D.3.3 ï Third Year Report
WP3 ï Repository Infrastructure

VersionFINAL

30 November 2011

Grant Agreement number:	231809
Project acronym:	3D-COFORM
Project title:	Tools and Expertise for 3D Collection Formation
Funding Scheme:	FP7
Project co-ordinator name, Title and Organisation:	Prof David Arnold, University of Brighton
Tel:	+44 1273 642400
Fax:	+44 1273 642160
E-mail:	D.Arnold@brighton.ac.uk
Project website address:	www.3d-coform.eu

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 231809.

Authors: Martin Doerr, Foundation for Research and Technology, Hellas (FORTH-ICS)

Contributing partner organizations: University of Brighton (UoB)

PIN scrl- Servizi didattici e scientifici per l'Università di Firenze (PIN)

Katholieke Universiteit Leuven (KUL)

Eidgenossiche Technische Hochschule Zurich (ETHZ)

Consiglio Nazionale Delle Ricerche, ISTI (CNR-ISTI)

Fraunhofer (Fhg-IGD)

Technische Universitaet Graz (TU Graz)

Centre de Recherche et Restauration des Musees de France (CNRS-C2RMF)

University of Glasgow (GU-HATII)

University of Bonn (UBonn)

Media Integration and Communication Centre, University Florence (MICC)

Victoria and Albert Museum (VAM)

Breuckmann

Spheron

Table of Contents

1 Executive Summary 5

2 Introduction and Objectives 6

3 T3.1 Scientific and Technical Coordination 8

3.1 Work planned 8

3.2 Work performed 8

3.3 Deviation from work plan 9

3.4 Plans for the next period 9

4 Integration of Repository Infrastructure Components & T3.2 Design architecture 11

4.1 Work planned 11

4.2 Work performed 11

4.3 Deviation from work plan 12

4.4 Plans for the next period 14

5 T3.3 Design and implementation of a metadata repository 15

5.1 Work planned 15

5.2 Work performed 15

5.3 Deviation from work plan 18

5.4 Plans for the next period 18

6 T3.4 Design and implementation of an object repository 20

6.1 Work planned 20

6.2 Work performed 20

6.3 Deviation from work plan 25

6.4 Plans for the next period 26

7 T3.5 Design and implementation of a query manager 27

7.1 Work planned 27

7.2 Work performed 27

7.3 Deviation from work plan 27

7.4 Plans for the next period ... 28

8 T3.6 Design and implementation of an annotation and co-reference manager ... 29
8.1 Work planned ... 29
8.2 Work performed ... 29
8.3 Deviation from work plan ... 29
8.4 Plans for the next period ... 29

9 T3.7 Implementation of a Long term Digital Preservation Management Tool ... 30
9.1 Work planned ... 30
9.2 Work performed ... 30
9.3 Deviation from work plan ... 34
9.4 Plans for the next period ... 35

10 T3.8 Implementation of Watermarking Tool ... 36
10.1 Work planned ... 36
10.2 Work performed ... 36
10.3 Deviation from work plan ... 37
10.4 Plans for the next period ... 37

11 T3.9 Integration and testing ... 38
11.1 Work planned ... 38
11.2 Work performed ... 38
11.3 Deviation from work plan ... 39
11.4 Plans for the next period ... 39

12 Publications ... 40

13 References ... 41

14 Non-public Appendices ... 42

APPENDIX A ARI API Specification ...

APPENDIX B Ingestion Tool Design ...

APPENDIX C Fundamental Categories, Relations ...

APPENDIX D IVB installation and test at VAM ...

1 Executive Summary

This document presents the work that has been done in the context of WP3 Repository Infrastructure during the third year of activity of the 3D-COFORM project.

WP3 plays a central role in the project since it is responsible for the design and the implementation of the 3D-COFORM integrated repository (RI). In the following table we present all the steps of the RI design and implementation during the three years of the project. Please note that there was a change (compared with the Description of Work) in the used numbering scheme. So, RIv3 which is extensively NJŠ Ŧ ŠNJŠ ў Ď Š Ř λ y ú K λ *fast version* of the Description of Work Ó l · t Š y ú ú 2 á

Official Revision Numbering	Actual Revision Numbering	Explanation
RI alpha	RI alpha	RI early version that was demonstrated in the first review (RI-API alpha stand-alone version), was issued at the end of Month 12, as a client-side application programming interface (RI-API) in the form of Java-library module (jar). It provided limited functionality: ingest/retrieve/query functions.
RI beta	RIv2 stand-alone	RI second stand-alone version that was demonstrated in the second review was first issued at the end of Month 21, as a client side application programming interface (RI-API) in the form of Java-library module (jar) It provided full functionality (with only access control and replica-handling missing) while it was based on remote MR installation in conjunction with remote single OR installation. It was completed in Month 25 due to the change of implementation platform (from Globus Toolkit to AFS) and the major re-engineering undertaken in OR.
RI 1.0	RIv2 webservice (never delivered)	RI first webservice version that was never officially delivered. It was planned to be issued in Month 25 but was issued at the end of Month 28, as a webservice interface and as a Java stub (webservice wrapper). It provided full MR functionality based on webservices and would separate the data transfer mechanism from the rest of RI interface (never delivered). OR service functionality was never delivered and RIv2 abandoned to be substituted by RIv3 (also called RI ORv3).
RI 1.0 and RI 1.1	RIv3 webservice	RI second webservice version was officially announced in Month 30 and implemented by Month 34. The version was based on the major re-design of the RI architecture and the major re-engineering undertaken in OR. The client side application programming interface (RI-API) was in C++ with a full set of RI interaction functions, providing full functional to the RI.

Major activities performed and results obtained during the third year are:

- RIv3 delivered Month 35
- Ingestion Tool delivered Month 36
- Long term preservation manager tool integrated with RIv3 stand alone, delivered Month 36
- Watermarking integrated within MeshLab , V2.0 delivered on Month 30 and V2.1 delivered on Month 36 (see also Deliverable D5.3, Third Year report on WP5)

Although a major redesign of the system was decided in July 2011 (M29) and it required six months of implementation, the overall outcome of WP3 is on time, with no major deviation from the work plan drafted in the project Description of Work (DoW). However, the redesign and the delay of the RIv3 release influenced the integration work of the client tools such as the Long term preservation manager, the Ingestion Tool, and tools of WP6 "Creating the 3D Collection Item" and WP7 "Searching and Browsing 3D Collections". All planned partners are contributing to WP3 while some partners exceeded their planned contributions.

The activities will continue in Year 4 according to the plan described in the project contract.

The overall organization of the document is as follows. Section 2 gives a brief presentation of the objectives of WP3 for the reporting period. Sections 3-11 present in detail the work done in each of the nine Tasks of WP3 during Years 3 and the results obtained. In particular, in Section 4, in addition to the presentation of the work of task *T3.2 Design architecture*, we also present the overall integration of the RI components. In Section 5, as part of the *T3.3 Design and implementation of a metadata repository* we include the *Design and implementation of the Ingestion Tool* which was not part of the initial DoW. In Section 6, as part of the *T3.4 Design and implementation of an object repository*, we include the description of the *MetadataGenerator* which was not part of the initial DoW. Section 12 reports on the publications produced so far. Finally, the RI API Specification is presented in detail in Appendix A, Ingestion Tool Design in Appendix B, the Fundamental categories and relations in Appendix C and the Integrated Viewer / Browser (IVB) testing at the Victoria and Albert Museum (VAM) in Appendix D.

2 Introduction and Objectives

In this report, we present the achievements made in WP3 "Repository Infrastructure" during the third year of 3D-COFORM. These mainly consist of a major redesign of the RI, implementation and deployment of RIv3, the design and implementation of the Ingestion Tool, the implementation of the Long term preservation manager and the integration of the Watermarking into MeshLab.

We present the work that was planned for the third year of the 3D-COFORM project and the work that was performed during this third year. The presentation is done per task, following the structure of the original Description of Work (DoW). In the task *T3.2 Design architecture* we also include the presentation of the overall integration steps of the RI components. In the task *T3.3 Design and implementation of a metadata repository*, we include the *Design and implementation of the Ingestion Tool* which was not part of the initial DoW. In the task *T3.4 Design and implementation of an object*

repository, we include the description of the *MetadataGenerator* which was not part of the initial DoW. For each task, a section is devoted to the analysis of the deviations between the work performed and the work planned, and an additional section presents the plans for the fourth year of the project.

It must be emphasized that the work in WP3 is a collaborative effort of all the involved partners.

3 T3.1 Scientific and Technical Coordination

The objective of this task is to coordinate and monitor the progress in the development of tools, including identification of deviations and amendments to the work plan. It also ensures the interrelation between work packages.

3.1 Work planned

The coordination and monitoring of the project technical activities continues through the whole duration of the project.

This task continues to support partners toward the objectives of the project. In order to achieve this, guidelines, points for synchronization and clear goals would be defined and provided to the technical partners. The progress of the project would be monitored and it would compare with the needs of the project, in order to adapt and achieve integration at different levels (from a technical and business point of view). In addition, a close cooperation with the business part of the project would be promoted, in order to test and train Cultural Heritage (CH) professionals on the tools developed within the project, aiming to collect feedback, which can be integrated within the development in improving the tools in view of usability, robustness, maturity, benchmarking, interoperability and scalability. This cooperation would be in line with the demonstrations for the end of the project.

The overall objective for the third period of the project was to mature the development of the tools and to increase the interaction with the CH users, by means of testing and training activities.

3.2 Work performed

In the third year of the project, the technical development was focusing on maturing the tools developed during the previous period, in order to achieve usability, robustness and scalability. This was pursued by involving more tools within the testing and training plans, by performing individual testing exercises, and also by implementing stories, involving practitioners and technicians within real workflows and with real data. These stories were:

1. Ivory Panel (VAM, KUL, Fraunhofer)
2. Boat (CyI, FORTH)
3. Tekke (CyI, PIN, UoB)
4. Vases (CNRS, C2RMF, KUL)
5. Mimba (VAM, CULTNAT, CNR-ISTI)
6. Mosque (VAM, CULTNAT, CNR-ISTI)

The orchestration of these stories generated a new level of integration and understanding on both sides - on the side of practitioners and the side of the technicians. This enabled new scenarios, where the practitioners had the opportunity to deeply comprehend the capabilities of the tools and on the other

hand the technicians could work with real cases and workflows, providing valuable insights for further improving the tools.

The experiences accumulated by the different activities in the third period and the further development of the tools have revealed challenges, which required the project to proactively react. In this context, internal restructuring of the responsibilities for some tools has been carried out, in order to answer to the identified challenges. Some of these restructuring actions were:

- Repository Infrastructure (FORTH, TU Graz): TU Graz took over the development of the RI interfaces from FORTH, in order to facilitate the communication with the new version of the Object Repository (OR) and the release of a consistent solution. More details can be found in the report of Task 3.2 Design Architecture.
- Integrated Viewer / Browser (CNR-ISTI, Fraunhofer): CNR-ISTI became responsible for the searching and browsing components, while Fraunhofer focused on the annotating and viewing components of the IVB. This action allowed for a clean development with fewer interdependencies without affecting the final goal. A detailed description of this process is reported in Task 7.1 - 3D Model Viewer/Browser and Task 7.2 - Semantic, shape & material retrieval (geometry / metadata search) (see also deliverable D7.3c, Third Year Report on WP7 Searching and Browsing 3D Collections).

The third period of the project was also monitored and controlled with general instruments for synchronization and reporting, as well as with dedicated activities for the tools with high level of integration. The following activities took place in the third period:

I. Pre-Review Meeting in Leuven, January 19th-20th, 2011.
II. Rehearsal Meeting on Cyprus, February 22nd-23rd, 2011.
III. Metadata Meeting and RI-IVB Meeting in Darmstadt, April 6th-7th, 2011.
IV. Post-Review Meeting, 3D-COFORM Workshop, Sector Advisory Board Meeting in Florence, May 3rd-5th, 2011.
V. RI Meeting in Darmstadt, July 26th-27th, 2011.
VI. Integration Review and 3D-COFORM State of the Art Workshop in Prato, October 17th-18th, 2011.

Additionally, several VOIP conferences have been scheduled during this period, easing the coordination and planning of the technical activities of the project without requiring physical meetings and the associated travelling time.

3.3 Deviation from work plan

No deviation from the work plan. The third year work plan has been fulfilled according to the DoW.

3.4 Plans for the next period

The technical development is planned to last until Month 42. Afterwards the remaining technical resources will exclusively be dedicated to testing, training, deployment experiments and the final

demonstration. The technical effort for the fourth and final year of the project will be focused on serving the practitioners in their real environment and within the established workflows. Thus the development will address these requirements rather than new risky and unstable features. Hence, an analysis for the consolidation of the tools will be performed, in order to identify the more matured and promising features, which will be brought forward until the end of the project. This will be realized by means of re directing some of the technical resources. Consolidation is the main objective of the final period and this will be strengthened by directly collaborating with practitioners and by proactively supporting the enterprise activities.

4 Integration of Repository Infrastructure Components & T3.2 Design architecture

In addition to the description of the task T3.2 "Design Architecture", in this section we also describe the integration workload needed for the several Repository Infrastructure (RI) components

4.1 Work planned

The work planned for WP3 during the third year was to provide the first version of the integrated RI.
Month 36 - Third Year Deliverable

- RI v1.1 Revised version. Integration of MR v1.1, OR v1.1, QM v1.1 and CRM services.

4.2 Work performed

The RI is composed of several modules which connect and interact with other as illustrated in Figure 1.

During the third year some of the decisions taken had to be revised and accordingly the design of the RI architecture was revised. A major re-design step started in Month 29 and was officially accepted as the new RI architecture design in Month 32, for reasons explained in Section 4.3. While the effort up until Month 32 was focused on the RI v2 webservice release, due to this re-design a considerable amount of the RI webservices code had to be re-implemented.

In short, the OR implementation was totally re-implemented and it was decided that the OR webservice ø²dzf Ř l· Ő ű l·wèbsérvícés in the Whole architecture, while the QM webservice and the MR webservice would merge into one webservice (QMMR webservice) that would only deal with metadata handling and lj dz Ŝ NJ ƛ Ŝ ă l· y˝ Ř ø²dzf Řwebservícés Ŝ ű ²ƛ ýwébSérvíce w l· á ă f l· Ø Ŝ

A major redesign on the RI API has also taken place (see Appendix A: ά ẇ! Lt L ă LJ Ŝ Ő ƛ Ŧ ƛ Ő l· ű ƛ ²y˝ é u totally new set of functions for interacting with the RI. The new API provides the functionality that was missing from the older RI API: user management, multiple location handling, access rights, replicas management, use of groups of objects, transactions, etc. The major development platform for the API was decided to be in C++, compromising an on-time RI API development in Java.

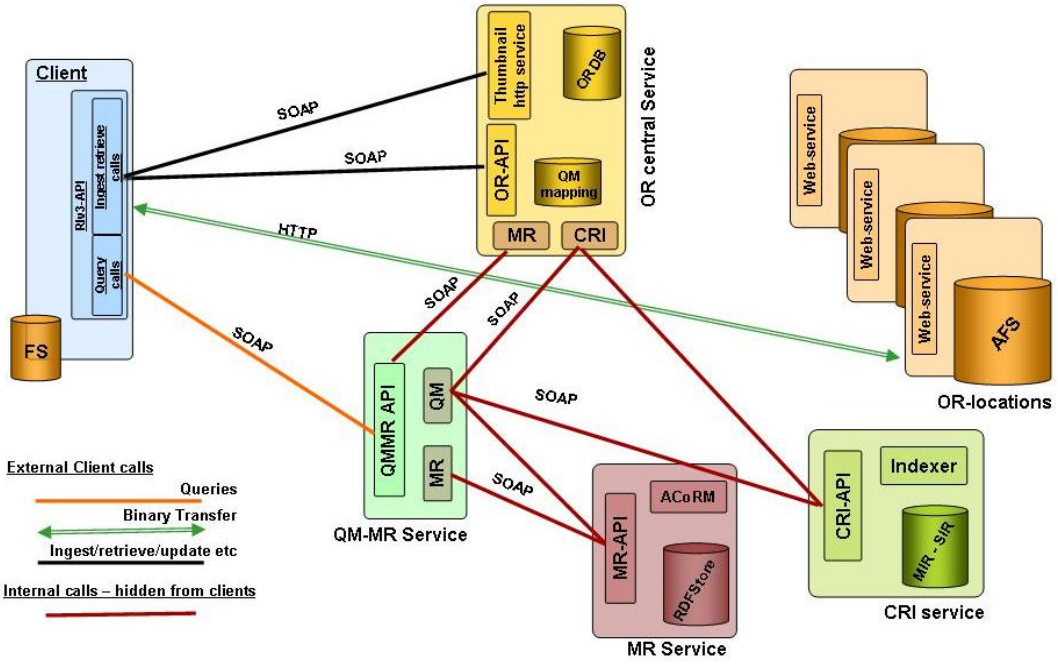


Figure 1: The new RI architecture. The OR central webservice is the only webservice in the whole architecture, while the QM functionality and MR functionality are merged into the QMMR webservice. The OR webservice and OR locations are separated from the OR central webservice

4.3 Deviation from work plan

In Year 3 there was a major deviation from the work plan in T3.2 "Design Architecture", T3.3 "Design and Implementation of a Metadata Repository" and T3.4 "Design and Implementation of an Object Repository". The reasons for the deviation are explained below. The main negative effect is that the RI is now completed almost a year later than it was planned. The main positive effect is that the RI as the main central infrastructure of the 3D-COFORM project has greatly improved in quality.

The RI is a complex piece of software. The requirements analysis in Year 1 had revealed a comprehensive set of functionalities that the system should deliver. Since there were various different options for the implementation of different system components, and for their interplay, the decision was taken to pursue an evolutionary approach. According to the principles of agile software development we started the implementation phase with a small and monolithic but well defined infrastructure core delivering only the essential functionality (RI alpha), which could already be demonstrated at the Year 1 review (Month 15).

It then became apparent that the original plan from the DoW could not be realized, which was to use existing repository software. The big problem was the feature mismatch, since each software we evaluated had features that we did not need and also lacked features that we did need. Finally, the most promising candidate, the Globus Toolkit (GT), also had to be ruled out because of the overly high complexity of setting up and running this grid computing framework. This was already one indication that the RI implementation task was definitely more complex than we had estimated when writing the DoW.

So the decision was taken to further pursue the so far successful evolutionary approach, which had taken us to the Year 1 prototype. We continued to add missing functionality in an iterative way, and at the same time split up the monolithic library into three webservices: One RI webservice as a front, and behind it the MR webservice and the OR webservice. It seemed a very good idea to have the RI as single point of contact for the client computers, offering a coherent interface and hiding the complexity of the services behind it. Another important reason was that it gave the MR/OR developers the freedom [illegible] worked successfully to a certain point, and at the Year 2 review (Month 25) we could demonstrate a working system based on web services.

The one thing, however, that was lost due to the evolutionary development was the global stringency of the system, and any elegance and simplicity. Due to time pressure (all partners were waiting to perform the RI integration in Year 3) there had not been enough time for thoughtful source code refactoring. This led to a certain administrative overhead, for example for the session management; both the MR and the RI need to keep track of the current users and the validity of their session tickets. Finally, the point came when adding further extensions became unmanageable. Heavy horizontal changes going through the whole infrastructure, in particular the management of user permissions, the generalization from a single OR node to multiple ones, and the replica management on the distributed storage, could simply not be carried out any more. This was the reason why the RIv2 was never delivered.

At the same time another serious issue came up, which was that using the RI webservice was not as easy as we had considered. The plan was that a client [illegible] webservice, using automatic code generation tools (e.g. gSoap for WSDL to C++, Eclipse for WSDL to Java) to call the RI functions from within C++ or Java programs. This worked very well for Java, and for a small-size RI. However, with about 16 data struct types and about 100 functions that need to be offered by the RI, the administrative overhead only for calling functions was growing disproportionally.

The project was in danger of having a central infrastructure with serious problems.

In this situation, mid April 2011 (Month 29), six weeks after the review, and under extreme time pressure from the partners, we took the somewhat bold decision to start as contingency measure a complete rewrite from scratch of the whole RI. Notably, this started as additional activity, using additional staff, since in parallel the RIv2 developers still tried to push it towards completion (until July 2011, Month 32); but tracing errors became ever more difficult due to the complexity of the distributed system.

Rewriting from scratch turned out to be a most beneficial decision, since we managed to finish the complete system in only six months. Since October 2011 (Month 35) it is rolled out; partners are installing their own ORLocations (storage nodes), Java and C++ developers have a stable API, datasets and metadata are being ingested, and the development of the RI system is now in maintenance/optimization mode.

In retrospect, this success was not as surprising as it seems. At first, 25 months of iterative development versus 6 months for rewriting from scratch seems as if the first two years had been a big waste. However, in fact it is just to the contrary:

- Without the experience from Month 1 - Month 25 it would have never been possible to build a clean system
- Through the intense work on RIv2 we knew exactly which were the difficult issues and problems
- We knew how *not* to solve them i.e. which solutions turned out to be disadvantageous
- We finally obtained a very clear picture of the full functionality of the whole system
- b 2 ø ø Š] y Š ø l · f t ă è ă ű Š Y Ő 2 YLJ 2 y Š y ű ă must support Ő ű Ќ Š ά Ќ 2 NJ
- ² Š Ќ l · Ř LJ dz ō f ƛ ă Ќ Š Ř ű ø 2 LJ l · LJ Š NJ ă 2 y ű Ќ Š ά ƛ Ř ă Š l · f w L ă had even received a best paper award at VAST 2010
- And last, but not least, partner TU Graz could recruit an experienced programmer of a distributed digital library system (www.probado.de), who proposed a development framework that greatly facilitates the testing, profiling, and debugging of webservice-based environments.

The resulting clean RI system is described in T3.3 and T3.4.

In summary, despite the six months delay in providing the partners with the RI, reactions were positive. Several partners expressed that they prefer a clean, complete, stable system that comes now almost a year later, over underachieving the project objectives and a multiplication of debugging efforts due to an overly complex central system that has still reduced functionality. RIv3 became available in Month 35 and integration is proceeding, covering the deviation from the original work plan.

4.4 Plans for the next period

The RI is in maintenance mode. The only functionality that still needs to be added is an advanced support for CRI (component retrieval indices), for which already a precise plan exists.

5 T3.3 Design and implementation of a metadata repository

5.1 Work planned

The major plan for the third year was to transfer the existing SwiftOWLIM reasoner implementation to BigOWLIM reasoner and the design and implementation of an ingestion tool to support the ingestion of the data of acquisition and processing simple procedures.

5.2 Work performed

5.2.1 BigOWLIM reasoner

In the third year almost all of our plans were achieved. We transferred our reasoning implementation from SwiftOWLIM to BigOWLIM and we worked mainly on the new required configuration. We set up new operation parameters for the internal reasoning rule engine in order to have controlled inferencing over the appropriate schemas and ontologies. Also, we extended the existing Sesame client (called openrdf-workbench) in order to support the BigOWLIM reasoner repository types and to be able to work with them in visual environment and thus save a lot of time rather than working by authoring.

During the first three months of the third year we worked on the preparation of the metadata for the six stories that were defined, in order to test the integration and demonstrate during the project review, using real data and answering to real needs in the Cultural Heritage domain. During this period support with guidelines and examples was provided from the MR development team to the partners about how to successfully construct and ingest metadata documents as well as how to query the MR with SPARQL.

. dz∃ Ŧ ƛ È ƛ y˝ ∃ Σ Y˝ l· ƛ y˝ f è n ingest and update functionality was done for MR in RIv2 stand-alone release. Also, in the period after the review, code reimplementation in specific parts improved performance and reliability.

Annotations ingest/update/delete functionality was implemented. This type of metadata has specific differences from the others because of the use of the concept of named graphs in the semantic network.

Finally, documentation has been created about installation and administration for MR and OWLIM reasoner.

5.2.2 RIv2 webservice and porting to RIv3

A major issue during the third year was the abandonment of RIv2 webservice release and the porting to RIv3 release. While the effort up until Month 32 was focused on the RIv2 webservice release, due to the

re-design described earlier, a lot of the code had to be re-implemented and the RIv2 interface implementation was abandoned. The MR service and the QM service merged into one service (QMMR service) that deals with metadata and query handling.

Work on RIv2 webservice

Porting from RIv2 stand-alone operation to RIv2 webservice operation caused changes to function signatures in order to support the new types of input and output parameters.

Create/Delete/Update Area functionality was finalized for RIv2 webservice release.

Although there were problems in RIv2 webservice release we attempted to complete a working system for the clients to work with, by directing effort mainly to finalizing the ORv2 module. This release was abandoned and never published, because of several technical difficulties as well as the fact that we should additionally have to spent time working for the migration of data to the RIv3 release which was till then under construction.

Work on porting to RIv3:

Additionally in the new RIv3 release, the creation of the new QMMR service which encapsulates the MR and the QM modules, led us to work on:

- Definition of new types of input and output parameters for functions
- Error handling and error codes adaptation to the new RIv3 requirements
- New logging functionality and log level definition per message case
- Tests for reliability and performance

5.2.3 Design and Implementation of a Simple Ingestion Tool

The ingestion tool provides an automatic way to create the metadata files of a simple acquisition process (single-step, e.g. Mindlome like process), using appropriate forms to assign metadata information on the acquisition event, identification information of the depicted object, and general acquisition set-up and capturing device(s) information. It also provides an automatic way to create the metadata files of a simple data-processing procedure, using the appropriate forms to assign metadata information on the processing event, for describing S/W and -S/W versions, along with the parameters that are passed to the specific S/W.

The tool supports the ingestion of metadata information of intermediate processing steps that users decide to store in the RI, while their intermediate data files are lost or erased. It also provides unified URI policy definition and full referential integrity control mechanisms: all forms interact with the RI, retrieve information and display it to the user to choose from. Data entry is supported with auto-suggestion editing capabilities based on auto querying specific categories such as things, persons, places, etc. All forms may be saved and re-loaded for further user editing (templates, or temporary files).

The design of the ingestion tool was delivered in April 2011, Month 29, as a document (see APPENDIX Ingestion Tool Design) and as a semi-interactive presentation that presented a mock-up tool, for the partners to have a more visual understanding of the tool look and feel.

The first implementation of the ingestion tool has been developed based on the RIv2 standard stable release, and was ready (with no delay) in August 2011, Month 33, but never delivered since RIv2 was abandoned. The final implementation included porting to RIv3 and was delivered in November 2011, Month 36 (with three months of delay due to the delayed availability of the Java API for RIv3).

It is available as a client-side Java tool (no installation required) and as a Java library: a good practice library of software components for metadata creation that 3D-COFORM tool providers can employ to integrate their tools more consistently and efficiently. A C++ wrapper (to the Java library) will be available in Month 37, for users that need a C++ library.

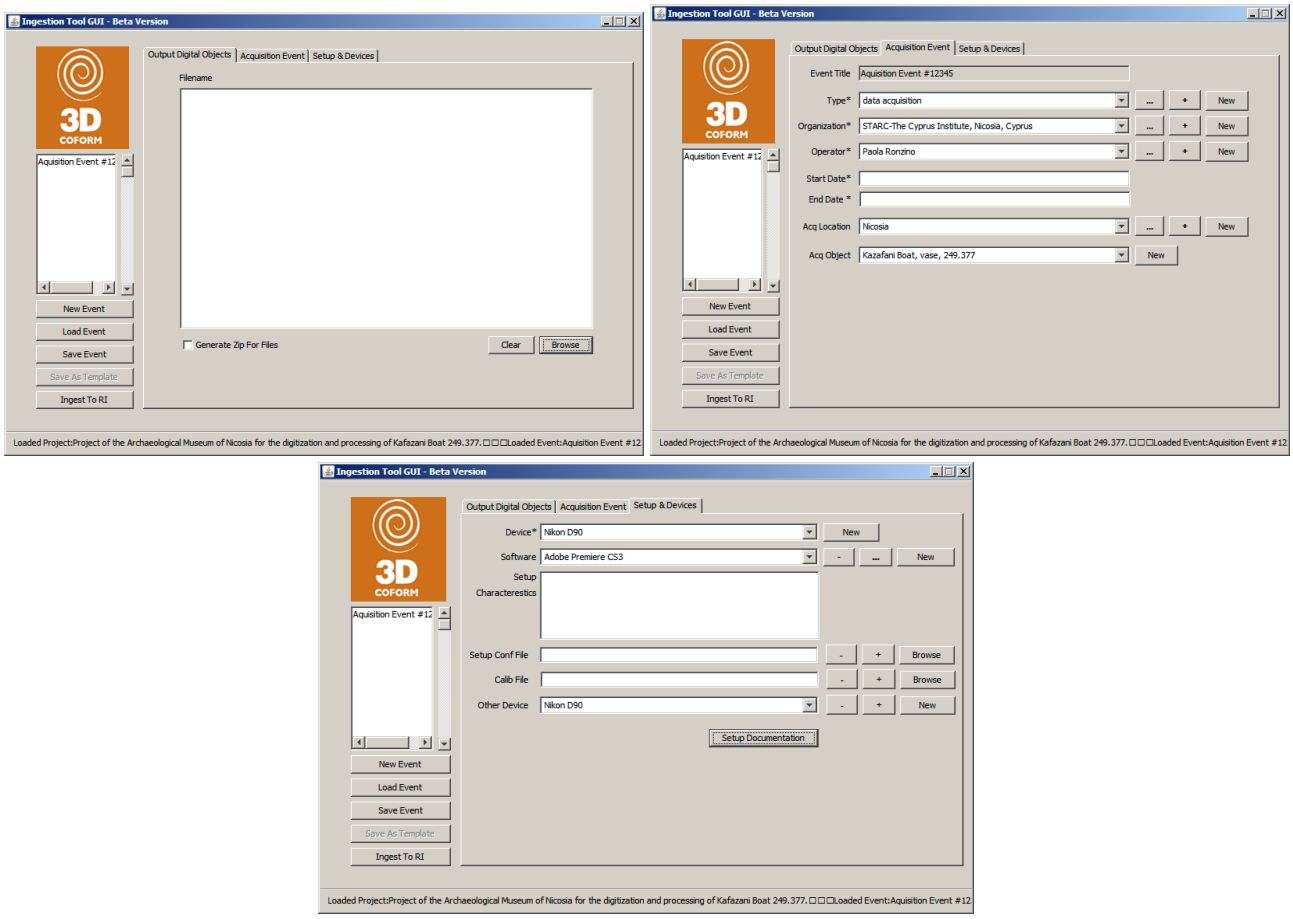


Figure 2: Ingestion tool forms for Acquisition Event

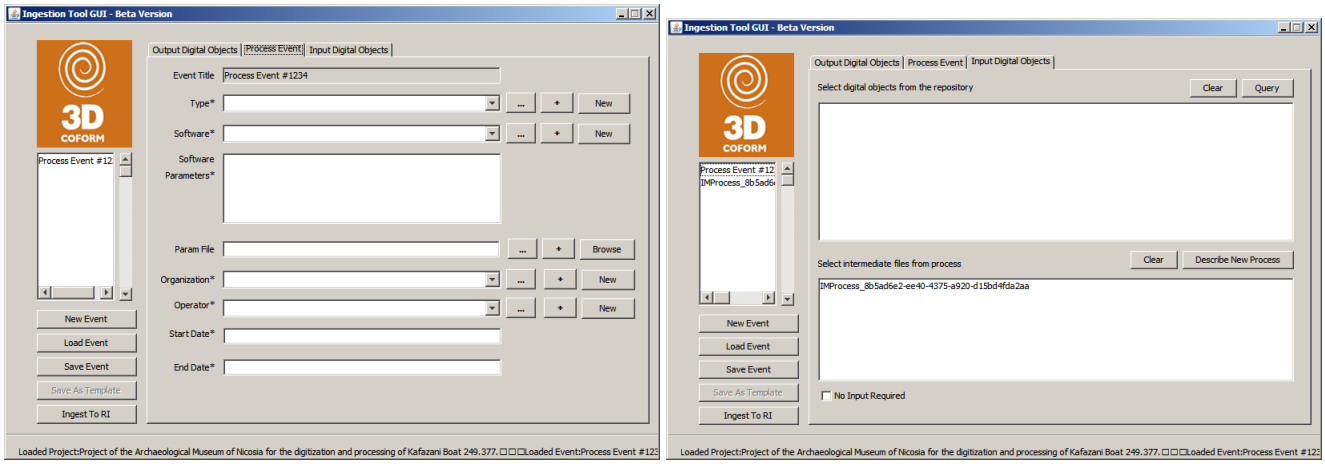


Figure3: Ingestion tool forms for Process Event

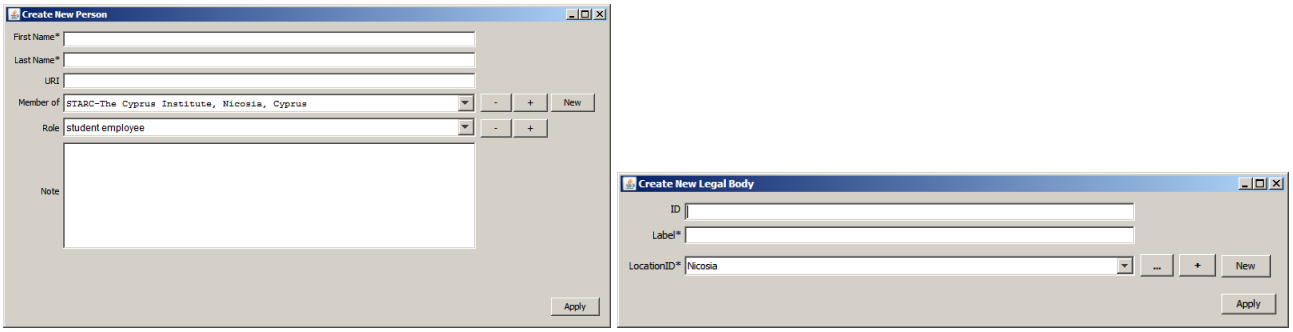


Figure4: Ingestion tool forms for creating Entities

5.3 Deviation from work plan

There was no deviation from the work plan regarding the Metadata Repository. There was a three month delay of the delivery of the Ingestion tool due to the delay of the Java API.

5.4 Plans for the next period

We plan to test and improve both the MR functionality and performance, making more efficient use of the advanced features of the BigOWLIM reasoner such as the full-text searching capabilities and query results scoring.

Additionally, we will work on the consistency check of the semantic data. Consistency check functionality which comprises the definition of consistency rules and the consistency check strategy and implementation, ensures the reliability of the semantic network.

Regarding the Ingestion Tool, we plan to implement a C++ wrapper (to the Ingestion tool Java library) that will be available in Month 37, and an update mechanism by month 38.

6 T3.4 Design and implementation of an object repository

6.1 Work planned

The major plan of this task of the third year was to implement the OR service and OR node services for data transmission.

6.2 Work performed

The Object Repository (OR) was completely rewritten from scratch in v3. The reasons for this were the problems with the evolutionary design of RIv2 explained in Section 4.3 (T3.2, Deviation from work plan). Nevertheless, through RIv2 a very good understanding of the issues and solutions was achieved, which allowed completion of the new implementation within six months.

¢ K Ŝ Ř Ŝ ă ƛ ∃ y˝ ƨ Ŧ ű K Ŝ y˝ Ŝ ǿ w L Ø o ă ű l· NJ ű Ŝ Ř Ŝ Ø Ŝ persons creating software tools in the 3D-COFORM projects. Since almost all tools in 3D-COFORM need to communicate with the RI, we chose in RIv3 an *API-centric design* to facilitate as much as possible the application development. Therefore this interface is not a web service, as in RIv2, but a client side library. This requires more effort for us since we have to create, in addition to the RI webservice, both a C++ and a Java client-side API, two respective software libraries (coform.lib and coform-ri.jar), and we need to keep them synchronised. The benefit was that we needed only two webservices, one OR webservice (running at TU Graz) and one QMMR webservice (running at FORTH). There is no more need for a third webservice acting as a facade, and the internal design of QMMR and the OR is much more decoupled.

Two documents describe the architecture of the new ORv3. The VAST publication [PBH*10] describes the design decisions and concepts behind the ORv3, including an early version of the database schema. The RI API Specification (Appendix A) describes the 14 data struct types and the 105 functions of the RI.

6.2.1 Overview of the client-side RI API

To facilitate tool development, the complete RI functionality is described in the RI API Specification (see Appendix A), a very concise 17-page document that contains all information that a developer needs. The API contains 13 sections each containing a small set of functions (typically less than 10). The RI API Spec defines exactly and concisely the behaviour of all the functions. The sections are organized as follows:

0. Datatype definitions: UniqueID, Dataset, Filestruct, Location, Group, User, Usergroup, etc. All in all 14 struct types that are used for the communication between tool and RI API.

1. Infrastructure: Functions to set the URLs of the services, to set start and count for list queries, error checking and messages, and XML description of groups, datasets, filestructs.
2. Session, Login, Location: At login, a user has to specify a location that will store their uploads. The result is a session ticket which is needed for any further communication with the RI.
3. Transaction, Undo, Rollback: Any editing operation (that changes the DB) needs to be enclosed in l· ű NJ l· y˝ ă l· Ő ű ƛ ƨ y˝ ó ά f ƨ y˝ Ǝ ű NJ l· y˝ ă l· Ő ű ƛ ƨ y˝ έ ú Φ ¢ NJ l· y˝ ă l· Ő ű ƛ
4. Ingest, Upload, Update: In tuple notation, Dataset = (Binary, Metadata, Thumbnail, Areas). The Metadata are mandatory (and can be updated), the Binary may be uploaded (at most) once.
5. Retrieval and Download: Retrieval means to fill structs from UUIDs (Dataset struct from given dataset UUID etc.) Download can also be done by streaming without obtaining a download URL.
6. Thumbnails: Every dataset can be provided with a thumbnail, a small characteristic image that will allow users faster browsing through large result sets, scanning for the result they need.
7. Areas: Every dataset can have attached a set of areas. Each area defines a region or a part of a 3D model that can henceforth be referenced and denoted, similar to a link anchor in hypertext.
8. Replica management: Binaries are stored in so-called Locations that belong to an institution or a company. Binaries can be replicated to other Locations for backup or workflow reasons.
9. Group management: Groups are basically directories containing datasets. They are structured hierarchically, and every dataset (and every group) can belong to any number of groups.
10. User management: Users are structured in Usergroups that facilitate the management of permissions. The root Usergroup has the weakest permission. The Usergroups form a tree.
11. Permissions are defined by (subject, object, permission) triplets where the subjects are users or usergroups, the objects are datasets, groups, locations etc., and there are 12 permission types.
12. Mimetypes and Software: Every dataset (every file) has a mimetype from an extensible list of mimetypes. Software can import and export different mimetypes, important for format rollover.
13. SPARQL Queries: These are rich questions that are sent to the QMMR to query the semantic network. The result is typically a set of dataset UUIDs that can then be retrieved by the user.

The client-side library (coform-ri.lib or coform-ri.java) is in fact only a thin layer that directly communicates with the ORCentral webservice (section 12.2) or the QMMR webservice (section 13) in a synchronous fashion.

6.2.2 Overview of the OR components and their implementation

The OR comprises a number of different software tools that have been created in M29 to 36. They are available to all partners via the central FusionForge development repository of 3D-COFORM. It contains an SVN with the source code, code releases as zip, documentation as pdf (specs and manuals), and a wiki.

The ORCentral is the central OR webservice at http://orcentral.cgv.tugraz.at. It is written in C# and uses a MicrosoftSQL database, the ORDB. The ORCentral realizes the actual functionality and offers via SOAP about 70 functions to the clientside API. It has its own extensive set of serverside unit tests that is continuously extended whenever code parts are frozen. The goal is to reach 100% coverage of the tests (see Figure 6). A second instance of the webservice, ORCentralDebug, is used for testing.

The Location VM is a preconfigured virtual machine (~20 GB of files) that can be set up inside an institution. It can be run using the VMWare player on a computer with a lot of hard disk space. These hard disks receive the binaries of datasets that are uploaded by users who are logged in at this location. This way, the precious binary data remain within the institution that legally owns them. The Location VM contains a Windows 7 and a Location webservice written in C#. The firewall must be configured such that the Location VM can communicate with the ORCentral, and with clients having a download ticket.

COyOTE (COform Object repositoTory Explorer) is the debug and administration tool for ORCentral (see Figure 5). It is a clientside application written in C# showing a treeview of the group structure. All new ORCentral features are tested through COyOTE. The frequent updates are dealt with using an auto update function. COyOTE can be used to browse the ORCentral group structure. However, the tool is not primarily targeted at end users as it shows perhaps too much internal information.

RI-Power is a set of commandline tools for the Microsoft Powershell. It is basically a GUI-less version of COyOTE that allows RI access via shell scripts. This is very useful for automating certain tasks, for example, when ingesting a whole directory tree, or to check whether metadata versions have changed on the server.

RI C++ Client Lib provides the API as a single C++ header and a simple static library provided to the tool developers as C++ source code because it also serves as the reference implementation for client-side libraries; the purpose is that other clientside libraries (e.g. Java, and perhaps PHP, Python etc in the future) can implement the same set of functions with identical behaviour. It contains not only the actual library, but also a set of test examples that show how to use the API functions in practice.

CORExplorer is a simple Qt application that shows the RI group structure in a tree view. It allows upload and download of files, and it shows thumbnails. The CORExplorer is also provided as source code, and it is part of the test examples of the RI C++ Client Lib. It is intended as an extensible source code tutorial.

RI Java Client Lib provides exactly the same functionality as the RI C++ Client Lib. As Java is a bit different from C++, the functions have a slightly different signature. However, they behave exactly identical to their C++ counterparts, so that the RI API Specification equally holds for them too.

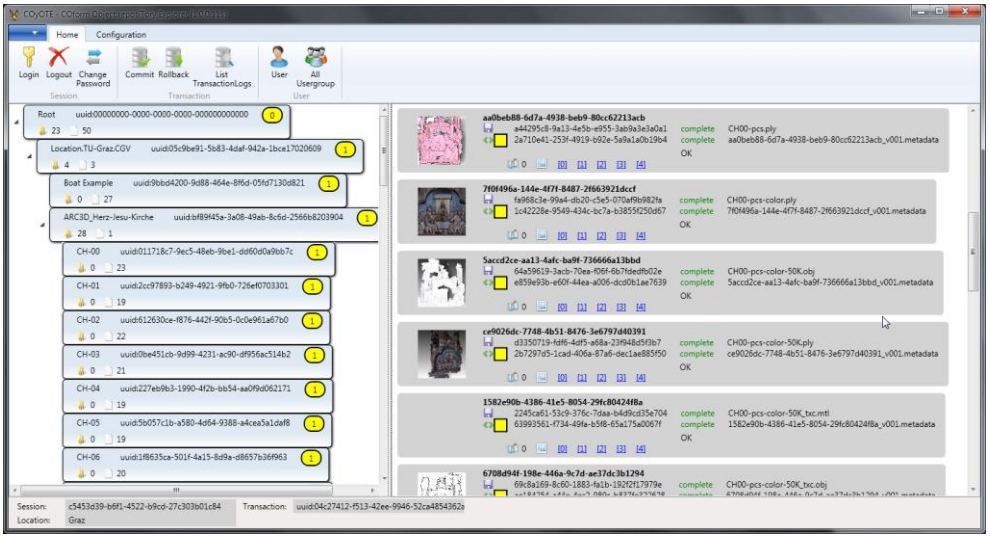
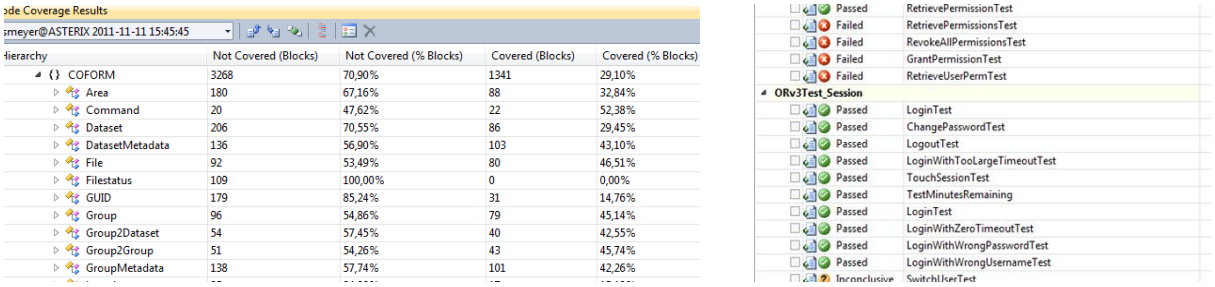


Figure 5: The COyOTE tree view client. It shows much information, such as UUID of the Dataset, Files and Metadata, allows displaying the replica locations and to trigger replica creation, and shows open transactions.



Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)	Covered (% Blocks)
COFORM	3268	70,90%	1341	29,10%
Area	180	67,16%	88	32,84%
Command	20	47,62%	22	52,38%
Dataset	206	70,55%	86	29,45%
DatasetMetadata	136	56,90%	103	43,10%
File	92	53,49%	80	46,51%
Filestatus	109	100,00%	0	0,00%
GUID	179	85,24%	31	14,76%
Group	96	54,86%	79	45,14%
Group2Dataset	54	57,45%	40	42,55%
Group2Group	51	54,26%	43	45,74%
GroupMetadata	138	57,74%	101	42,26%

Passed	RetrievePermissionTest
Failed	RetrievePermissionsTest
Failed	RevokeAllPermissionsTest
Failed	GrantPermissionTest
Failed	RetrieveUserPermTest
ORv3Test_Session	
Passed	LoginTest
Passed	ChangePasswordTest
Passed	LogoutTest
Passed	LoginWithTooLargeTimeoutTest
Passed	TouchSessionTest
Passed	TestMinutesRemaining
Passed	LoginTest
Passed	LoginWithZeroTimeoutTest
Passed	LoginWithWrongPasswordTest
Passed	LoginWithWrongUsernameTest

Figure 6: Systematic server-side testing using C# and VisualStudio. Left: The overall code coverage of test procedures is currently just about 30% because the code has grown quickly. Right: Each unit comprises positive and negative tests, which are run regularly after changes in order to spot problems

6.2.3 The MetadataGenerator

The MetadataGenerator is a tool for generating RDF metadata files in an offline setting, i.e. without the possibility to query a semantic network. It is targeted as a specific use case, which is that data from acquisition campaigns are arranged hierarchically in a dictionary structure.

The CIDOC-CRM semantic network allows the user to ask much richer questions, but it also requires much richer metadata. In order to prepare the data of an acquisition campaign for ingestion, somebody has to go through all the data files and provide the necessary markup. The MetadataGenerator takes advantage of the directory structure and fills, for example, drop-down boxes with values defined on higher directory levels. This speeds up the form filling process and helps to avoid errors.

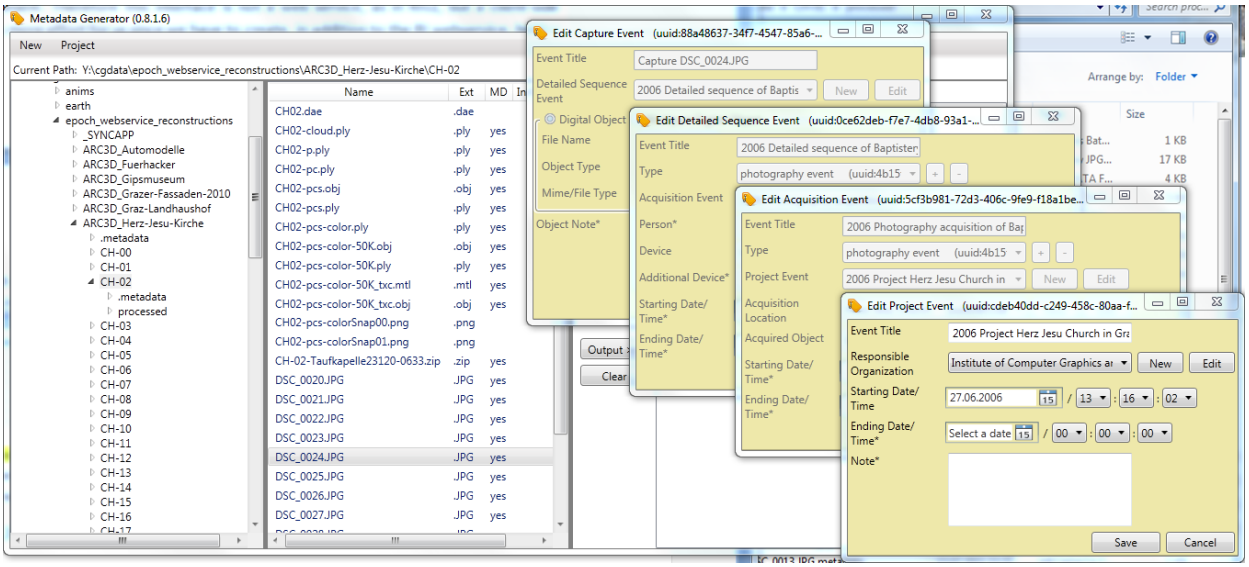


Figure 7: The MetadataGenerator. Left: Tree view of the historically grown directory structure. Right: The files contained in the respective directory. A JPG file is selected which is associated with a CaptureEvent that is part of a DetailedSequenceEvent that is part of an AcquisitionEvent that is part of a ProjectEvent.

The MetadataGenerator was initially created just to bootstrap the batch ingestion process at TU Graz, where initially no semantic network was available. However, it turned out that it has many advantages, and in fact, when during the process of metadata generation a systematic error is spotted, the operator can go back to the already ingested files and correct it, prior to ingestion. Furthermore, the form data are stored locally in an intermediate format in the ./processed subfolder of each directory, so using text based tools (e.g. automatic search and replace) can be performed for larger corrections. Finally, the RDF data are generated from the intermediate format using RDF templates. Since CIDOC CRM has certain flexibility with respect to the data modelling, the RDF templates are based to some extent on conventions. When these conventions change, the RDF files can be re-generated using an updated set of RDF templates, so that the changes are more consistent; remember that metadata can be updated.

TU Graz therefore provides the MetadataGenerator as additional tool for metadata generation. It complements the IngestionTool (which is for an online setting, another important use case). The MetadataGenerator is primarily targeted at large-scale metadata generation tasks. It has also already been used extensively for generating the metadata for at least three acquisition campaigns during ă dz Y Y Ŝ NJ н л м м Σ ǿ K ƛ Ŏ K Ŏ ƨ Y LJ NJ ƛ ă Ŝ l · ō ƨ dz ű м н л л Ř l · ű l · ă Ŝ ű ă Φ three school girls who did a 8-week internship at CGV TU Graz. Their feedback resulted in important usability improvements, since the tool was developed in parallel to the internship students already using it.

The limitation of the MetadataGenerator is, of course, that in order to avoid referencing problems, when referring to things that are l · f NJ Ŝ l · Ř è ƛ y ű K Ŝ a w Σ ű K Ŝ ities Ŏ ƨ NJ Ŝ Ŏ ú á ∃ f ¦ ¦ L 5 ă K l · Ø Ŝ ű ƨ ō Ŝ dz ă Ŝ Ř Φ ¢ K Ŝ Y l · ƛ y á ō l · Ŏ ļ ∃ NJ ƨ dz y Ř Y Ŝ

provides, for example, UUIDs of cameras, people, institutions, etc. needs to be checked more or less manually prior to generating metadata. We envisage continuing the development of the MetadataGenerator until we have reached a sufficient level of efficiency. We consider this very important since the metadata creation is one of the undeniable cost hurdles when using 3D-COFORM technology in a museum.

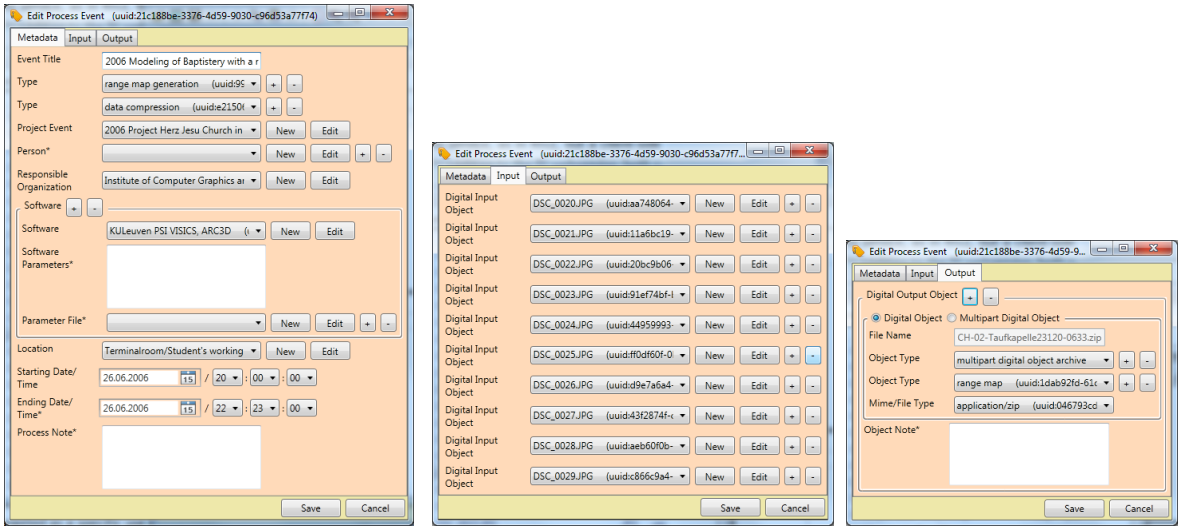


Figure 8: A process event. The input of the A3D process is a set of images which can be selected directly in the treeview of the MetadataGenerator. It creates a multipart digital object, a zipfile with a range image per image.

6.3 Deviation from work plan

With respect to the DoW, the implementation is behind because Milestone 3.3 (Month 20) was supposed to see the OR finished. This is delayed now to M36, for the reasons explained. However, with respect to the work planned (see Section 6.1) we are on time, since we have met the objectives stated:

- The OR/MR integration is finished. The bilateral communication between them is implemented.
- At metadata upload or update, a copy of the metadata is stored in the ORDB, and then sent to the MR webservice for (i) consistency checking and (ii) ingestion into the semantic network.
- The metadata ingestion/update process is fail-safe: data loss or inconsistencies are detected.
- The ORDB can be queried only through the RI, there are no direct SQL queries to the ORDB. Instead, a SPARQL-SQL query mapper in MR, the D2R server, isolates ORDB-relevant queries and sends them to the ORDB as SQL, collects the answers, and sends them to the client (API section 13).
    - client → QM MR webservice → D2R → ORDB → D2R → QM MR webservice → client

- For speed and consistency reasons, the metadata are held directly in the ORCentral, in fact in the ORDB, so that the locations really only need to store the large volume binary data.
- There is not only one or two but multiple locations being set up right now, as all partners need to get their data into the OR, and the developers need locations for testing and benchmarking.
- The user management and permission system is also implemented.

6.4 Plans for the next period

The OR is completed and will not change anymore. It is not yet at the level of a commercial product, but it is reasonably stable and sufficiently robust that it can serve as part of the central infrastructure of the project.

The only larger addition yet to be carried out is the integration of content retrieval indices (CBIR). Here ø Ŝ ø l· y˝ ű ű ƨ ǻ Y LJ f Ŝ Y Ŝ y˝ ű l· Ŏ f ƨ dz Ř l· LJLJNJ ƨ l· Ŏ K ō è ă Ŝ y˝ Ř ǻ y˝ location VMs. It is executed on all locally stored binaries, producing feature vectors that are then sent back to the indexing databases (e.g. shape search in Leuven and material search in Bonn). This is the only way to avoid the binaries to leave the institutions; only the derived feature vectors do. Furthermore, the indexing happens in a massively parallel fashion when all Location VMs are active.

Concerning the RI API Spec, we envisage only minor adjustments. The spec has not changed anymore since mid July 2011 (Month 32), which is an indication that it is now mature. We may add one or the other API function for convenience, but mainly focus on optimizations and bug hunting. We are pursuing l· á ō dz Ǝ Ŧ ǻ È ō Ŝ Ŏ ƨ Y Ŝ ă dz y˝ ǻ ű ű Ŝ ă ű ɛ l· LJLJNJ ƨ l· Ŏ K ø K Ŝ NJ Ŝ l· y˝ è respective unit test is written that tests exactly this case.

7 T3.5 Design and implementation of a query manager

7.1 Work planned

During the third year, it was planned to produce and integrate a web-based service performing the CRI services.

The Query manager would take advantage of the reasoner installed and running on the MR service.

7.2 Work performed

The Query manager (QM) has been migrated into the QMMR web service. Appropriate modifications on input/output parameters and functionality, for service operation, have taken place. Querying functions have been enriched with the missing SPARQL ASK query form, and two functions have been extended according to new user requirements, such as:

- querySelect() function now supports the JSON format in addition to the SPARQL query result format.
- queryRepository() function that is used to query OR, MR and CRI modules and was at first designed to return a ranked result list of objects <uuid, rank>, has changed to queryingWithCRI() function which supports the return of the full set of variables that participate [illegible]. The result is also available in JSON or SPARQL query result format.

D2R service, responsible to provide SPARQL query capabilities on the relational db of the OR, has been reconfigured for the new RIv3 db schema.

Finally, query fundamental categories of data have been defined in the semantic model and appropriate queries have been implemented in SPARQL to provide a powerful way to retrieve targeted data that are [illegible] Reasoning rules have been designed for this purpose to be integrated in QM.

7.3 Deviation from work plan

The major deviation from the work plan is the lack of integration with the CRI services, for reasons explained in Section 4.3. The plan for the integration is described in Section 6.4 and in the deliverable of WP7 (D7.3c, Third Year Report of WP7 "Searching and Browsing 3D Collections").

7.4 Plans for the next period

In the next period apart from integration, testing, debugging and optimizations in the QMMR service we will complete the communication with the CRI modules (see also the deliverable D7.3 where the CRI modules are described in detail). Thus, a new web service will be developed in order to perform the queries on the CRI service. The plan for the integration is described in Section 6.4.

8 T3.6 Design and implementation of an annotation and co-reference manager

8.1 Work planned

During the third year, it was planned to integrate the annotation and co-reference manager in the MR-service, the web-based service of the Metadata Repository.

8.2 Work performed

Following the redesign that we have already described in the previous section, the Annotation and Co-ReferenceManager (ACoRM) has been migrated into the QMMR web service. Appropriate modifications on input/output parameters and functionality, for service operation, have taken place.

8.3 Deviation from work plan

No deviation from the work plan. The third year work plan has been fulfilled according to the DoW.

8.4 Plans for the next period

The ACoRM will follow the integration, testing, debugging and optimization of the QMMR web-service.

9 T3.7 Implementation of a Long term Digital Preservation Management Tool

9.1 Work planned

The work plan for the third year was as follows:

- Continue to develop component for final release in month 30
- Undertake formal preservation planning process for 3D model data using the Plato Preservation Planning tool, absorbing results into preservation component knowledge base
- Undertake formal preservation capacity and risk evaluation (primarily associated with infrastructural aspects of preservation of 3D materials), absorbing results into knowledge base
- Evaluate 3D-COFORM repository infrastructure for preservation capacity as part of T3.2 Integration and Testing

9.2 Work performed

In the last period we have completed development of the long term preservation manager, which collates a series of individual applications aimed at ensuring the long term accessibility and comprehensibility of stored objects and metadata. The preservation manager is partially integrated with the repository infrastructure, although requires additional development to integrate with the revised repository specification (RIv3). Three principal (interconnected) applications are encapsulated within the preservation manager. These are as follows:

3D-COFORM Preservation Information Package Manager: This sub-component supports the conceptual formation of archival packages of object and metadata content and their export for offline archiving. The preservation package manager enables the creation of a package and its association with a parent where complex nested packages are required. Each is associated with a corresponding project event. The population of packages is performed by browsing or searching the RI and content can be accessed either by corresponding event or by reference to a range of metadata elements. The association of triples with the Archival Information Package (AIP- defined in ISO 14721:2003 (Reference Model for an Open Archival Information System)) is managed by the application and enables the conception of multidimensional packages which range in terms of chronology and across a variety of relationships. Functionality (not yet implemented) will enable the customisation of recursive content association where further relationships in the MR will be traversed and associate metadata triples added. A METS schema defines the format for offline export of packaged information.

Figure 9: 3D-COFORM Preservation Package Manager

3D-COFORM Preservation Risk Manager. This sub-component supports the formation of risk relationships and enables the association of instances to elements evident within defined preservation packages, or associated dependencies. The risk manager is a critical element in the identification of preservation issues, and is intended to enable prompt responses to be taken to safeguard information availability and accessibility. Risks are defined in terms of relationships between activities (events ...), resources (digital devices, software, physical objects ...), policies and duties (rights, responsibilities ...). A risk ontology illustrates standard preservation risk relationships (based on generic preservation objectives) although this is fully customisable to reflect specific individual circumstances.

Figure 10: 3D-COFORM Risk Ontology Browser