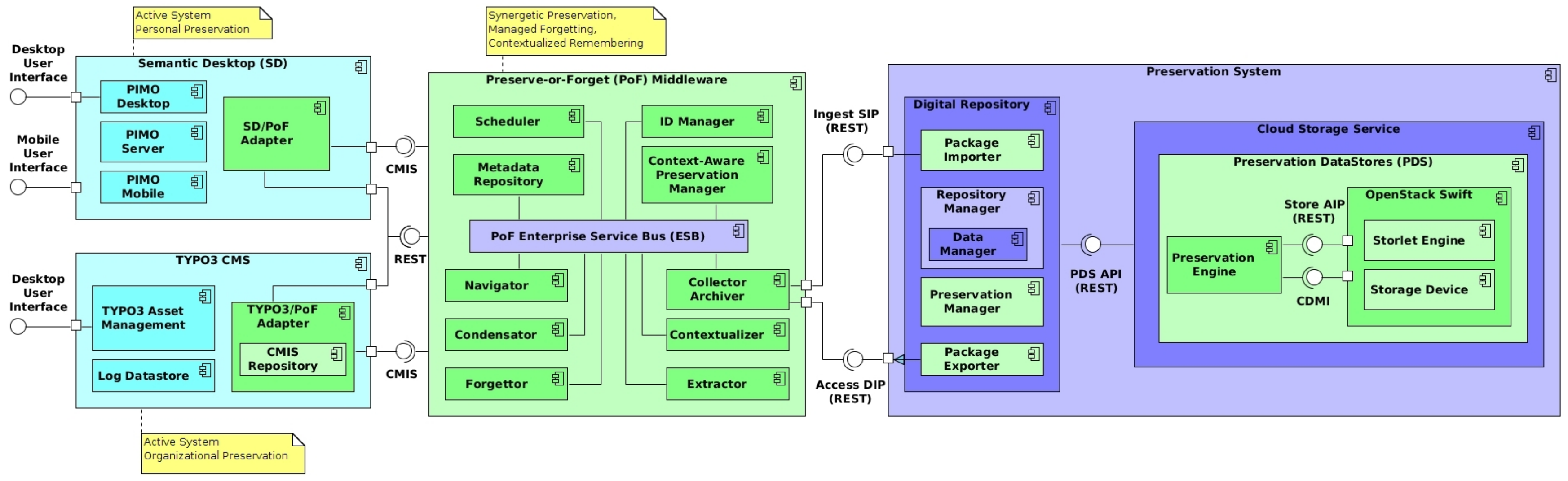

Figure 4: Component diagram of PoF architecture (*green*: components developed during the project; *blue*: existing components to be improved and customized; *turquoise*: existing active systems).

4.2 Integrated Workflows

Within ForgetIT it has been decided to identify a smaller set of workflows including core ForgetIT functionalities. These workflows will be used to define priorities for the integration activities in the first phase of the ForgetIT project (see also Section 5). Figures 5 and 6 show two workflows that have been defined for this purpose, identified as *Basic Synergetic Preservation* and *Basic Managed Forgetting Support*with the following notations: (1) complex activities are written in bold (a sub-diagram will be defined in the future), while non-bold is used for actions related to less complex actions, (2) red arrows distinguish ”object flow” (red) from ”control flow” (standard case, black).

The first workflow (see Figure 5) focuses on a basic form of Synergetic Preservation, which enables the smooth transition of a resource from the Active Systems into the Preservation System passing through the process of contextualization and packaging. The second workflow (see Figure 6) focuses on core functionality of information value assessment as it is required for realizing the Managed Forgetting process. In implementing these first workflows the focus will be on ensuring proper interaction between the individual components in the architecture. More advanced functionality for the individual components will be added stepwise to the individual components in later phases of the ForgetIT project. Both processes are described in more detail in the following Sections.

4.2.1 Workflow 1: Basic Synergetic Preservation

The workflow depicted in Figure 5 defines the process for a basic form of Synergetic Preservation and shows the involvement of the ForgetIT components into this process.

Preservation is initiated by a preservation process, scheduled either to start on a regular basis or to be triggered by an event. The *Scheduler*, connected to the *PoF Middleware* bus (see Figure 4), starts the archiving process for a selected set of resources. This process inspects the Preservation Value (PV) and the preservation status that is stored for the resources in the *Metadata Repository* and, based on this information, decides which of the resources are handed over to preservation. We focus on cases where the PV is explicitly manipulated in the *Active Systems* and a change of the PV is triggered by a preservation request in the user applications. The update of the PV is written into the *Metadata Repository*. As a next step after deciding about preservation, a simple contextualization activity is performed. This step adds core context information to the resource(s) to be archived. For this purpose, the *Contextualizer* is used, which interacts with the *Metadata Repository*, the *Extractor* and possibly also with the *Active Systems* and external sources to collect the required context information. The core challenge here is to derive concise context information, which enables future interpretation, while not overloading the *Preservation System* with unnecessary information. Both the resource(s) to be archived and the context information are handed over to the *Collector/Archiver* component for packaging them into a SIP package. After this step the resource is ready to be handed over to the *Digital Repository*, which is part of the *Preservation System*.

At this point the archival of the packages in the *Preservation System* is initialized. This includes checking the packages for fitness to be archived. As a result of this fitness test a transformation might become necessary. In the ForgetIT system this transformation will be performed by a *Storlet*, foreseen for this purpose in the *Cloud Storage Service*. Therefore, the transformation will be scheduled and will be performed once the packages have been transferred into the *Cloud Storage Service*, which is the next step in the workflow. If the archival is successful the archive ID(s) assigned to the resource(s) are returned to the *PoF Middleware* and stored in the *ID Manager* together with the time of archival, in order to enable translation of an ID of the resource in the *Active System* to the ID of the same resource in the *Preservation System*. Furthermore, the *Scheduler* will be informed about the completion of the archival action. In case of the failure of the archiving process, the *Scheduler* is notified accordingly and decides about the notification of the *Active System*.

4.2.2 Workflow 2: Basic Managed Forgetting Support

The workflow shown in Figure 6 is made up of two separate processes for information value assessment, which both serve the purpose of helping the user to better structure her information space according to the value or importance of information, the Memory Buoyancy (MB). Such information value assessment is the starting point for the Managed Forgetting process.

The first process starts from the observation of usage of resources in the *Active System*. Such information about the usage is collected by the *Active System* and transferred into a *message cache* on the side of the *PoF Middleware*. Independent from this activity, a forgetting action can be scheduled in the *PoF Middleware*. This is done by the *Scheduler*, e.g., on a regular basis. Once the scheduled forgetting action is started, the *Forgettor* component uses the relevant usage information from the cache (see above) and possibly also further context information to compute new values for MB and/or PV for the considered resources. In addition, it also uses previous MB and PV values as well as strategies for this computation. The new MB and PV values are stored in the *Metadata Repository*, connected to the *PoF Middleware* bus. The new values in the *Metadata Repository* can subsequently be accessed by the *Active System*, for example, in order to adapt the visualization of resources in the *Active System* according to their MB value.

The second process focuses on a quality-based contribution to MB, rather than a usage-based one as for the first process. This second process is started by the *Active System* by triggering a quality assessment process for a newly captured set of images (e.g., in the *Semantic Desktop*) via the *Scheduler*. The *Scheduler* activates a service for performing quality checking. This quality-assessment service, which is part of the *Extractor*, gets access to the collection of images under consideration, e.g., by uploading them. Subsequently the automated quality assessment of the images is performed and the results of this assessment are stored into the *Metadata Repository*. The quality assessment results in the *Metadata Repository* can be accessed by the *Active System* and can be used to trigger further activities such as suggesting some of the images for deletion.

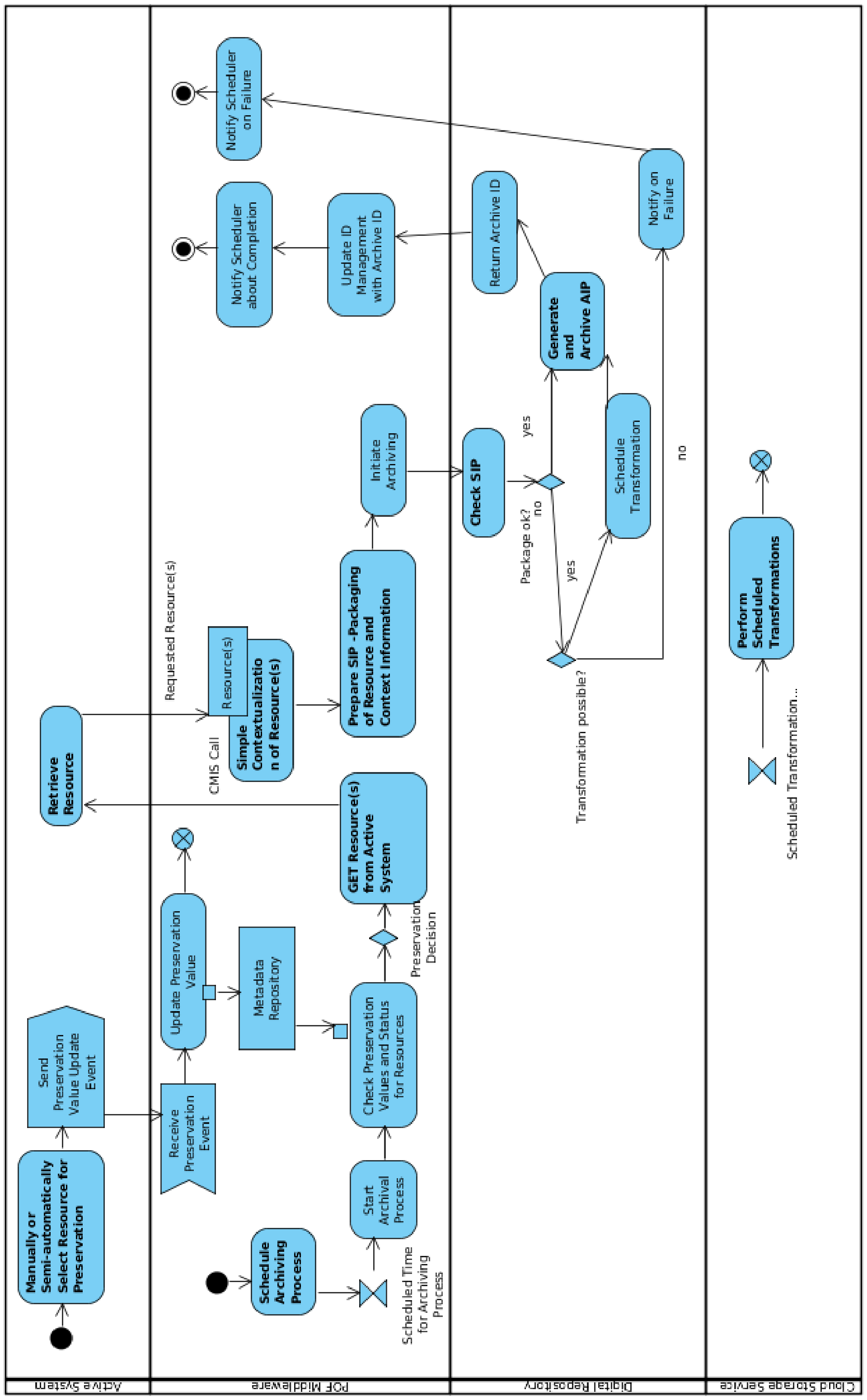
Active System
POF Middleware
Digital Repository
Cloud Storage Service
Manually or Semi-automatically Select Resource for Preservation
Send Preservation Value Update Event
Retrieve Resource
Schedule Archiving Process
Receive Preservation Event
Update Preservation Value
Metadata Repository
Scheduled Time for Archiving Process
Start Archival Process
Check Preservation Values and Status for Resources
Preservation Decision
GET Resource(s) from Active System
CMIS Call
Requested Resource(s)
Resource(s)
Simple Contextualization of Resource(s)
Prepare SIP -Packaging of Resource and Context Information
Initiate Archiving
Check SIP
Package ok?
yes
no
Transformation possible?
yes
no
Schedule Transformation
Generate and Archive AIP
Return Archive ID
Notify on Failure
Update ID Management with Archive ID
Notify Scheduler about Completion
Notify Scheduler on Failure
Scheduled Transformation...
Perform Scheduled Transformations

Figure 5: Priority Workflow for Basic Synergetic Preservation (complex actions in bold).

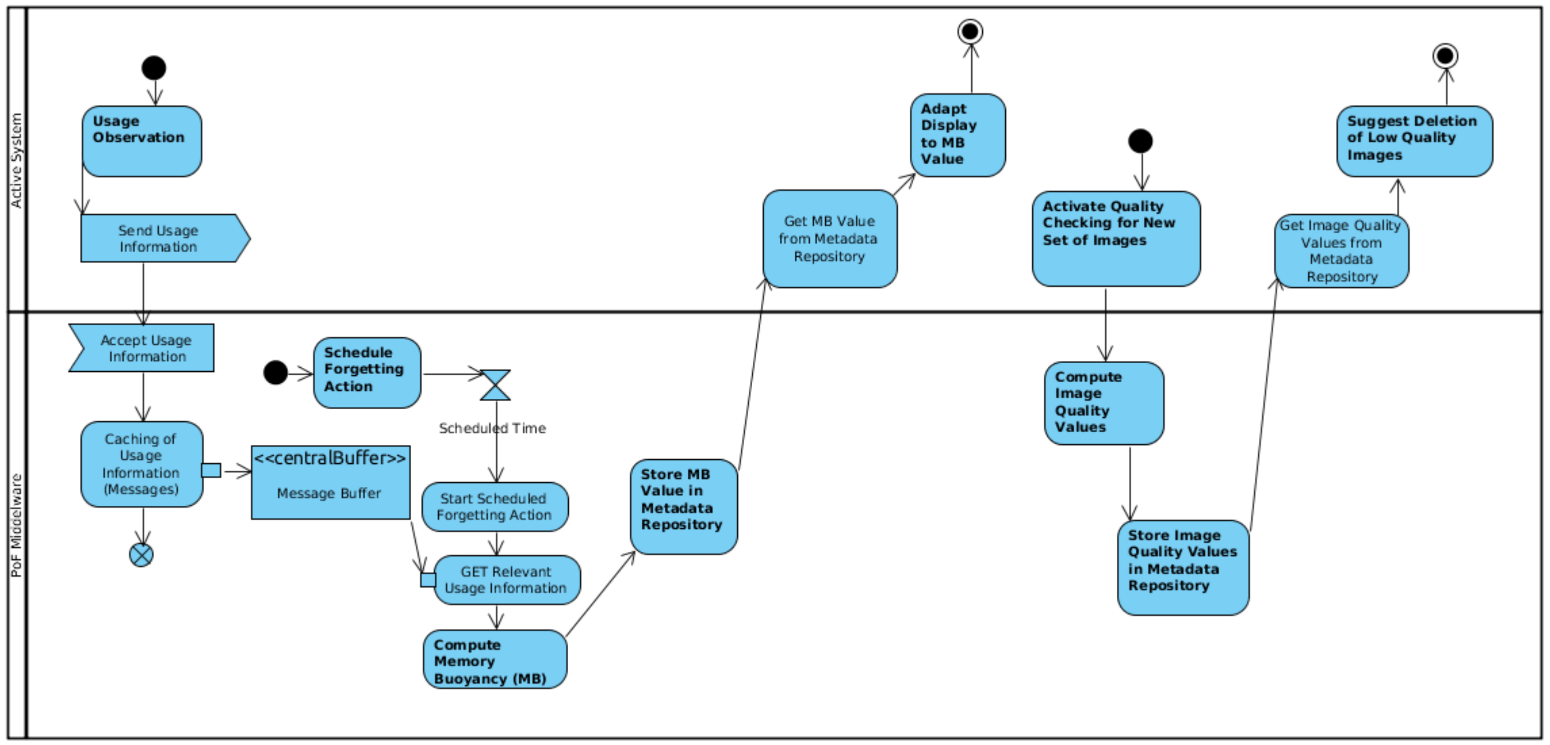
Active System
PoF Middelware
Usage Observation
Send Usage Information
Accept Usage Information
Caching of Usage Information (Messages)
<<centralBuffer>>
Message Buffer
Schedule Forgetting Action
Scheduled Time
Start Scheduled Forgetting Action
GET Relevant Usage Information
Compute Memory Buoyancy (MB)
Store MB Value in Metadata Repository
Get MB Value from Metadata Repository
Adapt Display to MB Value
Activate Quality Checking for New Set of Images
Compute Image Quality Values
Store Image Quality Values in Metadata Repository
Get Image Quality Values from Metadata Repository
Suggest Deletion of Low Quality Images

Figure 6: Priority Workflow for Information Value Assessment and Managed Forgetting (complex actions in bold; red arrows: object flows; black arrows: control flows).

5 Integration Plan

According to ForgetIT project proposal, three releases are expected for the ForgetIT PoF Framework, described in deliverables D8.3 (first release), D8.4 (second release) and D8.6 (final release), respectively. In parallel two releases of the PoF Reference Model will be available, reported in deliverables D8.2 (initial model) and D8.5 (final model), respectively.

The integration plan for the components of the PoF Framework is summarized in Table 15. For each component described in the previous Sections, we report the planned integration status for each expected PoF Framework releases.

For each component we distinguish three generic integration levels: *None*, *Partial* and *Complete*, where *None* refers to no or very limited integration, *Partial* is used when only some component functionalities have been integrated or when the integration still makes use of some temporary solutions and workarounds to be replaced or better engineered, while *Complete* refers to the integration of all available functionalities, developed to a satisfactory level and in compliance with integration best practices (e.g. loose coupling). It is worth noting that the levels above are associated to the degree of integration of a given component, but are also affected by the development status of a given component for a specific PoF Framework release. Some components will be designed and implemented at the very beginning, because they are crucial for the first release implementation or because their implementation is independent of other project results, while other components will require deeper analysis within technical WPs.

For each component the development plan has been discussed within the corresponding Work Package with all involved partners. First the role of the component in the overall architecture has been discussed in detail and shared among all interested partners, then a detailed plan for the integration in the first PoF Framework release has been prepared. The details about each component have been already discussed in Section 3, further information can be found in the deliverables by the corresponding WPs.

5.1 Plan for the first PoF Framework release

For the first release of the PoF Framework we will adopt a two-step approach: we will start with the integration of a limited number of components in order to demonstrate the two priority workflows discussed in Section 4.2, involving the three architecture layers. This early integration will demonstrate that a complete workflow can be executed, starting from active content use, through basic managed forgetting, simple contextualization, archiving and storage; restore of the content back to active use (after possible transformations based on simple rules) will also be shown. With this infrastructure in place, we will be able to test the main interfaces provided by each system and the communication protocols among the different layers. From this preliminary test phase we expect the explicitation of further requirements with respect to the interaction between components. We will also evaluate the packaging model used for archiving and the final deposit on the cloud storage. After this phase is completed, we will integrate more components into the PoF Middleware, adding additional functionalities to the overall framework.

According to Table 15, for many components a prototype will be ready by the time of the first release. They will be improved during the project lifetime, also taking into account evaluation results from the applications (e.g. the pilots scheduled at M23). In particular, for the PoF Middleware implementation and the Digital Repository, candidate implementations will be evaluated early in the project (see Section 7). For each component a partner responsible for development and testing has been identified, although many components will require contributions by several partners and different WPs.

In a nutshell, for the first release the project will deliver a preliminary version of the PoF Framework with the main architecture blocks already integrated and able to communicate. This will be used to demonstrate two initial integrated priority workflows for both the personal and organizational scenario. Finally we will make use of the packaging model and a number of Storlets running in the cloud storage performing content transformation.

5.2 Preliminary plan for other PoF Framework releases

A detailed integration plan for the other releases is not meaningful at this stage, anyway in Table 15 we also report the tentative plan for the second and third framework releases. For the components already integrated in the first release, they will be replaced by improved versions developed in the technical WPs, as discussed above. We can expect that after the first framework release, the evaluation from the users will provide additional feedbacks which will help in improving the overall system. As already mentioned, further development of the components will also provide additional features to be integrated. A detailed plan for the other releases will be discussed internally among all partners after completing the first release and will be updated in the incoming WP8 deliverables.

5.3 Testing components and integration

For each component, both white-box and black-box testing techniques will be used. White-box techniques will be used by partners responsible for each component to test core functionalities in a controlled environment, simulating input from other components or triggering specific events. When a new release of a given component will be ready, it will be integrated in the framework using the testbed environment (see Section 6). The components will be deployed as standalone binary executables or libraries or as web services deployed in virtual machines.

The testbed environment will be used also for black-box testing of the priority workflows, starting from events triggered by user applications, as discussed before. This black-box approach is also important to validate the stability and robustness of the PoF Framework APIs and to evaluate the performance of the overall framework.

Component	Resp.	Sections	1st Rel. (M18)	2nd Rel. (M27)	Final Rel. (M36)
Active Systems					
Semantic Desktop (PIMO)[a]	DFKI	2.1, 3.1	P	C	C
TYPO3 CMS[b]	dkd	2.1, 3.1	P	C	C
PoF Middleware Shared Components					
Metadata Repository	EURIX	2.2, 3.2	N	P	C
ID Manager	LUH	2.2, 3.2	C	C	C
Scheduler[c]	EURIX	2.2, 3.2	P	C	C
Context-Aware Preservation Manager[d]	LTU	2.2, 3.2	N	P	C
PoF Middleware Core Components					
Forgettor[e]	LUH	2.2, 3.3	P	P	C
Extractor[f]	CERTH	2.2, 3.3	P	C	C
Condensator	CERTH	2.2, 3.3	N	P	C
Contextualizer[g]	USFD	2.2, 3.3	P	C	C
Navigator[h]	USFD	2.2, 3.3	N	P	C
Collector/Archiver[i]	LTU	2.2, 3.3	P	C	C
Preservation System					
Digital Repository[j]	EURIX	2.3.2, 3.4	C	C	C
Cloud Storage[k]	IBM	2.3.3, 3.4	P	C	C

Table 15: Integration plan for PoF Framework components, classified according to Section 3. For each PoF Framework release, three different integration levels are used: *Complete* (C), *Partial* (P), *None* (N).

[a] First release: preliminary implementation of CMIS server, trigger resource preservation and restore.
[b] First release: CMIS server exposed by Alfresco, trigger resource preservation on a subset of available contents (text).
[c] First release: basic functionalities for two priority workflows, jobs scheduling.
[d] Second release: preliminary prototype implemented, input from PoF Reference Model.
[e] First release: component design completed, web service with preliminary APIs. Second release: integrate functionalities of preliminary prototype.
[f] Second release: update functionalities for text and image analysis (e.g. concept detection).
[g] First release: contextualization implemented only for text documents.
[h] Second release: preliminary prototype for basic search, browsing contents with associated context.
[i] First release: preliminary implementation of CMIS client, content packaging.
[j] First release: adopt available solution to be used out of the box or with minimal changes.
[k] First release: subset of available PDS APIs used for ingest, access, update. Second release: integrate updated version of Storlet Engine.

6 Test Environment

The activities related to development and integration of architecture components will be supported by a test environment available to all partners. We make use of virtualization to test the components developed by each WP. The dataset used for the preliminary integration is composed by different content types (pictures, text documents, videos, web pages) provided by the two user applications.

Test Infrastructure

The test environment for the deployment of the PoF Framework components, hosted by EURIX, in depicted in Figure 7. The test infrastructure includes a dedicated DELL PowerEdge R320 server, equipped with Linux (Ubuntu 12.04 LTS 64-bit), and a 8 TB NAS for data storage, connected via dedicated Gb Ethernet connection.

The address of the test server is `forgetit.eurixgroup.com` (see Figure 7) and will be used to publish project public demos, too.

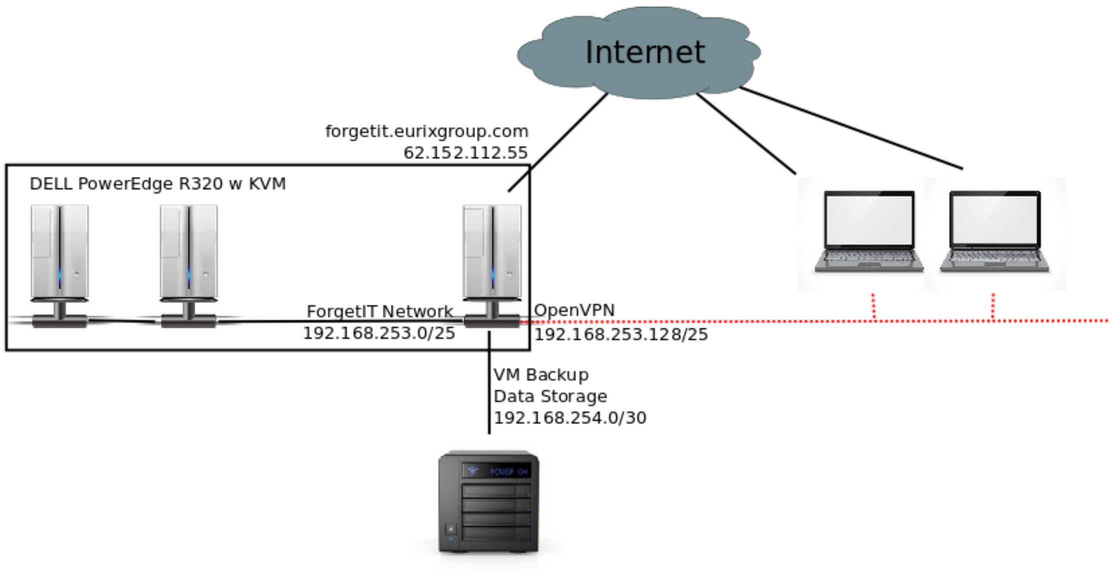

Figure 7: Configuration of the test environment.

Virtualization

The testbed server provides a virtualization environment based on Kernel-based Virtual Machine (KVM) [27]. KVM is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V) and consists of a loadable kernel module providing the core virtualization infrastructure and a processor specific module.

Each system or component to be tested and integrated in the ForgetIT platform can be deployed in a virtual machine. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc. The kernel component of KVM is included in the main Linux kernel starting from version 2.6.20. Virtual disks can be converted to/from other virtualization formats to be used with other virtualization solutions, such as VirtualBox, VMWare or XEN.

Network Configuration

During the first development phase access to applications and demos will be available from outside only to project partners, using a VPN connection to grant access to a ForgetIT dedicated private network. This will guarantee the appropriate level of confidentiality for all development activities under the consortium agreement. Then the project will provide public access to the PoF Framework for dissemination purposes.

A ForgetIT dedicated network has been created, the IP addresses of the different services available will be shared among all partners. An FTP area is also available for uploading and sharing huge files for the integration and test (e.g. virtual machines, installers, configuration files, test samples, etc.).

Test Dataset

The target of the first PoF Framework release is to test and validate the defined APIs and the protocols and to implement an end-to-end workflow based on the two priority workflows described in Section 4.2.

The dataset used for the preliminary integration tests contains pictures, text documents, web pages and short videos, with different formats, provided by the user applications. For example the *Peter Stainer* photo collection for the Semantic Desktop (see D9.1 [7]) and the *Spielwarenmesse Press Release* dataset for TYPO3 will be used for testing. Dedicated datasets will be provided by other partners for testing specific components, for example text documents for contextualization or images for feature extraction.

Development and Integration

The initial activities of development and test for each component will be performed by each partner at their own institution. For example the Cloud Storage Service based on OpenStack Swift will be initially developed and tested at IBM premises and when a new release is ready for testing by other partners will be deployed to the test environment.

For specific purposes or events, the partners could decide to deliver part of or the whole system to other servers and infrastructures, for example in case of specific testbed events which require datasets or other resources which cannot be supported or managed in the test environment.

7 Candidate Components for the PoF Framework

In this Section we describe existing software applications which can be used to implement specific components in the overall architecture. We focus here on candidate solutions for two WP8 components: the Digital Repository and the PoF Middleware. For the Digital Repository the evaluation criteria and the results of the assessment are reported. For the PoF Middleware we list existing applications, based on different approaches in the field of enterprise application integration, which can be used to implement the ESB and the integration framework for the components developed by technical WPs. Concerning other components listed in the previous Sections, namely the two Active Systems (PIMO and TYPO3), the Cloud Storage Service and all components internal to the PoF Middleware, they are described in detail in the deliverables of corresponding WPs.

7.1 Digital Repository solutions

In this Section we describe the assessment of candidate applications for the Digital Repository, one of the two components inside the Preservation System. The Digital Repository exposes a REST API to the PoF Middleware and integrates with Cloud Storage Service.

The approach described in the following focused mainly on the evaluation of existing open source solutions compliant to OAIS, developed by other research projects and initiatives in the field of digital preservation. The reason behind this approach is twofold: on the one hand, several initiatives already developed preservation platforms inspired by OAIS, supporting a large variety of content types and formats, and adopting an existing solution for the Digital Repository rather than reinventing the wheel is crucial to ensure effective use of project resources, in this way effort is focused on implementing the PoF Middleware to support core ForgetIT principles; on the other hand, commercial solutions available on the market have been discarded on purpose, since an open source and free solution rather than a proprietary implementation better suits the objectives of the project and the possibility to integrate the results from technical WPs, preventing vendor lock-in.

In the following paragraphs we first identify the assessment criteria for the evaluation, then we list the most relevant candidate solutions for the Digital Repository and provide a candidate software matrix based on the criteria above and finally discuss the results of the evaluation and the selection of the best candidate solution.

Assessment criteria

In order to evaluate candidate solutions for the Digital Repository and select the most suitable one for the PoF Framework, a list of assessment criteria has been defined. We provide a criteria catalog following the approach described for example in [28], which suggests an assessment methodology of digital preservation systems based on a market survey and shows the application of the criteria catalog to some example products.

We extended the approach in [28] by tagging each criterion as *mandatory*, *desirable* or *optional*. The requirement level associated to each criterion is assigned taking into account the specific ForgetIT context. For example, support for cloud storage is a mandatory requirement based on the PoF Framework architecture and on the need to integrate the Cloud Storage System with the Digital Repository solution selected here.

The decision-making process outlined in [28] provides three starting points for building the criteria catalog: *reference models*, *technologies* and *use cases*. Concerning reference models, ForgetIT will deliver a PoF Reference Model to extend OAIS functional and information models in order to support core ForgetIT principles. The PoF Reference Model is not yet available at the moment of writing, but we can envisage that requirements originating from PoF Reference Model will affect mainly the PoF Middleware. Concerning the Digital Repository, the main requirements are related to the smooth transition from and to Active Systems based on robust APIs and standardized protocols, to the need for a data model supporting links between digital items and extensible with context information and to the integration with external preservation aware storage systems such as the Cloud Storage System. Concerning technologies, the mandate for the project is to select open solutions which can be adopted by the digital preservation community and future adopters of the PoF Framework. Based on the expertise within the consortium and on the languages adopted to implemented the other components in the architecture, Java EE technologies have been chosen for implementing the core architecture components and the same requirement is applied to the Digital Repository candidates. Concerning the use cases, support for the main ForgetIT content types (images, text documents, web pages), content versioning and standard information packages are required.

Criteria have been split into the following categories:

- *General criteria*: general requirements for a digital preservation system;
- *Ingest criteria*: functional criteria related to the ingest process;
- *Access criteria*: functions concerning procedures for user access to ingested objects;
- *Data Management criteria*: functional criteria describing the management of ingested objects;
- *Archival Storage criteria*: functional criteria related to storage of digital items.

Assuming OAIS as reference model for functional criteria, other non functional criteria relevant for ForgetIT include software quality and costs required to install and operate the adopted solution. This can be fulfilled by selecting widely adopted open source solutions with an active community supporting software development, maintenance and documentation, which also prevents vendor lock-in and provides an adequate technical support.

Table 16 and Table 17 describe the adopted assessment criteria, their requirement level and a brief description, split according to the list above. Many criteria have been selected from the assessment procedure described in [28].

ID	General Criteria	Requirement	Description
M1	Open source	Mandatory	The platform is available as open source
M2	Out of the box	Mandatory	The user can use the platform after default installation
M3	Community support	Mandatory	The platform is adopted and supported by a large community of users
M4	Stability	Mandatory	The application is delivered as a stable platform and not as a prototype
M5	Interoperability	Mandatory	The platform supports standard formats for exchange of objects and metadata
M6	Java technologies	Mandatory	The platform is implemented with Java EE technologies
ID	**Ingest Criteria**	**Requirement**	**Description**
M7	Versions	Mandatory	The platform manages multiple versions of the ingested data
M8	Ingest Procedures	Mandatory	The user can define policies for ingesting contents; user profiles and ACL
D1	Metadata Registry	Desirable	The platform can be extended to support new metadata schema
D2	Format Registry	Desirable	The platform can be extended to support new formats
D3	Ingest Queue	Desirable	Jobs for batch ingestion allowed
O1	Ingest GUI	Optional	Ingestion using guided procedure offered by the GUI
ID	**Access Criteria**	**Requirement**	**Description**
M9	Access Procedures	Mandatory	The user can define policies for accessing archived contents
M10	Accounting	Mandatory	The platform provides an registration process to manage contents
D4	Search	Desirable	The platform provides advanced search engine for the user to find stored data
D5	Objects Dissemination	Desirable	The user is allowed to export stored element into various dissemination formats
D6	Metadata Dissemination	Desirable	The user is allowed to export metadata schema of stored element into various formats
D7	Federation	Desirable	The platform provides common search and retrieval methods for federated access
D8	Extensibility	Desirable	The platform can be extended with custom add-ons and plug-ins

Table 16: General, Ingest and Access Functional Criteria

ID	Data Management Criteria	Requirement	Description
M11	Descriptive Information	Mandatory	The platform manages AIP descriptive information
M12	Database Management	Mandatory	Management of internal database, including update and maintenance
M13	Virus Check	Mandatory	Integrated tools for virus check of ingested content
M14	Standards	Mandatory	Standard metadata formats for information packages (e.g. METS, PREMIS)
D9	Audit Trail	Desirable	Consistency checks for information stored in the database
D10	Content Quality Control	Desirable	Integrated tools for Quality Control
O2	Reporting	Optional	Functions to produce reports about stored information
O3	Metadata and Format Migration	Optional	Embedded tools for converting metadata and formats based on policies
O4	Dashboard	Optional	Dashboard for job monitoring and administration
ID	**Archival Storage Criteria**	**Requirement**	**Description**
M15	Object Identification	Mandatory	The platform manages content identifiers, assigned uniquely
M16	Cloud Storage	Mandatory	The platform supports cloud storage services
M17	Original Content	Mandatory	The original file received by the producer is stored in the archive
D11	Content Organization	Desirable	The user is allowed to organize data hierarchically
D12	Object Type Support	Desirable	Virtually any content type is supported
D13	No Limits	Desirable	The number or the size of ingested digital objects is not limited, ability to deal with huge files
O5	Timeline	Optional	The platform provides a time line of the objects organization
O6	Logical Storage Organization	Optional	The platform maps the conceptual organization to database objects
O7	Normalization	Optional	Automatic creation of preservation and access copies with normalization tools

Table 17: Data Management and Archival Storage Functional Criteria

The following disclaimers point out the limitations of the adopted approach as well as some considerations related to ForgetIT context:

Disclaimer 1 Two OAIS functional entities have not been mentioned above: Preservation Planning and Administration. The former will be covered mainly by the Cloud Storage System (preservation tasks implemented as Storlets) and by other components in the PoF Middleware. Concerning the latter, it is related to the day-to-day management of the archive, to auditing of producers and agreements with them and also to monitoring of archive operations and is less relevant in the specific research context of ForgetIT.

Disclaimer 2 It is worth mention that a systematic assessment procedure is beyond the scope of ForgetIT. Other EU initiatives have been funded in order to provide technology scouting and assessment in evaluating digital preservation tools and applications. Further information concerning assessment criteria is available in [29], [30], [31] and [32]. Recently the Presto4U project [33] published the assessment results for two platforms, P4 and Archivematica, as part of the technology assessment based on standards defined by ISO and IEEE, and other platforms such as DSpace will be added to the next technology assessment report. Concerning OAIS compliance, a detailed list of requirements is reported in [20], some of these criteria have been selected for the ingest and access assessment of the candidate digital preservation platforms.

Disclaimer 3 The approach adopted here is based on a criteria catalog rather than on a single label summarizing the results of several assessment criteria. An example of a single measure approach is the Technology Readiness Level (TRL) [34], used to assess the maturity of evolving technologies, where the assessment is performed during its development and provides a common understanding of technology status, although readiness does not necessarily fit with appropriateness or technology maturity. TRL gained popularity also in software applications.

Disclaimer 4 Commercial and proprietary solutions haven't been considered here.

Candidate solutions for the Digital Repository

For the selection of a candidate implementation of the Digital Repository we identified a list of platforms, selecting among the outcomes of relevant projects and initiatives in the field of digital preservation or among popular solutions maintained and supported by an active community. The following platforms have been selected:

- DSpace, described in Table 18
- RODA, described in Table 19
- Archivematica, described in Table 20
- Fedora, described in Table 21
- P4, described in Table 22
- iRODS, described in Table 23

DSpace and Fedora are content management systems and also popular solutions for institutional repositories; RODA, Archivemativa and P4 claim they are compliant to OAIS (Compliance with the OAIS model is difficult to demonstrate and measure): RODA is a preservation system built on top of Fedora developed by Keep Solutions, Archivematica is a preservation system by Artefactual and P4 is a preservation platform originating from AV broadcaster domain and developed by EURIX.

As already mentioned, proprietary solutions have not been taken into account in the evaluation and therefore have not been selected. For each platform we provide a fact sheet with a short description of the following features, based on available documentation and preliminary tests on default installations available in the ForgetIT testbed:

- Open source license, documentation, support community, last stable release
- Programming language, adopted technologies, runtime environment
- APIs and protocols for integration
- Archive data model and packaging
- Supported content types (see D9.1 [7])
- Integration of Cloud Storage Service and other components

Additional information, including technical details and support are available on the reference web site provided for each solution. The descriptions are meant to provide an overview of each candidate and are preliminary to the actual assessment based on the criteria reported at the beginning.

Name	**DSpace**
Project Page	`http://www.dspace.org`
License	BSD
Short Description	DSpace is an out of the box open source repository application for delivering digital content to end-users, typically used for creating open access repositories for scholarly and/or published digital content. It is considered the most widely used open source repository software for non-profit and commercial organisations. DSpace captures, stores, indexes, preserves and redistributes an organization's research material in digital formats. Research institutions worldwide use DSpace for a variety of digital archiving needs - from institutional repositories (IRs) to learning object repositories or electronic records management, and more. DSpace can be customized and extended. An active community of developers, researchers and users worldwide contribute to DSpace community. While DSpace shares some feature overlap with content management systems and document management systems, the DSpace repository software serves a specific need as a digital archives system, focused on the long-term storage, access and preservation of digital content.
Source Code, Documentation, Community	Source code on Sourceforge [35] and GitHub [36], documentation on project web site [37], project community supported by DuraSpace [38] and institutions supporting DSpace.
Last Stable Release	4.2 (July 2014), assessment based on version 3.2 (July 2013)
Language, Runtime	Java, application server and RDBMS required
APIs and Protocols	REST APIs, supports OAI-PMH and SWORD
Data Model and Packaging	DSpace defines a structured data model [39], representing digital contents in terms of collections, communities, items, and sites, associating different levels of support for objects to be preserved. AIP is a Zip file containing a METS manifest and all related content bitstreams. Each bitstream is associated with one bitstream format. For preservation, it is important to capture the specific formats of files that users submit. In DSpace, a bitstream format is a unique and consistent way to refer to a particular file format. An integral part of a bitstream format is an either implicit or explicit notion of how material in that format can be interpreted. Each bitstream format additionally has a support level, indicating how well the hosting institution is likely to be able to preserve content in the format in the future. There are three possible support levels that bitstream formats may be assigned by the hosting institution: *Supported*, *Known* or *Unsupported*. Although DSpace provides some default values for Supported, Known and Unknown formats, each institution should determine the appropriate values based on local preservation strategy. Each item has one qualified Dublin Core metadata record. Other metadata might be stored in an item as a serialized bitstream, but Dublin Core is stored for every item for interoperability and ease of discovery. The Dublin Core may be entered by end-users as they submit content, or it might be derived from other metadata as part of an ingest process. Items can be removed from DSpace in one of two ways: they may be 'withdrawn', which means they remain in the archive but are completely hidden from view or may also be 'expunged' if necessary, in which case all traces of it are removed from the archive. DSpace holds descriptive, administrative and structural metadata.
Content Types, Formats, Standards	Virtually any type of file, regardless of format or extension can be stored in DSpace. Support here is defined as being able to upload the file, and offering it for download to end users. For text formats, DSpace offers full-text indexing and searching. For images, DSpace offers thumbnail generation and display. DSpace supports HTML documents keeping cross references, relevant for TYPO3. Packagers translate between DSpace Item objects and a self-contained external representation, or "package".

Content Types, Formats, Standards (continued)	A Package Ingester interprets, or ingests, the package and creates an Item. A Package Disseminator writes out the contents of an Item in the package format. A package is typically an archive file such as a Zip or ”tar” file, including a manifest document which contains metadata and a description of the package contents. DSpace Simple Archive Format can be used for export and ingest. Currently, handles are used as internal identifiers. DSpace uses the Handle System from CNRI as the persistent identifier for each digital object. Handles are resolved to actual URLs via a resolution service. Handles in DSpace (and elsewhere) are currently implemented as HTTP URIs, but can also be modified to work with future protocols. The Handle system supports existing bibliographic identifiers such as ISBN or ISSN.
OAIS Compliance	OAIS also serves as a framework for developers of digital repository software. The impact of the OAIS model on DSpace is apparent, even if DSpace has evolved independently following several requirements. Data Management functionality leverages DSpace Data Model for handling archived items, bitstreams and metadata. Ingest is implemented by a batch ingester and a web submit UI. For Archival Storage, DSpace offers two means for storing bitstreams: the first is in the file system on the server; the second is using SRB (Storage Resource Broker), a data grid management system enabling distributed storage. Both are achieved using a simple, lightweight API. SRB is a storage manager that offers unlimited storage and straightforward means to replicate (in simple terms, backup) the content on other local or remote storage resources. Concerning Access, DSpace allows end-users to discover content in a number of ways, including via external reference, such as a Handle (containing the AIP ID), searching for one or more keywords in metadata or extracted full-text, browsing though title, author, date or subject indices, with optional image thumbnails. DSpace’s indexing and search module provides a configurable Lucene-based search engine and a API which allows for indexing new content, regenerating the index, and performing searches on the entire corpus, a community, or collection. Web UI enables views of different indexes and browsing. DSpace identifies two levels of digital preservation: bit preservation, and functional preservation. Bit preservation ensures that a file remains exactly the same over time (not a single bit is changed) while the physical media evolve around it. Functional preservation goes further: the file does change over time so that the material continues to be immediately usable in the same way it was originally while the digital formats (and the physical media) evolve over time. Some file formats can be functionally preserved using straightforward format migration. Other formats are proprietary, or for other reasons are much harder to preserve functionally. Compared to other platforms, DSpace does not include, in the vanilla installation, tools for migrating Bitstreams from one format to the other, but these can be easily written using DSpace Java API. Concerning Administration, DSpace offers also additional features such as usage and system statistics, a checksum checker and reports. The purpose of the checker is to verify that the content in a DSpace repository has not become corrupted or been tampered with. The functionality can be invoked on an ad-hoc basis from command line, or configured via cron or similar. Options exist to support large repositories that cannot be entirely checked in one run.
Integration of ForgetIT Components	A plugin manager is provided, although deeper analysis on the source code is required. An Add-on mechanism is provided to extend DSpace with additional components. Extension and add-ons provided by the community are maintained on the project wiki. DSpace supports DuraCloud as cloud storage solution, in order to integrate PDS additional software and a plugin must be implemented.

Table 18: DSpace

Name	**RODA**
Project Page	`http://www.roda-community.org/`
License	GNU LGPLv3
Short Description	RODA is a complete digital repository that delivers functionality for all the main units of the OAIS reference model, is maintained by KEEP SOLUTIONS [40], and is built on top of Fedora (see Table 21). RODA is based on open source technologies and is supported by existing standards such as METS, EAD and PREMIS. A plug-in and task scheduling mechanism is provided to add more functionality to the system (e.g. new preservation events, alerts, tools, etc.). The repository natively supports normalization on ingest for different file formats. RODA can be extended to comply with more file formats or better preservation action tools. Support for migration-based preservation actions is built into the system. Preservation actions and management within RODA is handled by a task scheduler, defining the set of rules that trigger specific actions, and when these should take place. Preservation actions include format conversions, checksum verifications, reporting (e.g. to automatically send SIP acceptance/rejection emails), notification events, etc. The basic services in RODA are provided by Fedora Commons, the application framework that supports RODA. These services account for elementary tasks at the Data Management and Archival Storage level. Examples of such services are: store and index a digital object, add a data stream to a Fedora object, get a data stream, purge an object, find objects and list data streams. RODA Core Services are responsible for carrying out more complex tasks such as handling the ingest workflow, querying the repository in advanced ways and carrying out administrative functions on the repository. RODA enables tight integration of systems already existing in the client institution.
Source Code, Documentation, Community	Source code available on GitHub [41], documentation available on project web site [42], community is supported by KEEP SOLUTIONS [40] and institutions adopting RODA.
Last Stable Release	1.2.0 (October 2013), assessment based on version 1.1.0 (July 2013)
Language, Runtime	Java, requires application server and RDBMS. Built on top of Fedora.
APIs and Protocols	REST and SOAP APIs, supports OAI-PMH
Data Model and Packaging	RODA's content model is atomistic and very much PREMIS-oriented. Each intellectual entity is described by an EAD-component metadata record. These records are organized hierarchically in order to constitute a full archival description but are kept separately within the Fedora content model. Relationships between EAD-components are created using Fedoras own RDF linking mechanism. Additionally, each leaf record (i.e. a file or an item) is linked to a representation object, i.e. a Fedora object that embeds all the files and bitstreams that compose the digital representation. Finally, each of these objects are linked together by a set of PREMIS entities that maintain information about the digital objects provenance and history of events (PO nodes). Each preservation event that takes place inside the repository is recorded as a new preservation-event node. Special events, like format migrations, establish relationships between two preservation-representation nodes. These are called linking events. Each preservation event is executed by an agent, whether this be a system user or an automatically triggered software application. The agent that triggered the event is recorded in PO agent nodes.
Content Types, Formats, Standards	RODA is capable of ingesting and normalizing (according to the preservation plan in place) text documents, raster images, relational databases, video, and audio. A plug-in mechanism enables RODA to support additional formats. RODA follows open standards using EAD for description metadata, PREMIS for preservation metadata, METS for structural metadata, and several standards for technical metadata (e.g. NISO Z39.87 for digital still images).

OAIS Compliance	RODA is composed of several functional modules supporting processes of a common archival information system. Ingest is composed by a configurable multi-step workflow that validates submitted information and also extracts technical metadata from ingested files. The ingest process also normalizes formats according to the preservation policy in place and includes both automatic and human quality assurance steps. RODA supports the ingest of new digital material as well as associated metadata in 4 distinct ways: (1) online submission (self-archiving), (2) off-line submission using an client application, (3) batch import, and (4) integration with third-party document management software via invocation of SOAP Services or client API. SIPs are submitted to a series of tests to assess their integrity, completeness and conformity to the ingest policy. After decompressing the SIP, the validation process performs different tasks, such as virus check, METS envelope syntax check, SIP completeness check, file integrity check, descriptive metadata check, preservation metadata check, representation check (at least one representation exists within the SIP) and normalization. Representations whose format do not conform to the preservation formats defined by the preservation policy are automatically converted to the correct format. The original representation is maintained by the repository. Descriptive metadata is based on ISAD. RODA fully implements a configurable ingest workflow that not only validate SIPs, but also enables manual appraisal by data management professionals. Preservation management is handled by scheduled events. Preservation actions include format converters, checksum verifications, reporting tools (e.g. to automatically send SIP acceptance/rejection emails), etc. As a fallback strategy, the system always retains the original versions of digital representations, so that an emulation preservation strategy still remains viable in the future. RODA implements preservation planning through the possibility of running and scheduling preservation actions right in the administration module. Administration components allow editing of the descriptive metadata and definition of rules for preservation interventions such as scheduling integrity checks, initiate a format migration processes, or control the users or groups that are authorized to perform certain actions in the repository. RODA includes administration features such as user management, reporting, ingest workflow configuration, log viewer, permissions management, etc. Quality assurance and preservation metadata ensure authenticity of records while providing traceable records of all changes and events that occur to a digital representation. All actions performed in the repository are logged for security and accountability reasons. Access to data is provided through embedded web viewers and downloads. Several versions of the same data are provided, including the originally ingested digital representation. The consumer is able to browse over available collections to view or download digital representations kept in the repository. Depending on the type of the digital object, different viewers or disseminators are used. For example, text documents are delivered to consumers without resorting to any particular artifacts. They are delivered in PDF format, so the consumer should use its favourite PDF viewing application. Documents composed of several images (such as digitised works) on the other hand are displayed in special Web viewing applications that allow consumers to navigate through the pages of the representation. Data storage is managed by Fedora Commons, the data layer backend. Data is stored on the file system separately from the metadata.
Integration of ForgetIT Components	RODA exposes all its functionality via Web Services. Java APIs are available to integrate external components programmatically. Integration with cloud storage requires customization of the underlying Fedora services.

Table 19: RODA

Name	**Archivematica**
Project Page	`https://www.archivematica.org`
License	GNU AGPL v3
Short Description	Archivematica is a free and open source digital preservation system that is designed to maintain standards-based, long-term access to collections of digital objects. Archivematica uses a micro-services design pattern to provide an integrated suite of software tools that allows users to process digital objects from ingest to access in compliance with the ISO-OAIS functional model. Users monitor and control the micro-services via a web-based dashboard. Archivematica uses METS, PREMIS, Dublin Core and other best practice metadata standards. Archivematica implements format policies based on an analysis of the significant characteristics of file formats. Archivematica is maintained by Artefactual Systems [43], in collaboration with UNESCO and other institutions.
Source Code, Documentation, Community	Source code available on GitHub [44], documentation on Archivematica wiki [45], community supported by Artefactual Systems [43]
Last Stable Release	1.1 (May 2014), assessment based on version 0.10 (April 2013)
Language, Runtime	Python for implementing micro-services, requires Django MVC framework. Virtual Appliance provided for different virtualization environments (VirtualBox, VMWare, KVM)
APIs and Protocols	REST APIs, default access system is AtoM. Provides export to DSpace format. Programmatic access to indexed AIP is available through Elasticsearch [46].
Data Model and Packaging	SIP based on METS, normalization process during ingestion. LoC BagIt format (zip) used for AIP. Export to DSpace data model is supported, Archivematica can act as dark-archive for DSpace, providing back-end preservation functionality while DSpace remains the user deposit and access system. Archivematica supports also DIP upload to AtoM and CONTENTdm services.
Content Types, Formats, Standards	METS supported for ingest and access. PREMIS and DC are supported standards for preservation and descriptive metadata. Tested with documents, pictures and videos. Defines access and preservation formats for each media type and includes normalization tools (mainly ffmpeg).
OAIS Compliance	Archivematica implements a micro-service approach to digital preservation. The Archivematica micro-services are granular system tasks which operate on a conceptual entity that is equivalent to an OAIS information package. The physical structure of an information package will include files, checksums, logs, submission documentation, XML metadata, and others. These information packages are processed using a series of micro-services. Micro-services are provided by a combination of Archivematica Python scripts and one or more of the free, open source software tools bundled in the Archivematica system. Each micro-service results in a success or error state and the information package is processed accordingly by the next micro-service. There are a variety of mechanisms used to connect the various micro-services together into complex, custom workflows. Preservation plans available for different media types, based on analysis of the significant characteristics of the files. The user dashboard provides interface mapped onto OAIS functional entities. The web dashboard allow users to process, monitor and control the Archivematica workflow processes. It is developed using Python-based Django MVC framework. The Dashboard provides a multiuser interface that will report on the status of system events and make it simpler to control and trigger specific micro-services. This interface allows users to easily add or edit metadata, coordinate AIP and DIP storage and provide preservation planning information.

OAIS Compliance (continued)	Archivematica maintains the original format of all ingested files to support migration and emulation strategies. However, the primary preservation strategy is to normalize files to preservation and access formats upon ingest. Normalizing is the process of converting ingested digital objects to preservation and/or access formats. In Archivematica the original objects are always kept along with their normalized versions. Archivematica groups file formats into format policies (e.g. text, audio, video, raster image, vector image, etc.). Archivematica's preservation formats must all be open standards. Additionally, the choice of formats is based on community best practices, availability of free and open source normalization tools, and an analysis of the significant characteristics for each media type. The choice of access formats is based largely on the ubiquity of web-based viewers for the file format. Not all files can be normalized on ingest because for example there are no available Linux-based open source tools to handle the conversions and/or no agreed upon preservation formats. In addition, some filetypes are not necessarily in the best preservation format but are still so ubiquitous and well-supported that they need not be normalized at the present time. In these cases, the files are kept in their original formats. A Format Policy Registry is available to implement rules of Preservation Planning.
Integration of ForgetIT Components	Archivematica provides a full-fledged preservation platform which can be installed and used out of the box. Extending Archivematica for integration with external components requires modification of the source code. Default storage mechanism is local file system.

Table 20: Archivematica

Name	**Fedora**
Project Page	`http://www.fedora-commons.org/`
License	Apache License, v2.0
Short Description	Fedora is a digital repository, developed and maintained under the stewardship of the not-for-profit organization DuraSpace [38]. The Fedora Repository Project provides a robust open source software system based on a core repository service (exposed as web-based services with well-defined APIs) and an array of supporting services and applications including search, messaging and administrative clients. Fedora aims at ensuring that digital content is durable by providing features that support digital preservation. The FedoraCommons refers to the community surrounding the Fedora Repository Project.
Source Code, Documentation, Community	Source code available on GitHub [47], documentation on Fedora Commons wiki [48], project community supported by DuraSpace, a registry of institutions adopting Fedora is maintained.
Last Stable Release	3.7.1 (October 2013), 4.0 Beta 1 (June 2014), assessment based on version 3.7.0 (Sept. 2013)
Language, Runtime	Java, requires application server and RDBMS
APIs and Protocols	REST and SOAP APIs, supports OAI-PMH
Data Model and Packaging	Fedora Digital Object Model [49], digital objects stored internally using FOXML.
Content Types, Formats, Standards	Fedora Digital Object Model supports videos, images documents and others. FOXML format is preferred schema for ingest and access, METS (using Fedora extension) supported for ingest and access, also MPEG-21 DIDL.
OAIS Compliance	Ingest and Access available thorugh REST or SOAP APIs or Web UI. Batch ingestion supported. Supported SIP formats are FOXML or METS. Export formats are FOXML, METS and ATOM. Export to new archive or purging existing object are supported. Search possible using Web UI or REST APIs, using identifier or DC metadata. Data Management based on own object model, specific component for archive Administration is available. Archival Storage implemented by Low Level Storage interface, supporting disks and cloud services (experimental). Periodic activities for Preservation Planning are supported at the datastream level (e.g. checksums). Datastreams can be updated or migrated. Descriptive metadata can be modified, too.
Integration of ForgetIT Components	A plugin or adapter to integrate with PDS is required. DuraSpace provide their own cloud solution (DuraCloud), but it is not free. Integration of other components delivered as REST services or command line tools is possible.

Table 21: Fedora

Name	**P4**
Project Page	`http://prestoprime.eurixgroup.com/p4`
License	GPLv3
Short Description	P4 is the preservation platform developed by EU FP7 PrestoPRIME project [50]. P4 implements the main functional entities of the OAIS model for an archive managing AV content and is made up of three main components: (1) core libraries, implementing OAIS components for storage, metadata management, ingest, access, administration and preservation actions; (2) web server, providing REST interfaces for interacting with the archive; (3) web UI, providing ingest, access and administrative functionalities according to the user profile. The web server provides interfaces for ingest, access and administration. The user can ingest SIP files into the platform, get information about the status of the submitted jobs and of the whole system, search for AIP available in the archive, and get access to the DIP, through the web interface. The user interface manages local users and can connect to multiple P4 instances with different user identifiers, each associated to a specific role (consumer, producer, administrator) for that platform. The external tools and services can be integrated using a plugin framework, the motivations for this being twofold: on one hand it provides a flexible way to integrate new components (e.g. to execute some specific steps during ingestion), on the other hand the platform and the core components are decoupled from specific tools or scenarios and P4 users have access to an open framework which can be used out of the box, by configuring a minimum set of parameters. P4 includes a workflow engine, a lightweight execution environment to configure custom tasks based on external tools and services, exploiting the APIs of core modules. The external tools used to implement a specific workflow can be deployed within a P4 plugin. Tools developed within the project and integrated in P4 cover metadata extraction (e.g. MXF tools), quality assessment, storage (disks via NFS or CIFS, LTO tapes, shared or federated storage systems such as iRODS and MServe), emulation (Multivalent), SLA and monitoring, rights, search and indexing (Solr), AV material segmentation and access, format migration, fixity checks. Concerning the storage configuration, different workflows have been tested in PrestoPRIME. In particular the configuration with two copies of the master quality file was implemented either with LTO tapes (two copies on two different tapes) or with iRODS as policy-driven storage (the automatic replica rule, with periodic fixity checks was defined).
Source Code, Documentation, Community	Source code available on GitHub [51], documentation available on the web site [52], currently the platform is part of the PrestoCentre tools library [53]
Last Stable Release	2.2.0 (Dec. 2012)
Language, Runtime	Java, Servlet Container required, no RDMS (XML DB)
APIs and Protocols	REST APIs, supports OAI-PMH.
Data Model and Packaging	The data model makes use of METS as the main wrapper format for descriptive and technical metadata, as well as for mapping AV resources within the AIP. Other metadata standards are supported, such as MPEG-7 for technical metadata, PREMIS for preservation events, MPEG-21 for rights representation, DublinCore for descriptive metadata and others. P4 also supports DNX, a metadata format built on top of PREMIS vocabulary, used in Rosetta. Using P4 plugins, virtually any metadata standard can be used in the AIP. Access interface supports also OAI-PMH protocol. The data model is tailored to broadcast environment (editorial entities, master and browsing qualities, B2B contracts). No compressed formats such as zip, BagIt or tarball used for AIP, METS contains references to metadata and AV files.

Content Types, Formats, Standards	Focus on videos, but other content types can be supported defining new workflows. Based on METS, supports DC, MPEG-21 CEL, MPEG-7 AVDP, DNX, PREMIS.
OAIS Compliance	Ingest and access provided by web UI or REST APIs, using METS as unique format for all OAIS information packages, common to other platforms. An advanced search engine based on Solr allows indexing of different descriptive and technical metadata. Several solutions are available for Archival Storage, supporting local and distributed storage. Preservation Planning is provided by integrated tools for fixity checks or format migration, no scheduler is implemented in the platform, makes use of external systems (e.g. iRODS). The index is stored in a fast native-XML DB and periodic triggers are executed for backup and integrity checks of the AIP XML files. Additional preservation operation are provided by storage solutions (e.g. the LTO component). Data Management and Administration are provided by the P4 web UI, including monitoring of jobs and workflows.
Integration of ForgetIT Components	The favourite integration mechanism is making use of REST interfaces over HTTP, to get loose coupling and reduce dependencies. P4 provides a plug-in mechanism to integrate external components or services in the workflow. In order to integrate cloud services, a new storage plugin should be implemented and added to the storage layer (if we use REST APIs this should be straightforward).

Table 22: P4

Name	**iRODS**
Project Page	`https://www.irods.org`
License	BSD
Short Description	iRODS is the integrated Rule-Oriented Data-management System, a community-driven, open source, data grid software solution. It is a policy-based data management system, implementing a micro-services pattern, based on rule engine. iRODS helps manage (organize, share, protect, and preserve) large sets of computer files. Collections can range in size from moderate to a hundred million files or more totaling petabytes of data. The requirements to manage large collections of data include both a number of generic capabilities and diverse features that depend on the details of different applications. iRODS is also highly configurable and easily extensible for a very wide range of use cases through user-defined micro-services, without having to modify core code. iRODS is used by many projects and teams, small and large, national and international, computer technologists and non. iRODS includes a set of features that blend together well and augment each other to form a comprehensive whole. iRODS major features include high-performance network data transfer and a unified view of disparate data. iRODS uses unique logical names that are separate from the names as stored physically, providing a global logical name-space via the iCAT Metadata Catalog in a DBMS to keep track of the names and locations of files so users don't have to. iRODS also supports a wide range of physical storage, including Unix and Windows files systems, archival storages systems (HPSS, tapes), etc. iRODS provides easy, automated replication and backup to multiple storage devices/locations at the physical level. So, users access the files via the logical names and the system finds and gets the physical files. iRODS also manages metadata, both system (automatic) and user-defined, and stored in the iCAT Metadata Catalog running in a DBMS. Users can query the system to find, use, verify, etc. files with particular attributes (metadata). iRODS provides fine-grained controlled access, by user or group. iRODS innovative Rule Engine applies local and community policies expressed as rules and executed via server-side micro-services. Rules invoke other rules and/or micro-services making the system highly configurable for site-specific needs and automated for cost-effective administration of today's mushrooming data collections. Workflows can be executed as part of normal operation (e.g. a Rule can be run as a file is initially stored to automatically make an offsite replica) or as delayed or periodic Rules. iRODS can operate as a complete stand-alone system (utilizing storage systems, database systems, and networks underneath) and also as middleware where higher-level and application-specific software makes use of iRODS as part of its infrastructure.
Source Code, Documentation, Community	Source code and documentation available on the project wiki [54], supported by the DICE group of the University of North Carolina at Chapel Hill and the University of California San Diego.
Last Stable Release	4.0 (March 2014), assessment based on version 3.3 (July 2013)
Language, Runtime	C, Perl, Shell. Provides a service running the catalog, other nodes can be distributed, requires a RDBMS.
APIs and Protocols	iRODS provides GUI, Web, WebDAV, command line interfaces, as customized shell commands used for managing content and administering the archive, and also a Java API (Jargon) allowing programmatic integration in external systems. Development of rules for specific tasks requires a custom language.

Data Model and Packaging	Data Virtualization is the underlying idea in the iRODS data grid system. Instead of a physical naming, iRODS adopts a virtual (or logical name) for every entity that interface with user or application. The mapping from the logical name to physical name is maintained persistently in the Metadata Catalog and the mapping is done at run time by the Virtualization sub-system. The virtualization pervades all aspects of iRODS and is seamlessly integrated into the various modules. iRODS provides different types of data transfer mechanism. iRODS can get and put files from a remote storage system (which is fronted by an iRODS server) or can transfer from one storage to another (as a third-party transfer). The access of the files can be either as a single file transfer or as a whole collection/sub-collection transfer. It can transfer these files in bulk mode (when several small files are being transferred) or in parallel mode when a large file is being transferred. All these different options are optimally selected depending upon the file sizes and the number of files being transferred. iRODS data model defines logical name spaces for files (POSIX, Grid and collection attributes), users, resources, rules, micro-services and states. A resource, or storage resource, in iRODS terminology, is a software/hardware system that stores data. An iRODS resource is a logical mapping of a ”resource name” to a number of physical attributes that define the resource. The iRODS clients/servers can then operate on remote or local data on different types of resources through a common interface. Currently, iRODS supports 3 resource types - unix file system, HPSS, and Amazon S3.
Content Types, Formats, Standards	Any kind of file can be virtually stored. Provides support for metadata, search and other operations on the content.
OAIS Compliance	iRODS software was designed to allow curators utilising heterogeneous storage and computing facilities to define policies without being concerned with the technical detail of how the system implements those policies and without having to respond to changes in technical infrastructure. iRODS uses a data grid architecture, running server software and a rule Engine on each server that will become part of the virtual repository. A separate, unique iRODS iCAT Metadata Catalog uses a database to track descriptive and preservation metadata. Users determine workflows and automated tasks that the Rule Engine carries out regardless of the originating server. iRODS was not conceived as an implementation of the OAIS model, but to fullfil other requirements for a rule-oriented data management system leveraging grid technologies in the context or research and academic institutions. Nevertheless, iRODS is widely used in the research community, in high performance computing projects, and in preservation environments and digital libraries. Several principles borrowed from OAIS could be implemented in iRODS. For example, OAIS describes a standard model for access to information repositories that could be ported on top of iRODS. Within the iRODS data grid, standard functions (micro-services) are defined which can be composed into workflows to support procedures that are applied to the contents of the information repository. Data Management procedures are controlled by policies that are managed in a distributed rule engine.
Integration of ForgetIT Components	iRODS provides a mechanism for integrating external components as additional micro-services or rules. Additional components for ingest and access should be developed on top of iRODS, unless some of the add-ons developed by the community already fits with ForgetIT requirements. iRODS is already based on Data Grid and provides a policy-driven storage, enabling micro-services and rules running close to the data. This feature conflicts with one of the expected outcomes of the project, namely the cloud storage services, which should provide analogue features using the Storlet technology.

Table 23: iRODS

Other solutions, projects and initiatives

Several EU projects have been funded in the field of digital preservation. Almost all projects developed tools for digital preservation which have been partially or totally delivered as open source. The outcomes of such projects have also provided valuable feedbacks to the digital preservation community and have been often used to support the initial development of popular digital preservation systems. Typically, the main limitation of such tools is due to the short lifetime of the supporting projects, which have the primary focus of demonstrating the project objectives rather than providing stable software which is suitable for usage out of the box by others. Software engineering and maintenance require resources which cannot always be guaranteed after the project ends.

After the initial phase where the focus of projects like DELOS [55] was to raise awareness on the digital preservation issues, different initiatives such as the Digital Curation Centre and Digital Preservation Europe started, several projects have been funded focusing on research and development of digital preservation technologies, processes, audit and other technical aspects. Such group includes for example CASPAR [56], PLANETS [57], Shaman [58], PrestoPRIME [50], SCAPE [59], ENSURE [24], TIMBUS [60] and many others. Some Coordinated Actions such as Presto4U [33] have also been funded. Several national archives worldwide have developed solutions for the digital preservation. For example DPSP [61] is a collection of software applications supporting the goal of digital preservation developed by National Archives of Australia, while DVA-Profession [62] is a complete solution for digitizing video for archiving purposes.

Several initiatives such as the Digital Preservation Coalition (DPC) [63], the PrestoCentre [64] or The Planets Foundation [65] or APARSEN [66] (just to mention a few), try also to support the community in maintaining catalogs of software tools and libraries for the digital preservation. For example DPC provides technology watch and training, supporting the dissemination and adoption of best practices and technologies in the digital preservation community and the PrestoCentre library provides a list of processes, tools and techniques available to assist in the digital preservation of audiovisual materials.

The non-exhaustive list above is just intended to point out that a lot of tools, libraries and platforms have been produced by research projects, initiative and institutions, not only in Europe, hence within ForgetIT the assessment of candidate solutions for implementing a preservation system can only focus on widely adopted and supported applications.

ID	Assessment Criteria	DSpace	Archivematica	RODA	Fedora	P4	iRODS
M1	Open source	√	√	√	√	√	√
M2	Out of the box	√	√	×	×	×	√
M3	Community support	√	√	×	√	×	√
M4	Stability	√	√	×	√	×	√
M5	Interoperability	√	√	√	√	×	×
M6	Java technologies	√	×	√	√	√	×
M7	Versions	√	×	√	√	×	×
M8	Ingest Procedures	√	×	√	√	√	√
D1	Metadata Registry	√	√	√	√	×	×
D2	Format Registry	√	√	√	√	√	√
D3	Ingest Queue	√	√	√	√	√	√
O1	Ingest GUI	√	√	√	√	√	×
M9	Access Procedures	√	×	√	√	×	×
M10	Accounting	√	√	√	√	√	√
D4	Search	√	√	√	√	√	√
D5	Objects Dissemination	√	×	√	√	×	×
D6	Metadata Dissemination	√	×	√	√	×	×
D7	Federation	√	√	√	√	×	√
D8	Extensibility	√	√	√	√	√	√
M11	Descriptive Information	√	√	√	√	×	×
M12	Database Management	√	√	√	√	√	√
M13	Virus Check	√	√	√	√	×	√
M14	Standards	√	√	√	√	√	×
D9	Audit Trail	√	√	√	√	√	√
D10	Content Quality Control	×	×	×	×	×	×
O2	Reporting	√	√	√	√	×	√
O3	Metadata and Format Migration	√	√	√	×	√	×
O4	Dashboard	√	√	√	√	√	√
M15	Object Identification	√	√	√	√	√	√
M16	Cloud Storage	√	×	×	×	×	√
M17	Original Content	√	√	√	√	√	√
D11	Content Organization	√	√	√	√	×	√
D12	Object Type Support	√	√	√	√	×	√
D13	No Limits	√	√	√	√	√	√
O5	Timeline	×	×	×	×	×	×
O6	Logical Storage Organization	√	√	√	√	√	√
O7	Normalization	×	√	√	×	√	√

Table 24: Preservation Platforms Assessment

Selection of the Digital Repository: DSpace

Taking into account the assessment results reported in Table 24, among all candidate solutions for the implementation of the Digital Repository, we selected *DSpace*.

DSpace is stable and supported by a huge community of users and developers and has been adopted by about one thousand institutions worldwide as the reference solution for their institutional repositories. The maturity of the project and the stability of the last release allows usage out of the box, the documentation (including developer guide) is complete, the source code is available as open source. DSpace is developed using Java EE technologies. One difference with respect to other solutions is that DSpace already supports cloud storage services as backup for stored content. Functionalities for backup of the whole repository configuration and advanced features for content management are also valuable. Concerning the compliance to OAIS model, this is still progressing, mainly for what concerns aspects related to Preservation Planning, although in the specific ForgetIT context a preservation aware storage (Cloud Storage Service) will be used to actively preserve and curate archived content.

Fedora, the other solution from DuraSpace, is not conceived as an out of the box implementation and the amount of work required to customize and prepare the platform is still considerable. RODA, which is built on top of Fedora, provides additional features related to the actual preservation of the contents and is strongly OAIS-oriented, it is a relatively new project which could benefit from wider adoption in the future. Archivematica, on the other hand, is strongly focused on OAIS and its customization in terms of workflows and formats requires deep knowledge of source code and provides no support for cloud storage. One important feature is that Archivematica can act as a *dark archive* for a DSpace repository, providing back-end preservation functionality while DSpace remains the user deposit and access system, supporting the common interoperability standards used in the Institutional repository domain, such as OAI-PMH [67], SWORD [68] and OpenSearch [69]. Interoperability is important when choosing a specific application for the PoF Framework, because the adoption of the PoF Framework by new users would be easier. P4 and iRODS, for different reasons, cannot be considered equivalent to the previous solutions, from the ForgetIT point of view. The former is still in a early prototype status and is mainly focused on videos, although it integrates several useful technologies for AV digital preservation, while the latter was not conceived as a digital repository but as an advanced storage solution (policy-driven storage), with many features (including rules and processes executed close to data) overlapping with the Cloud Storage Service.

It is worth noting that adopting DSpace is not a constraint because SIP and DIP format is based on METS, in common with all the other platforms mentioned before, the only differences being related to the specific profile adopted. In addiction, the Digital Repository REST APIs are defined within the project and are not the ones provided by the specific platform: when moving to another Digital Repository implementation, only the component responsible for processing REST requests and for interacting with the Digital Repository should be modified, while all other components in the PoF Middleware would be the same.

7.2 PoF Middleware Solutions

The PoF Middleware plays a crucial role in the overall ForgetIT architecture, since it provides the bridge between the Active Systems and the Preservation System, as already described before. Almost all components developed in the project by the technical WPs will be integrated in the PoF Middleware.

For the implementation of the PoF Middleware, we will adopt the following approach: for the early integration, targeting the first release of the PoF Framework, we will use a lightweight solution, which should allow an easy integration of a few components in order to demonstrate the two priority workflows described in Section 4.2; then we will evaluate more complex and robust middleware solutions, depending on the requirements.

Different enterprise solutions are available, ranging from full-fledged integration suites, to integration platforms, application servers or simple Enterprise Service Bus (ESB) implementations. Even if several commercial platforms implementing an enterprise-level middleware are available, the approach adopted in the project is to take into account only free and open source solutions, supported by an active community of developers.

Concerning the evaluation criteria, the PoF Middleware should provide, among other functionalities, a communication layer for all components (the ESB), as well as the function to integrate components and to manage different business processes. A REST interface should be provided, to integrate with other components of the architecture.

The main advantage of using an ESB component in the middleware is that it can act as a transit system for carrying data between applications which can be in the enterprise or spread across the web, in this way different applications can communicate with each other using a shared protocol. The ESB provides service creation and hosting, service mediation, message routing and data transformation, with exchange of data across varying formats and transport protocols. The ESB architecture is adopted by several integration frameworks and the message oriented approach has become the reference for implementing the concept of ESB (see deliverable D5.2 [12]).

The middleware solutions listed below will be considered as possible candidates for the implementation of the PoF Middleware.

Mule ESB [70] is a lightweight Java-based ESB and integration platform enabling quick and easy connection of applications and data exchange among them. As any ESB solution, it aims at providing an easy integration of existing systems, regardless of the different technologies that the applications use. Examples of such technologies include JMS, Web Services, JDBC, HTTP, and more.

JBossESB [71] is part of the JBoss Enterprise SOA Platform. The software is based on Enterprise Application Integration (EAI) principles and is a middleware used to connect systems together, especially non-interoperable systems, providing business process monitoring and management, connectors, transaction manager, security, application containers, messaging services, naming and directory service and others.

Apache ServiceMix [72] is an open source integration container that unifies the features and functionality of several other components into a runtime platform, which can be used to build integration solutions. It provides a complete, enterprise ready ESB powered by OSGi Alliance, released under Apache License v2. Apache ServiceMix provides reliable messaging, routing and enterprise integration patterns, RESTful web services, a workflow engine based on BPEL, a JMS component and an orchestrator.

Taverna [73] is an open source and domain-independent workflow management system. The Taverna suite is written in Java and includes the Taverna Engine (used for enacting workflows) that powers both the Taverna Workbench (the desktop client application) and the Taverna Server (which allows remote execution of workflows). Taverna is widely adopted to implement digital preservation workflows and has been used also in other projects related to digital preservation, such as SCAPE [59].

Two other solutions could be also considered, even if they implement different paradigms, because they provide useful features.

Apache UIMA [74] is an Unstructured Information Management (UIM) application, a software system able to analyze large volumes of unstructured information in order to discover knowledge that is relevant to an end user. An example UIM application might ingest plain text and identify entities, such as persons, places, organizations, or relations. UIMA provides capabilities to wrap components as local or remote processors, to define data structures, data flows and workflows to implement processing pipelines. It can scale to very large volumes by replicating processing pipelines over a cluster of networked nodes.

Cloud Foundry [75] is an open source cloud computing Platform as a Service (PaaS), providing a choice of clouds, developer frameworks and application services, aiming at making faster and easier to build, test, deploy and scale applications. Cloud Foundry is developed by VMware and released under the terms of the Apache License 2.0.

The middleware solutions mentioned above will be analyzed for implementing the PoF Middleware and the results will be reported in deliverables D5.2 [12] and D8.3 [13].

8 Summary and Future Work

In this deliverable we have described the architecture of the PoF Framework, which is composed by three layers: the Active Systems (i.e. the user applications), the PoF Middleware and the Preservation System.

The main systems and components developed by the technical WPS and described here will be integrated in the overall framework. Moreover some candidate solutions for implementing two main building blocks of the architecture, namely the PoF Middleware and the Digital Repository, have also been presented.

For each component the role in the overall framework, the main technologies and interfaces have been discussed. The interfaces and the protocols used to integrate components throughout the layers are also described. A preliminary plan for integrating and testing the different components has been outlined and the setup of the testbed environment has been described.

In the next WP8 deliverables, the PoF Reference Model and the PoF Framework implementation based on the present architecture will be reported. In particular, deliverable D8.3 [13] will describe the first release of the PoF Framework, with integrated components, supported workflows and content types as well as the adopted solutions for the Preservation System and the PoF Middleware.

The first release of the PoF Framework will be tested and the collected feedbacks will be included in the second release, reported in deliverable D8.4 [76].

Acronyms

AIP Archival Information Package. 14, 16, 31, 32, 47, 50, 51, 54, 57, 58

AIS Archival Information System. 13

CMIS Content Management Interoperability Services. 11, 13, 20, 30, 31, 41

CMS Content Management System. 5, 10, 11, 17, 20, 41

DIP Dissemination Information Package. 14, 30, 54, 57, 63

ESB Enterprise Service Bus. 17, 18, 23, 44, 64, 65

MB Memory Buoyancy. 25, 36

MDA Model Driven Architecture. 7, 10

METS Metadata Encoding and Transmission Standard. 15, 31, 47, 50, 52–54, 56–58, 63

MOM Message Oriented Middleware. 13

OAIS Open Archival Information System. 3, 5, 7–10, 14–16, 18, 31, 32, 44, 45, 48, 51–58, 60, 63

PDS Preservation DataStores. 16, 31, 32, 51, 56

PIMO Personal Information Model. 19, 28, 41, 44

PoF Preserve-or-Forget. 3, 5–15, 17–31, 33–36, 39–45, 48, 63–66

PV Preservation Value. 25, 35, 36

SIP Submission Information Package. 14, 15, 30, 35, 52–54, 56, 57, 63

TRL Technology Readiness Level. 48

UML Unified Modeling Language. 7, 9–11, 33

WYSIWYG What You See Is What You Get. 20

References

[1] CCSDS. Reference Model for an Open Archival Information System (OAIS). `http://public.ccsds.org/publications/archive/650x0m2.pdf`, June 2012. Retrieved 31 July 2014.

[2] ForgetIT Project. Deliverable D3.1: Report on Foundations of Managed Forgetting, August 2013.

[3] ForgetIT Project. Deliverable D4.1: Information Analysis, Consolidation and Concentration for Preservation – State of the Art and Approach, July 2013.

[4] ForgetIT Project. Deliverable D5.1: Foundations of Synergetic Preservation, July 2013.

[5] ForgetIT Project. Deliverable D6.1: State of the Art and Approach for Contextualization, July 2013.

[6] ForgetIT Project. Deliverable D7.1: Foundations of Computational Storage Services, July 2013.

[7] ForgetIT Project. Deliverable D9.1: Application Use Cases & Requirements Document, August 2013.

[8] ForgetIT Project. Deliverable D3.2: Components for Managed Forgetting – First Release, February 2014.

[9] ForgetIT Project. Deliverable D4.2: Information Analysis, Consolidation and Concentration Techniques, and Evaluation – First Release, February 2014.

[10] ForgetIT Project. Deliverable D6.2: Contextualisation Tools – First Release, February 2014.

[11] ForgetIT Project. Deliverable D7.2: Computational Storage Services – First Release, February 2014.

[12] ForgetIT Project. Deliverable D5.2: Workflow Model and Prototype for Transition between Active System and AIS, February 2014.

[13] ForgetIT Project. Deliverable D8.3: Preserve-or-Forget Framework – First Release, August 2014.

[14] ForgetIT Project. Deliverable D8.2: Preserve-or-Forget Reference Model – Initial Model, September 2014.

[15] ForgetIT Project. Deliverable D9.2: Use Cases & Mock-up Development, February 2014.

[16] ForgetIT Project. Deliverable D10.2: Application Mockups and Prototypes, February 2013.

[17] OASIS. Content Management Interoperability Services (CMIS). https://www.oasis-open.org/committees/cmis. Retrieved 31 July 2014.

[18] David Chappell. *Enterprise Service Bus*. O'Reilly Media, Inc., 2004.

[19] Library of Congress. Metadata Encoding and Transmission Standard (METS). http://www.loc.gov/standards/mets. Retrieved 31 July 2014.

[20] CCSDS. Audit and Certification of Trustworthy Digital Repositories (TDR). http://public.ccsds.org/publications/archive/652x0m1.pdf, September 2011. Equivalent to ISO 16363:2012, retrieved 15 July 2014.

[21] OpenStack. Open Source Cloud Computing Software. https://www.openstack.org. Retrieved 31 July 2014.

[22] TYPO3. Source Code Repository. http://typo3.org/about/typo3-the-cms. Retrieved 31 July 2014.

[23] ObjectDB. http://www.objectdb.com. Retrieved 31 July 2014.

[24] ENSURE Project. http://ensure-fp7-plone.fe.up.pt/site. Retrieved 31 July 2014.

[25] ENSURE Project. PDS Interface Specification. http://ensure-fp7-plone.fe.up.pt/site/deliverables/pds-cloud-external-interface-specification/at_download/file. Retrieved 31 July 2014.

[26] Object Management Group (OMG). Unified Modeling Language (UML). http://www.uml.org. Retrieved 31 July 2014.

[27] Kernel-based Virtual Machine (KVM). http://www.linux-kvm.org. Retrieved 31 July 2014.

[28] Uwe M. Borghoff, Peter Rödig, Lothar Schmitz, and Jan Scheffczyk. *Long-term Preservation of Digital Documents*. Springer, 2006.

[29] The Technology Watch Report Institutional Repositories in the Context of Digital Preservation. http://www.dpconline.org/advice/technology-watch-reports?q=technology+watch+report. Retrieved 31 July 2014.

[30] Trusted Digital Repositories: Attributes and Responsibilities. http://www.oclc.org/content/dam/research/activities/trustedrep/repositories.pdf?urlm=161690. Retrieved 31 July 2014.

[31] An Audit Checklist for the Certification of Trusted Digital Repositories. http://library.oclc.org/cdm/ref/collection/p267701coll33/id/408. Retrieved 31 July 2014.

[32] Report for the DCC/DPC Workshop on Cost Models for Preserving Digital Assets. http://www.dpconline.org/graphics/events/050726workshop.html. Retrieved 31 July 2014.

[33] Presto4U Project. https://www.prestocentre.org/4u. Retrieved 31 July 2014.

[34] Technology Readiness Level (TRL). http://en.wikipedia.org/wiki/Technology_readiness_level. Retrieved 31 July 2014.

[35] DSpace SourceForge Repository. https://sourceforge.net/projects/dspace/files. Retrieved 31 July 2014.

[36] DSpace GitHub Repository. https://github.com/DSpace/DSpace. Retrieved 31 July 2014.

[37] DSpace Documentation. https://wiki.duraspace.org/display/DSDOC3x/DSpace+3.x+Documentation. Retrieved 31 July 2014.

[38] DuraSpace. http://www.duraspace.org. Retrieved 31 July 2014.

[39] DSpace Data Model. https://wiki.duraspace.org/display/DSDOC3x/Functional+Overview#FunctionalOverview-DataModel. Retrieved 31 July 2014.

[40] Keep Solutions. http://www.keep.pt. Retrieved 31 July 2014.

[41] RODA GitHub Repository. https://github.com/keeps/roda. Retrieved 31 July 2014.

[42] RODA Documentation. https://github.com/keeps/roda/wiki/Developer-guide. Retrieved 31 July 2014.

[43] Artefactual Systems. http://www.artefactual.com. Retrieved 31 July 2014.

[44] Archivematica GitHub Repository. https://github.com/artefactual/archivematica. Retrieved 31 July 2014.

[45] Archivematica Documentation. https://www.archivematica.org/wiki/Documentation. Retrieved 31 July 2014.

[46] Elasticsearch. Open Source Distributed Real Time Search & Analytics. http://www.elasticsearch.org. Retrieved 31 July 2014.

[47] Fedora GitHub Repository. https://github.com/fcrepo. Retrieved 31 July 2014.

[48] Fedora Documentation. https://wiki.duraspace.org/display/FEDORA37/Fedora+3.7+Documentation. Retrieved 31 July 2014.

[49] Fedora Digital Object Model. https://wiki.duraspace.org/display/FEDORA37/Fedora+Digital+Object+Model. Retrieved 31 July 2014.

[50] PrestoPRIME Project. http://www.prestoprime.eu. Retrieved 31 July 2014.

[51] P4 GitHub Repository. https://github.com/prestoprime/p4. Retrieved 31 July 2014.

[52] PrestoPRIME Preservation Platform (P4). http://prestoprime.eurixgroup.com/p4. Retrieved 31 July 2014.

[53] P4 Documentation. https://www.prestocentre.org/library/tools/p4. Retrieved 31 July 2014.

[54] iRODS Documentation. https://www.irods.org/index.php/Documentation. Retrieved 31 July 2014.

[55] DELOS Project. http://www.dpc.delos.info. Retrieved 31 July 2014.

[56] CASPAR Project. http://www.casparpreserves.eu. Retrieved 31 July 2014.

[57] PLANETS Project. http://www.planets-project.eu. Retrieved 31 July 2014.

[58] Shaman Project. http://shaman-ip.eu. Retrieved 31 July 2014.

[59] SCAPE Project. http://www.scape-project.eu. Retrieved 31 July 2014.

[60] TIMBUS Project. http://timbusproject.net. Retrieved 31 July 2014.

[61] National Archives of Australia. Digital Preservation Software Platform (DPSP). http://dpsp.sourceforge.net. Retrieved 31 July 2014.

[62] Austrian Mediathek. DVA-Profession. http://www.dva-profession.mediathek.at. Retrieved 31 July 2014.

[63] Digital Preservation Coalition (DPC). http://www.dpconline.org. Retrieved 31 July 2014.

[64] PrestoCentre. Keeping Audiovisual Content Alive. https://www.prestocentre.org. Retrieved 31 July 2014.

[65] Open Planets Foundation. http://www.openplanetsfoundation.org. Retrieved 31 July 2014.

[66] APARSEN. Alliance Permanent Access to the Records of Science in Europe Network. http://www.alliancepermanentaccess.org/index.php/category/community/projects/aparsen. Retrieved 31 July 2014.

[67] OAI-PMH. Open Archives Initiative Protocol for Metadata Harvesting). http://www.openarchives.org/pmh. Retrieved 31 July 2014.

[68] SWORD. http://swordapp.org. Retrieved 31 July 2014.

[69] OpenSearch. http://www.opensearch.org. Retrieved 31 July 2014.

[70] Mule ESB. http://www.mulesoft.org. Retrieved 31 July 2014.

[71] JBoss ESB. http://www.jboss.org/jbossesb. Retrieved 31 July 2014.

[72] Apache ServiceMix. http://servicemix.apache.org. Retrieved 31 July 2014.

[73] Taverna. http://www.taverna.org.uk. Retrieved 31 July 2014.

[74] Apache UIMA. http://uima.apache.org. Retrieved 31 July 2014.

[75] Cloud Foundry. http://www.cloudfoundry.com. Retrieved 31 July 2014.

[76] ForgetIT Project. Deliverable D8.4: Preserve-or-Forget Framework – Second Release, April 2015.