# Usability of the TOSCA-MP Distributed Repository Framework

Deliverable D5.4


| | |
|---|---|
| TOSCA-MP identifier: | TOSCAMP-D5.4-DTO-v08 |
| Deliverable number: | D5.4 |
| Author(s) and company: | Stefan Kubsch (DTO)<br>Frank Gläser (DTO) |
| Internal reviewers: | Mike Matton (VRT) |
| Work package / task: | WP5 |
| Document status: | Final |
| Confidentiality: | Restricted |

| Version | Date | Reason of change |
|---|---|---|
| 1 | 2014-03-14 | Initial version of deliverable created |
| 06 | 2014-03-21 | Review |
| 07 | 2014-03-25 | Review input included |
| 08 | 2014-03-31 | Final version |

**Acknowledgement:** The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287532.

# Table of Contents



 

# 1 Executive Summary

The TOSCA-MP deliverable D5.4 describes the usability of the Distributed Repository Framework (DRF) in professional media production systems. The first version of the DRF was delivered in D5.2.1 [1] and the final version of the DRF is delivered in D5.2.2.

The DRF software has been tested and enhanced throughout the lifetime of the TOSCA-MP project. In two Integration meetings and during the final field trails, the framework was shown to experts in the field of media production. Valuable comments have been collected and will be used to advance the usability of the DRF in professional media production systems.

Functional tests have been performed in order to verify the DRF Representational State Transfer (REST) interface, and load tests with the DRF software have been used to enhance the software.

 

# 2 Introduction

## 2.1 Purpose of this Document

TOSCA-MP deliverable D5.4 describes the usability of the developed Distributed Repository Framework (DRF).

## 2.2 Scope of this Document

Deliverable D5.4 gives an overview of the usability of the Distributed Repository Framework(DRF) in the Professional Media Industry and describes the testing conducted with the DRF.

## 2.3 Status of this Document

This document represents the final version of D5.4.

## 2.4 Related Documents

Before reading this document, it is recommended to be familiar with the following documents:

- D5.1 – TOSCAMP-D5.1-DTO-v04.pdf
- D5.2.1 – TOSCAMP-D5.2.1-DTO-v03.pdf
- D5.2.2 – TOSCAMP-D5.2.2-DTO-v04.pdf
- D5.3 – TOSCAMP-D5.3-DTO-v02.pdf

 

# 3 Usability of the DRF

This deliverable is about the usability of the developed Distributed Repository Framework (DRF). Assessing the usability of the DRF means on one hand to investigate whether the gathered requirements for the DRF are met. This requires checking whether the DRF is able to facilitate the usage in the media production industries. Also the usability of the DRF in the TOSCA-MP demonstrator is important. Feedback from media production experts in the final field trials helps judging usability.

On the other hand, assessing the usability of the DRF means to investigate whether the DRF software architecture is capable of a higher data throughput as it is used in real media production systems. This has been tested by means of load tests of the final DRF software developed in the TOSCA-MP project.

The document is structured as follows: In section 3.1 the achievement of the specified requirements is discussed. Section 3.2 describes key concepts of the DRF with regard to media production systems. The usability of the DRF in the TOSCA-MP demonstrator is depicted in section 3.3, and the results of the final field trails are given in section 3.4. Section 3.5 describes the results of the load tests with the DRF software. Section 3.6 completes this deliverable with a conclusion.

## 3.1 Requirements Defined for the DRF

During the work on the deliverable D5.1 specifying the DRF, requirements of the DRF had been collected from the TOSCA-MP partners. These requirements have been consolidated and listed in section 3.1 of D5.1 [2]. On the basis of these requirements the Distributed Repository Framework has been designed. Deliverable D5.1 referenced the requirements by IDs, and this section provides a short recapitulation of the requirements and assesses their compliance in the DRF.

### *3.1.1 Non-functional Requirements of the DRF*

Table 1 provides a short recapitulation of the non-functional requirements, and compliance of the DRF with them is assessed first. The DRF software has been developed in Java on the server side and in JavaScript for the web client software. A developer, however, who wants to use the DRF in its own application, is free to use the programming language of his choice, because the application will be programmed against the web service interface, based on a Representational State Transfer (REST) library. REST libraries are available for all common programming and scripting languages thus meeting requirement RN2. The DRF software uses the Open Source Databases MySQL, Redis and CouchDB for content and configuration metadata storage and therefore meets requirement RN3. The internal messaging between a Local Module and the Main Module is based on the ZeroMQ library [4], and the DRF software runs in the Jetty HTTP server and Java servlet container [5].

| *Requirements ID* | *Description* |
|---|---|
| RN1 | Support multi-platform programming on at least Microsoft Windows and Linux platforms |
| RN2 | Support common programming and scripting languages like C++ and Java |
| RN3 | Use Open Source Software, with an active community and comprehensive library documentation |

**Table 1: Non-functional requirements of the DRF**

The DRF runs on Windows and Linux platforms. At DTO the DRF has been developed and tested on Windows 7, while the DRF software has been provided to the TOSCA-MP partners as an Ubuntu Virtual Machine. On both platforms the software runs with the same reliability. Thus requirement RN1 is met.

 

### *3.1.2 Functional Requirements of the DRF*

Table 2 provides a short recapitulation of the functional requirements, and compliance of the DRF with them is assessed next. With its good technical capabilities the DRF can be integrated in any kind of IP-based company network. The overall architecture of the DRF uses TCP/IP network technology with Local Modules and a Main Module connected by a conventional Local Area or Wide Area Network, thus meeting requirement RF2 [2]. All media assets are stored in the local repositories (thus meeting RF3) and can be accessed through the Main Modules. DRF repositories are metadata format agnostic and can store any kind of metadata (thus meeting RF4) [2]. DRF administrative metadata can be created and accessed via web services (meeting RF6) [2], and media assets can be uploaded through a Representational State Transfer (REST) interface. These are the main reasons why the DRF should be universally applicable and why it supports the defined TOSCA-MP usage scenarios very well (thus meeting RF1) [2]. Also every broadcaster has his own reasons for designing an infrastructure in one or another way. The DRF can be used in different scenarios, starting from a single Local Module which is independently usable for configurations where several Local Modules at different broadcasting facilities can be accessed via a Main Module. Therefore, also requirement RF5 is met [2].

| *Requirements ID* | *Description* |
|---|---|
| RF1 | DRF should be universally applicable and support multiple TOSCA-MP usage scenarios |
| RF2 | DRF should consist of distributed repositories, interconnected with a TCP/IP network |
| RF3 | Content and content metadata is stored only in the local repositories |
| RF4 | DRF repositories are metadata format agnostic and can store any kind of metadata |
| RF5 | Local repositories should be also independently usable |
| RF6 | DRF administrative metadata can be created and accessed via web services |

**Table 2: Functional requirements of the DRF**

All defined requirements have been met by the DRF in the TOSCA-MP project, and it was possible to react very flexibly to the needs and expectations of the TOSCA-MP partners. This led to a good usability of the Distributed Repository Framework in the TOSCA-MP demonstrator.

## 3.2 Key Concepts of the DRF with Respect to Professional Media Production Systems

In the following, the most important key concepts of the DRF are described. Also the relevance of these concepts for professional media production systems is referred to. In general, the DRF has been designed to fit a broad range of applications. Special purpose applications like the manual annotation application still use the general purpose DRF and need to add additional layers of software without modifying the core DRF software.

### *3.2.1 Distributed Repository*

The architecture of the DRF is based on TCP/IP networks technology with Local Modules and a Main Module connected by the TCP/IP network. All media assets are stored in the local repositories and can be accessed through the Main Module. The DRF is an alternative to using a central cloud system from an external service provider. The main benefit is that the content can be kept in a Local Module rather than at a remote cloud provider facility, so there are no bandwidth issues when accessing large files. There are no versioning issues with the files where you cannot be sure which file is the most recent one, you simply maintain the recent file in your Local Module and everyone can use this version of the file.

Today, media production systems are distributed over physically distant facilities. Every facility uses highly specialized repositories in its departments – specific for the locally used applications, tools or

 

work. Other facilities or departments cannot access the data because they either don't have access to the proprietary systems or there are a number of different specialized interfaces for the repositories, which are in many cases not compatible with each other. The DRF is able to integrate various types of repositories, and many locally produced metadata in a media production can now be used by other applications during the following media production steps via the Main Module.

### *3.2.2 WEB-based Interfaces (REST)*

DRF administrative metadata can be created and accessed via web services, and media assets can be uploaded through a Representational State Transfer (REST) interface. With its good technical capabilities it can be integrated in any kind of IP-based company network. Broadcasters are relatively free how to integrate the DRF in their systems. Since REST libraries are available for a wide range of programming and scripting languages, broadcasters should be able to find an appropriate library for their own development environment. The DRF provides common web service interfaces to all data in all connected repositories, and through these interface media assets are accessible from all possible locations of a media production system. For example, a broadcasting company can perform audio analysis in production facility A on content stored in a Local Module at location B, and after finishing the processing, results can be stored on a Local Module at location A. All media assets are accessible in the whole company. This leads to another advantage: A broadcaster can centralize services, e.g. for metadata creation, at a single location and this service can be used within the whole company.

### *3.2.3 Integration with Legacy Systems (Plug-in Mechanism)*

Deliverable D5.3 describes the implemented plug-in mechanism of the DRF, which allows the integration of third-party repositories in the DRF system. The plug-in mechanism has been used for the integration of the MAMMIE repository of TOSCA-MP partner VRT [6]. The main MAMMIE repository consists of over 10,000 pieces of stored media assets. Broadcasters may have large set of assets stored in their current storage system, and it is expensive and time-consuming to migrate to a new storage system. With the DRF plug-in system the content can reside in the current system and the plug-in mechanism wraps the access to the legacy system and integrates the existing media assets in the DRF. All content is accessible through the Main Module, it is transparent to the user if the content is stored in a "normal" Local Module or a Local Module wrapping a legacy system. The DRF provides a common interface to the stored data in all connected repositories. This feature is very useful for media production systems, because there often exist repositories containing large sets of valuable media assets, which can be integrated in the DRF and can be used together with other stored content. For broadcasting companies this is a great advantage.

### *3.2.4 Security*

Security is an important feature, and without Identity and Access Management (IAM) no company would consider integrating such a system in their production network. Media assets of broadcasting companies have to be protected appropriately. In a media production workflow, access rights to the media assets should be individually granted according to the job function of a user. Currently the DRF integrates Lightweight Directory Access Protocol servers (LDAP), but in principle other IAM systems can be integrated through connectors. User Authentication and Authorization are performed by the LDAP server. Read, write and delete access to the DRF can be configured in the LDAP server by adding the groups defined for the DRF access to the users (persons). This right management allows granting different data access rights to users according to their job function.

Furthermore it is possible to enable Transport Level Security (TLS) [7] and Hypertext Transfer Protocol Secure (HTTPS) [8]. This provides a reasonable guarantee that a user is communicating with the intended website by providing authentication, as well as ensuring that the content of communications between the user and the website cannot be read or forged by any third party. This is essential to ensure the content rights of the media assets stored in the DRF.

## 3.3 Usability of the DRF as Part of the TOSCA-MP Demonstrator

The TOSCA-MP demonstrator used for the final field trails (see results in section 3.4) was a very good opportunity to show the capabilities of the distributed media production system. The demonstrator was spread over a Wide Area Network, and the final field trails have shown how well all the parts of the system work together. Regarding the DRF performance in this demonstrator, the REST web service

 

interfaces in combination with file-based data storage enabled an easy integration with the MPMF and all other services in the TOSCA-MP demonstrator, because the DRF supports all necessary distributed data access requirements. The functional requirements listed in Table 1 and Table 2 guided to a good usability of the DRF in the TOSCA-MP demonstrator.

Also during the development of the DRF the distributed architecture was very helpful. The main DRF server is located in Hannover, and it is easy to deploy a new software version on this server. After the software deployment, all TOSCA-MP partners can use the latest software version instantly. This eased the development work on the demonstrator significantly and enabled a good collaboration with the TOSCA-MP partners.

## 3.4 Results of Field Trails

The final field trails have been carried out from January 20 to 21 at RAI in Turin, Italy, and from January 23 to 24, 2014 at VRT in Brussels, Belgium. This event provided a valuable exchange with experts in the field of media production (see deliverable D6.4.3 for more details [9]). Four exercises have been performed by the participants, which gave them an opportunity to use the TOSCA-MP system. One of the exercises was about handling the DRF in a distributed scenario and gave experts an impression of the methods for networked distributed storage of assets and metadata in media production workflows. After the exercise, discussion with participants took place about the DRF and three questions have been asked about the DRF, in written form as part of a questionnaire. Below is an evaluation of the recorded answers [10] for the three questions related to the DRF and its usability:

First question: **"Is the general DRF architecture useful for distributed media production workflows?"**

In general, participants agreed that the DRF is helpful for media production workflows. This was also the feedback from the discussions with the media production experts during the exercise: the architecture of the DRF would be suited for their daily work. Their estimate was that the DRF could fit in their current infrastructure.

Second question: **"Do you see any advantages or disadvantages compared to central data cloud based repositories?"**

There is no clear point of view among the participants about the differences to cloud-based systems. This could be because both systems are very close in their use. A benefit of the cloud system is the sharing of data and information; another benefit mentioned is the security and manageability. An advantage of DRF is the distributed architecture with good scalability; a side effect mentioned is that metadata could be aggregated more easily during the production process.

Third question: **"Do you know applications in which the architecture has major advantages or disadvantages?"**

Participants mentioned the following examples where the DRF would be beneficial:

- EBU collaboration projects like Eurovision Song Contest.
- Application of the DRF in the production process of programs for VRT.
- Interesting to see how integration with systems like Avid, Quantel, OpenMedia, FESAD, etc. would look like (see section 3.2.3).

## 3.5 Load Tests with the DRF Software

Load tests have been carried out with the final DRF software to get an evidence of the usability of the DRF concerning the handling of mass data. It is an important hint whether the DRF software is also capable of realistic application in media production systems.

For the tests the same software has been used as for the final field trails. The DRF was running in Ubuntu Linux Virtual Machines, the same DRF VM running in the TOSCA-MP demonstrator. For the tests the open-source software Robot Framework has been used: a generic test automation framework based on Python. The test setup included two computers, both running one DRF Local Module

 

instance, and one of them hosted the DRF Main Module in addition. Both computers had the Robot Framework editor and test runner RIDE installed. Depending on the test scenario both computers have been used to run the test scripts in parallel. This was the case when two Local Modules where connected to a Main Module. Table 3 shows the system information for both computers.

| **Local Module 1 (LM1)** | |
|---|---|
| Processor | Dual-Core AMD Opteron(tm) Processor 2218<br>2.60 GHz (2 processors) |
| RAM | 16 GB |
| Operating System | 64-bit Windows 7 |
| **Local Module 2 (LM2)** | |
| Processor | Intel(R) Core(TM) i7 CPU 860<br>2.80 GHz (4 processors) |
| RAM | 8 GB |
| Operating System | 64-bit Windows 7 |

**Table 3: Test computers system information**

| Test Cycle | Number of LM | Repeti-tions (n) | Write Time Components (min) LM1/LM2 | Write Time Res. & File (min) LM1/LM2 | Verification (min) LM1/LM2 | Deletion (min) LM1/LM2 | Total (min) LM1/LM2 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 500 | 0:39/0:35 | 2:30/2:19 | 5:06/2:50 | 0:41/0:34 | 8:56/6:18 |
| 2 | 1 | 500 | 0:38/0:35 | 2:25/2:19 | 4:43/2:48 | 0:38/0:34 | 8:24/6:16 |
| 3 | 1 | 500 | 0:39 /0:35 | 2:27/2:19 | 5:15/2:53 | 0:39 /0:33 | 9:00/6:20 |
| 1 | 1 | 1000 | 1:19/1:10 | 4:59/4:44 | 10:16/5:44 | 1:29/1:15 | 18:03/12:53 |
| 2 | 1 | 1000 | 1:20/1:11 | 5:09/4:50 | 10:14/5:37 | 1:29/1:16 | 18:12/12:54 |
| 3 | 1 | 1000 | 1:18/1:06 | 5:09/4:28 | 10:02/5:53 | 1:28/1:18 | 17:57/12:45 |
| 1 | 1 | 1500 | 2:00/1:36 | 7:42/6:19 | 15:56/8:48 | 2:22/1:37 | 28:01/18:20 |
| 2 | 1 | 1500 | 1:58/1:45 | 7:41/7:20 | 16:01/8:28 | 2:20/2:04 | 28:01/18:57 |
| 3 | 1 | 1500 | 2:00/1:45 | 7:39/6:47 | 15:34/8:42 | 2:20/2:02 | 27:34/18:36 |
| 1 | 2 | 500 | 0:37/0:33 | 2:20/2:07 | 4:57/2:54 | 0:39/0:34 | 8:33/6:08 |
| 2 | 2 | 500 | 0:38/0:33 | 2:24/2:06 | 4:55/2:51 | 0:39/0:33 | 8:36/6:03 |
| 3 | 2 | 500 | 0:38/0:33 | 2:24/2:07 | 4:56/2:49 | 0:40/0:33 | 8:33/6:02 |
| 1 | 2 | 1000 | 1:16 /1:06 | 4:51/4:21 | 9:49/5:47 | 1:30/1:11 | 17:26/12:25 |
| 2 | 2 | 1000 | 1:15 /1:06 | 4:49/4:21 | 9:52/5:49 | 1:26/1:15 | 17:22/12:31 |
| 3 | 2 | 1000 | 1:17/1:10 | 4:51/4:41 | 9:50/5:54 | 1:24/1:11 | 17:22/12:56 |
| 1 | 2 | 1500 | 1:54/1:40 | 7:19/6:40 | 15:04/8:38 | 2:22 /2:03 | 26:39/19:01 |
| 2 | 2 | 1500 | 1:55 /1:38 | 7:26/6:36 | 15:21/8:44 | 2:13/1:51 | 26:55/18:49 |
| 3 | 2 | 1500 | 2:08/1:44 | 7:37/7:20 | 15:11/8:37 | 2:21/2:15 | 27:17/19:56 |

**Table 4: DRF load test results**

 

Table 4 shows the load test results. Tests have been started with a single Local Module (LM) connected to the Main Module. Three test cycles have been carried out to obtain mean values of characteristic times of the Local Module (LM1/LM2) for writing, deleting, etc. The repetition column specifies the number of created Components and Resources. In addition, for each Resource a file has been uploaded to the DRF (size 145 KByte). The write time specifies the time needed to perform all write cycles in the DRF. This can be split in the time for creating Components in the Local Module and the time for writing resource information including the file upload. The verification time is the time needed to read metadata and files from the DRF through the Main Module and to verify them. The deletion time is the time needed to erase all Components and Resources from the DRF.

The creation of Components and Resources, including the file upload, puts the biggest load on the system, because in the test cases around 16 web service calls per second and the same number of messages are sent to the Main Module. For the tests with a single Local Module the write time increases linearly with the number of repetitions. The deviation in the measurement results for LM1 and LM2 are due to the different performance of the computers (see Table 3). This is getting worse when LM2 performs multiple I/O operations during the checking of metadata and files. These times are listed in the verification column of Table 4. The results of the Component writing and the deletion tests are approximately on the same level since in both tests the same number of web service requests are sent to the DRF. The different results for the Component and Resource writing tests are caused by the fact that in the Resource test files are uploaded to the DRF in addition. The long verification times are due to the many web service requests performed to get Component and Resource metadata from the Main Module, and then the verification of the data has to be performed, too. Also the files will be downloaded via the Main Module, so first a redirect response containing a link to the file in the Local Module is returned and in a second step the file is downloaded from the Local Module. Finally the downloaded file has to be compared to the original file, all these additional operations result in the long test runtimes.

When testing with two Local Modules connected to a Main Module the measured times are on the same level compared to the tests with a single Local Module. So the DRF software scales very well. The test results do not show any deviation from the linear characteristics. These results indicate that the web service and the messaging module of the DRF are capable of dealing with mass data, and that the architecture of the DRF has been designed for high data throughput. We consider these test setups as a realistic application of the DRF in media production systems, so the DRF passed the load test with success. DTO will continue the tests with the DRF to find out the limits of the software, but for assessing the usability of a demonstrator these load tests are good and the results are very promising.

## 3.6 Conclusions

In deliverable D5.1 requirements of the Distributed Repository Framework (DRF) had been collected from the TOSCA-MP partners. These non-functional and functional requirements have been met successfully by the DRF developed in TOSCA-MP as reported above. The distributed and web service based architecture can be used in modern media production workflows and will enhance current processes. Collaboration of media production departments of broadcasting companies could be significantly intensified when using the DRF. Moreover, the gathering and storage of metadata throughout the production processes would eliminate a common flaw of today's production systems.

The final field trails have shown that the overall architecture provides good usability in professional media production. This is the general feedback from the participants. It is also important to continue developing the DRF towards the intended markets. For the usage of the DRF in the professional media production, it is important to maintain or even intensify the contact to the media production industry in order not to lose the competitive advantages of the DRF.

Promising results of the load tests have shown a good maturity of the DRF software at this point in time. The software is an excellent foundation for further developments, which are needed to add further functionality desirable for a production system.

 

# 4 References

[1] F. Gläser, M. Weber, S. Kubsch: “Distributed Repository Framework in a first version”, Deliverable D5.2.1, TOSCAMP-D5.2.1-DTO-v03.pdf.

[2] F. Gläser, M. Weber, S. Kubsch: “Overall Architecture, Interfaces and Protocols of the Distributed Repository Framework”, Deliverable D5.1, TOSCAMP-D5.1-DTO-v04.pdf.

[3] F. Gläser, S. Kubsch: “Repository Representation Layer“, TOSCAMP-D5.3-DTO-v02.pdf.

[4] ZeroMQ, The Guide, http://zguide.zeromq.org/

[5] Jetty webserver, http://en.wikipedia.org/wiki/Jetty_(web_server)

[6] Karel Braeckman, Robbie De Sutter, Mike Matton, Tine Blomme: A Media Sharing Platform Built With Open Source Software. DMS 2010: 98-103.

[7] Transport Layer Security, http://en.wikipedia.org/wiki/Transport_Layer_Security

[8] HTTP Secure, http://en.wikipedia.org/wiki/Https

[9] Alberto Messina, Fulvio Negro, Mike Matton, Bart Vandenbroucke, Carlos Ruiz, Werner Bailer: Field trial report v2, Deliverable D6.4.3, TOSCAMP-D643-RAI-FieldtrialreportV2_FINAL.docx

[10] TOSCA-MP field trails feedback document; Field_trials_2_feedbacks_v2.xlsx

 

# 5 Glossary

**Terms used within the TOSCA-MP project, sorted alphabetically.**

| | |
|---|---|
| DRF | Distributed Repository Framework |
| MPMF | Metadata Production Management Framework |

**Partner Acronyms**

| | |
|---|---|
| DTO | Deutsche Thomson OHG, Technicolor, DE |
| IRT | Institut für Rundfunktechnik GmbH, DE |
| RAI | Radiotelevisione Italiana S.p.A. , IT |
| VRT | Vlaamse Radio- en Televisieomroeporganisatie, BE |

Acknowledgement: The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 287532.

 