The 11th International Symposium on Virtual Reality, Archaeology and Cultural Heritage VAST (2010)
A. Artusi, M. Joly-Parvex, G. Lucet, A. Ribes, and D. Pitzalis (Editors)

A Distributed Object Repository for Cultural Heritage

X. Pan[1], P. Beckmann[1], S. Havemann[1], K. Tzompanaki[2], M. Doerr[2] & D. W. Fellner[1,3]

[1]Institute of Computer Graphics and Knowledge Visualization, Graz University of Technology, Austria
[2]Institute of Computer Science, FORTH-ICS, Crete, Greece
[3]TU Darmstadt & Fraunhofer IGD, Germany

Abstract

This paper describes the design and the implementation of a distributed object repository that offers cultural heritage experts and practitioners a working platform to access, use, share and modify digital content. The principle of collecting paradata to document each step in a potentially long sequence of processing steps implies a number of design decisions for the data repository, which are described and explained. Furthermore, we provide a description of the concise API our implementation. Our intention is to provide an easy-to-understand recipe that may be valuable also for other data repository implementations that incorporate and operationalize the more theoretical concepts of intellectual transparency, collecting paradata, and compatibility to semantic networks.

Categories and Subject Descriptors (according to ACM CCS): H.2.1 [Database Management]: Data models—Schema and subschema H.2.7 [Database Management]: Data warehouse and repository—Data models H.3.7 [Information Storage and Retrieval]: Digital Libraries—Dissemination

1. Introduction

The production of high-quality digital assets from cultural heritage artefacts requires a complex workflow and therefore faces a number of serious problems. The workflow typically starts by *3D acquisition*, which is a measurement process producing large-volume raw data. The raw data are subsequently refined and undergo a number of processing steps (cleaning, de-noising, hole filling, etc) until, at some point, a *digital master model* is produced. The result is a large number of semantically connected datasets, which can quickly become a mess to manage, especially when the work is carried out by different persons at different locations and times.

The master model may then be re-targeted depending on the intended purpose to produce different sorts of *output models*. For example, high-quality complex material and high-resolution geometry are needed for scientific detail inspection, for documentation, or for producing photo-realistic images; a mid-poly version can be used for interactive 3D environments; and a simple textured low-poly model for internet dissemination and as 'preview model'. A digital model may be archived, may be water-marked, chopped into separate pieces, marked and drawn upon by conservators, or sent to a 3D printer for reproduction.

1.1. The difficulty of collecting paradata

From a research and documentation point of view, the greatest problem is that the resulting output models have no clear relation to the measured input data. It is almost impossible to assess the *authenticity* of some part of a model, since the input data underwent so many different processing steps that it is not easy to tell whether a particular part of the surface stems from measurements, or was 'invented' by some algorithm, e.g., a hole filling procedure. This makes output models, strictly speaking, useless for academic reasoning since no reliable conclusions can be drawn, e.g., to prove or disprove scientific hypotheses about the artefact at hand. This leads some researchers with a Cultural Heritage background to the extreme position of denying the usefulness of 3D digitization and subsequent visualization altogether ("just beautiful misleading images").

The solution is to track the processing history of each dataset as completely as possible. The *London Charter* [Lon06] defines the notion of *intellectual transparency* that enables to distinguish between fact and interpretation. It is achieved by faithfully collecting *paradata* that describe the *digital provenance* of each dataset. However, collecting paradata requires a well-structured workflow: Each process-

DOI: 10.2312/VAST/VAST10/105-114

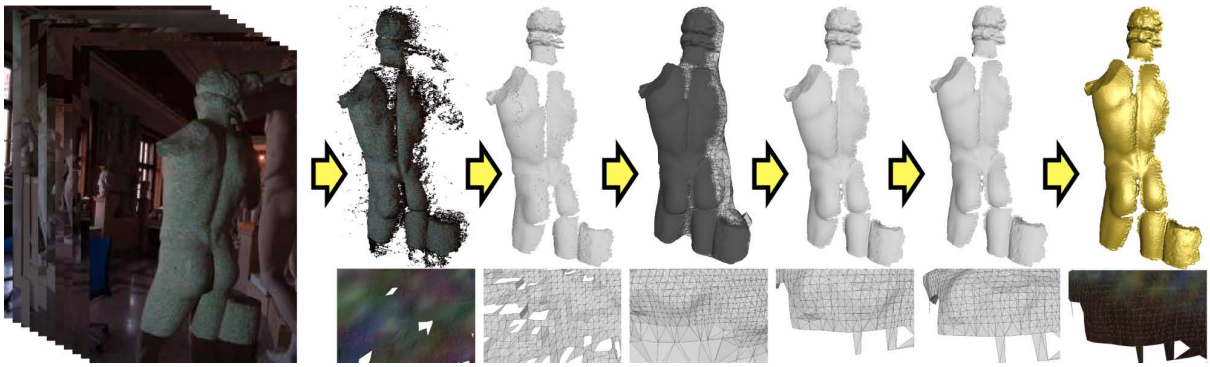

Figure 1: *3D reconstruction of a single photo sequence. Today the workflow typically proceeds in six stages: (1) dense matching produces one depth value for each pixel of each photo in the sequence; (2) the resulting overlapping range maps are cleaned (noise removal) and (3) merged and resampled (*Poisson reconstruction*) to produce a closed, single-layered, water-tight mesh. (4) The superfluous large faces are removed, (5) a 50% simplification removes short edges, and (6) the mesh is vertex colored. – This workflow was performed for each of the parts in Figure 2.*

ing step must be documented, preferrably in a format that is both human- and computer readable.

1.2. Workflow management problems

From a practical point of view, the 3D reconstruction workflow is difficult to manage and structure, and often it is likely to end up in a huge mess. Note that the workflow performs successive *data fusion* over different stages, in which the separate input data are processed and merged into a single model. A specific example of a practical workflow that illustrates the problems is the 3D reconstruction from photographs shown in Figures 1 and 2: Each photo sequence is processed in six stages to produce one mesh part, and 20 of these parts are fused to obtain the final model. Each tool that is used provides numerous options and parameters that are typically not documented. Sometimes the end result is not satisfactory, so that some of the intermediate processing steps need to be re-done in higher quality. Scheiblauer et al. [SZW09] have created a master model of the Domitilla Catacomb with a size of 30 GB - and the intermediate processing steps take a multiple of this. And it is easy to see that versioning is a nightmare when different operators at different locations are involved, or the processing stops for a while and is continued later.

Another level of complexity arises from the *heterogeneity* of the tools that are used, and also from the indispensable *ad-hoc workflows*: Each type of artefact requires slightly different treatment, even if the general workflow remains basically the same. The workflow that works well for statues is not the same as for busts, and it is totally inappropriate for the 3D-reconstruction of building remains. And with an increasing differentiation and specialization of a commercial 3D reconstruction sector, the range of workflows in practice will also become much more diverse.

1.3. List of issues to solve

To summarize, the infrastructure and reconstruction workflow must resolve and find good answers to these issues:

- **Storage:** On which machine is enough free disk space for processing the datasets?
- **Location:** Which datasets to process are on which network drive / on which machine?
- **Versioning:** Which datasets differ just in tool settings, which are for further processing?
- **Purpose:** Is the goal a high-quality master, an interactive model, or a preview model?
- **Soundness:** What was the reason for using these specific tool settings, and not others?
- **Retargeting:** Could the work be automatically reproduced in lower or higher quality?
- **Responsibility:** Who is to blame for which mistake in which step, and who did a good job?
- **Traceback:** Which set of input data were used for a certain sequence of processing steps?
- **Completion:** Can missing or low-quality data, or just parts, still be completed later on?
- **Accounting:** Which step did take how much time to complete in which quality?
- **Sustainability:** Which parts of the whole processing sequence need to be preserved?

1.4. Solution approach: Repository-centric process

3D reconstruction has two main ingredients, the *infrastructure* (the technical tools), and the *process model*, the conceptual frameork for the organization of the different workflows. One process model that is very often used is the *pipeline*.

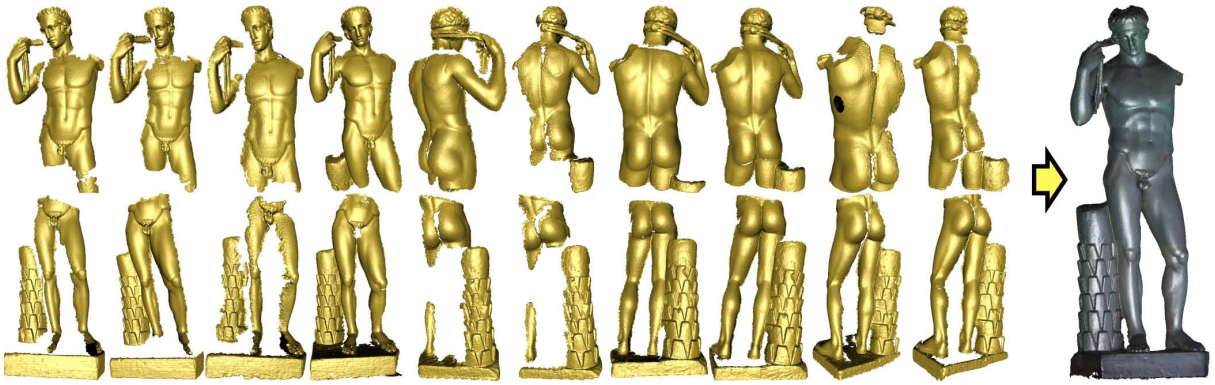

Figure 2: *Alignment and fusion. 20 sequences were taken for this statue in a very systematic way (upper / lower part), but the resulting model still has holes under the arm and on top of the head. Such problems become apparent often only with the final model. So despite careful planning, each reconstruction has exceptions and needs a slightly different workflow.*

The idea is to provide a set of tools and to make sure that the output produced by one tool can be used as input for the next. The problem of this somewhat naive approach is that there is no monitoring framework that enforces the consistency of the workflow. Quality standards can not be enforced and, more seriously, can not be verified a posteriori.

Our approach to solving the aforementioned issues and problems is to switch to a *repository-centric* process model. The idea is to use a centralized (but distributed) data repository, both for storage and for metadata management. This approach is based on the observation that any management unit introduces a certain administrative overhead that is typically not accepted by users that work under tight time and cost constraints. So we combine the management unit with a storage service that has undeniable advantages also for practitioners: Data to be processed can be retrieved from any location, even large-volume processing results are reliably stored, and the process information is gathered on a per-file basis using an as-lightweight-as-possible reporting scheme. So the semantic context of each processing step, and of each file, can be faithfully preserved.

A refinement of the overall idea leads to the design decisions explained in section 3, and to the implementation described in the following sections. But before that we look at possible alternative approaches and tools.

2. Related Work

Many solutions exists for managing digital content, ranging from digital libraries such as *Fedora* [fed] with MIME-type based content storage, aggregation and metadata tagging, over a plethora of *content-management-system* (CMS), to 3D-PDF [Ado08] as dissemination standard. Such systems and formats are typically strong on the metadata side, e.g., they can enforce the referential integrity of internal links. Problems are that these are heavy-weight solutions with their own data management philosophy (like Fedora), they are not for large-volume datasets (scalability issue), they have no notion of (ad-hoc) workflows, and they provide only limited support for reasoning on digital provenance.

Content-based retrieval systems focus on collecting quality-controlled multimedia datasets, allow markup and link insertion, and make data accessible via sophisticated retrieval engines. Berndt et al. [BKHS09] have presented an extensible multimedia library for architectural 3D-models and music. But again, there is no workflow support.

Versioning systems like CVS, GIT [git], and in particular Subversion (SVN) [SVN] are very convenient to use and provide data transfer capabilities. However, their metadata support is limited and specialized, they also have scalability problems (SVN doubles the space locally required), and, most importantly, we explicitly deny the possibility of changing datasets (*write-once policy*, see sec. 3).

The *grid computing* community provides great tools for scalable transmission, e.g., for physics institutions like CERN to share their huge datasets with other institutions. Indeed we tried to incorporate one grid system with a large user base, the *Globus Toolkit* (GT). But we found that it was much like breaking a fly on the wheel since it is extremely complex to install and use, to the point that it became totally unmanageable with our limited resources. Furthermore, depending on external developments can be a problem (known as *vendor-lock-in* anti-pattern in software engineering). GT has many components we do not need, and from version 5 on (Jan 2010) the Java API of GT was suddenly discontinued.

Object repositories specifically tailored for Cultural Heritage have received only limited attention in literature. Sometimes they appear as a side note in processing pipeline frameworks. Cosmas et al. [CIG*03] have described a comprehensive CH digitalization pipeline ranging from data acquisition over 3D reconstruction to visualisation and data

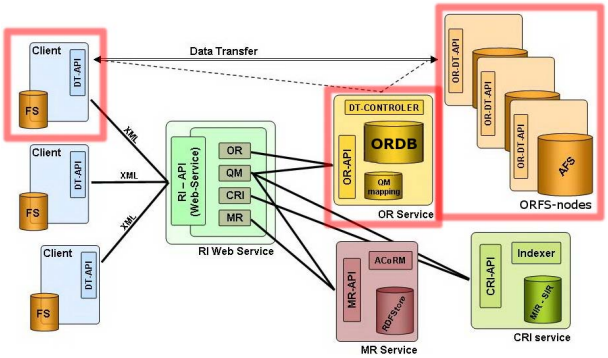

Figure 3: *The complete repository. Clients communicate via a central web service, binary transfer is de-centralized via AFS. This paper describes only the highlighted components.*

storage. But during a field campaign, new data can not be ingested due to the centralized design; little attention was given to workflows; collecting paradata was not enforced.

Very interesting systematic foundations for so-called *distributed digital object services* for CH in general were laid out by Kahn et al. [KW95]. The central notion in their seminal work is the *digital object* consisting of two parts, a (typed) dataset and descriptive metadata that must contain (at least) a globally unique ID (called *handle*). They propose a distributed repository infrastructure consisting of a number of *repository nodes* to store data, again each with a unique ID. Actors (*originators*) provide data and interact with one node at a time via a *Repository Access Protocol* allowing to store and access digital objects. A globally available *handle service* is responsible for registering objects, it knows about each object in each repository node and is invoked whenever a digital object is ingested or retrieved.

Our system incorporates many of these ideas and concepts, as well as it introduces new ones, as described next.

3. Repository Design Decisions

This section presents the fundamental *design decisions* (DD) of our system, in the order from very general to very specific, that result from the guiding idea of a repository-centric approach with enforced paradata collection. Note that our implementation is in fact just one example; we hope that the concepts we provide serve as guideline also for slightly different systems.

DD1: Repository-centric workflow. All data are *ingested* in a centralized storage, the *repository*. For processing, data are *retrieved*, processed, and the processing results are again ingested.

DD2: No dataset without metadata / paradata. Each file to be ingested must be accompanied by a metadata file (in XML) that describes in a formalized way the process that produced it, and the input data that were used for it. Both the dataset file and the metadata file are stored.

DD3: All identifiable units are assigned a UUID. Each object that may be subject to reasoning (facts expressed in XML), in particular each ingested dataset and metadata file, is assigned a *universally unique identifier* (UUID).

DD4: Write-once policy for datasets. Datasets are immutable. Changing a dataset, e.g., editing a triangle mesh, is considered a processing step that must be documented. Intellectual transparency requires *referential integrity*: When dataset A is processed to produce dataset B, changing A can make it impossible to produce B from it.

DD5: Datasets can be deleted, metadata can not. It may be that a very uninteresting 30 GB dataset is not retrieved for many years. In such cases, garbage collection is allowed to free the storage. However, referential integrity demands that the (typically small) metadata are preserved because a link may point to the reference.

DD6: Metadata are versioned. The XML metadata for a specific dataset may somentimes contain errors, e.g., the description of the digital provenance was incomplete, or the standards for describing provenance change, which requires changing the metadata. The older metadata versions are kept, e.g., to clean the semantic network reliably.

3.1. Why is using UUIDs so convenient?

The UUID is defined in two ISO standards [ISO96, ISO08] as 128-bit number with a string representation of 32 hex digits with 4 hyphens. Their generation rule has the important property that it is extremely improbable that the same UUID is generated twice (*UUID clash*). The set of 2^{128} possible UUIDs is in fact gigantic: Six billion persons can generate, over a period of 2000 years, in every nanosecond 500 million UUIDs before the whole set is used up. Since UUID clashes are so improbable, a UUID can be generated locally, on the *client side*, where the processing is carried out. UUID generating functions are part of almost all operating systems, e.g., the `uuidgen` commandline tool on Unix. UUIDs have therefore the important advantage that reasoning about datasets, which requires IDs, becomes possible without having to contact a centralized ID generator. An archeologist can acquire data in the field and already start to process, while each processing step is documented: The metadata manager on the local machine simply assigns a new UUID to each file that is being generated. It uses it, e.g., to build up the *processing dependency graph* that contains a node for each dataset and a directed edge connecting input(s) and output(s) of each processing step. After the field campaign, data and XML metadata are batch-ingested, and in the repository it is still completely clear in which stage of processing each dataset is.

3.2. Object repository and metadata repository

3D datasets and metadata are in fact two very different kinds of information: 3D data allow spatial or geometric reason-

ing, but have no attached semantics. The notion of 'spatial semantics' is slightly problematic for 3D datasets, as was explained by Havemann et al. in [HF07]. We account for this by distinguishing between two types of repositories (Fig. 3):

DD7: RI = MR + OR. The core of our *repository infrastructure* consists of a *metadata repository* (MR) and an *object repository* (OR). Neither of them is directly accessible from outside; all requests go to the (thin) RI layer, which dispatches them to the MR and the OR.

DD8: The MR contains a semantic network. In the XML metadata files, facts are expressed as *RDF triplets*, basically as declarative sentences (subject-verb-object) such as Dataset A - was measured by - Device X. The whole of such facts makes up a network of so-called *entities* connected by *properties* (the verbs).

DD9: OR = ORDB + ORFS. The ORDB is a (relational, see sec. 4.2) database with tables for datasets, groups, users, UUIDs etc. The file store (ORFS) is basically a very large, flat list of files. Any part-of relations (virtual directories) are stored in the ORDB.

In our case, paradata and processing information is expressed as RDF facts in the semantic network. It uses an extension of the CIDOC-CRM [CDG*05], an ISO-standard for cultural information modeling, that is capable of describing measurement events, processing events, actor roles, digital derivations, etc. The principles of information modeling in the MR are described in a separate publication.

A potential problem with semantic networks is *network pollution*: Wrong facts (RDF triplets) can lead to false reasoning. But the effects can be subtle, and pollution is sometimes difficult to detect. The OR can help here.

DD10: All changes to the MR are ingested in the OR. Since it is very difficult to undo modifications of a semantic network, especially after further modifications, it was decided to store all modifications also as XML metadata files in the OR. This way, the MR can be rebuilt from scratch when it is messed up.

DD11: The MR refers to OR datasets via their UUID. UUIDs provide the link between both parts of the repository. A semantic query can yield, for example, all datasets that were produced using a particular sort of device, since the identifiers A and X in the example above are UUIDs.

3.3. Privacy considerations and requirements

The great vision for the future is that 3D reconstructions are no longer carried out by research organization in pilot projects, but by professionals in a commercial, quasi-industrial setting on a mass scale. 3D digitization offers great opportunities since digital assets have undeniable advantages, only to mention: They are much easier to inspect, manipulate, disseminate, archive, compare, and reproduce – which in turn leads to a much better maintenance and protection of the real artefacts.

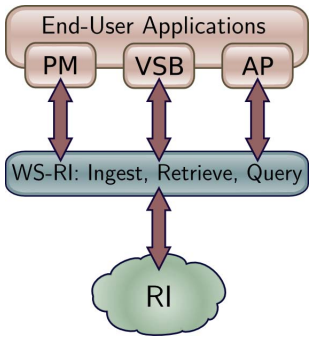

Figure 4: *Layered Component Architecture. Higher layers build on top of lower layers and add functionality for domain-specific tasks: end-user application domain with tools for e.g. data aquisition and processing (AP), visual querying and browsing (VSB), presentation and modeling (PM). Below, the web service layer, finally the core distributed repository infrastructure (RI) from Fig. 3.*

This level of professionalization, however, requires a reliable exchange platform. 3D digitization of valuable artefacts, e.g., pieces of art, is a delicate issue since the digital replica must be considered equally valuable. Especially museums have very high demands in terms of security. The Louvre database, for instance, has no direct Internet connection, and in France, museum data are treated using security classifications from military [oral communication]. Such a setting is a serious obstacle since, e.g., commercial companies doing contract work on 3D reconstruction have difficulties to access the data. To some extent, we are faced with a paradoxical situation: A centralized repository is the only way to solve the problem, but none of the affected institutions are willing to share their data.

DD12: ORFS is distributed, but ORDB is centralized. Each institution may run its own ORFS node. The central ORDB knows about the existence of the files in all ORFS nodes. This is a prerequisite for gathering paradata, since files from one instituition may be processed in another.

DD13: Institution have full control over datasets they own. Every transfer of a dataset from one node to another, or to a client, is faithfully recorded, and explictly stored in the ORDB. This is possible since all transfer request go through the RI. In particular, the owner of a dataset can trigger its physical deletion on all repository nodes.

DD14: Each dataset can have any number of replicas. A dataset has no unique or preferred storage position. The actual file can reside on any number > 0 of repository nodes, each such instance is called a *replica*. Deleting the last replica means deleting the dataset (but not its metadata, see DD5).

3.4. Poly-hierarchical grouping

One complication is that each dataset may be used for different purposes. It may be that only a subset of the raw data is used for a particular reconstruction, and a different subset for another one. It may be that a master model is produced, but it turns out to be of insufficient quality, so that some intermediate processing steps have to be re-done in with different tool settings. So it is very convenient if it is possible to flexibly create groups of files needed for a particular purpose; but each file can be part of different of such groups.

DD15: Virtual directories by poly-hierarchical grouping. The ORDB supports a grouping facility; each group has a UUID to enable reasoning about it in the MR. Groups can contain datasets as well as other groups. Each group has metadata attached that describe its purpose.

3.5. Area tables as link anchors in datasets

Sometimes it is necessary to express facts not about a whole model, but only about some part of it. All busts show a face; all faces have mouth, ear, and eyes; and so on. To make it in principle possible to retrieve, e.g., all noses of all statues in the repository, the ORDB supports the notion of *area tables*.

DD16: Each dataset can have an area table attached. An area table is a special kind of XML file with a list of media dependent area definitions: The partition of an image, e.g., into fore- and background (segmenation); a bounding box around a set of triangles; a curve in space describing a feature line; and so on.

DD17: Area tables are versioned. At any given moment, there is only one specific table of areas for a dataset. However, unlike datasets, area tables contain *interpretations* and can, thus, be changed. Area tables typically only grow; an area can only be deleted if it is certain that it is no longer referenced *anywhere*.

3.6. ONI datasets: *Objects Not Ingested*

In order to maintain the paradata consistent, it may be necessary to perform reasoning about digital assets without an associated dataset. This may be, for instance, inaccessible data, very large volume data, or data that are simply not stored on a location that is part of the repository.

DD18: ONI-datasets have no binary file *Objects not ingested* are incomplete datasets. They behave like normal datasets, i.e., can be reasoned about, added to groups, etc. But when retrieved, a file of size 0 bytes is created. The binary file can be ingested anytime later, thus turning an ONI dataset into a normal dataset.

4. Implementation

The purpose of this section is to describe our example implementation of the concepts and design decisions from the previous sections. It shall convey a very concrete idea of how the design decisions can be put into practice. According to DD9, the two parts of the OR are described in two sections.

4.1. ORFS: The OR File Store uses AFS

The *Andrew File System* (AFS) [MSC*86, HKM*88] is a distributed file system with user management, access control lists, Kerberos-5-based user- and service-authentication, efficient caching to reduce bandwith usage and, as an option, encrypted network traffic. AFS is conceptually mature and reliable in practice. The OpenAFS [AFS] implementation is available under free IBM Public License, clients exist for Linux, MacOS X, Windows, BSD and Solaris.

The highest logical level are *AFS cells* spanning a name space for files and directories, consisting of one or more file servers and several administrative database servers. There is usually one cell per institution, e.g., per museum, per archive, public office or enterprise. AFS stores its data in *volumes* that can be mounted anywhere in the cell. A sophisticated caching scheme makes sure that often used data reside on a computer close to the user that needs it.

How AFS is used as ORFS. Normally, the AFS client is installed on end user computers. It gives access to the AFS network file system, AFS drives are mounted and mapped to a local device name. Repository users, however, never get in direct contact with AFS. No AFS client is installed on their client machines. The system overview in Figure 3 shows the main *repository server* in the center. There is one AFS cell for the whole repository. Each participating organization running a repository node has one *AFS file server* running on their node computer (right).

End users typically belong to an institution and have an account on the respective node. The node computer holds a hidden AFS ticket for each user that is logged in, to retrieve datasets from this or other nodes. Access to files is granted based on the access rights stored in the ORDB, not the AFS.

4.2. ORDB: Relational DB, not semantic network

The OR works closely together with the semantic network of the MR (DD7-9). Since all information about OR datasets could in principle be stored in the network, the question arises why the OR needs a separate, relational DB. One reason is performance: Searching in a relational DB with its known structure is much, much faster than traversing the semantic network. Second, it is easier to actually enforce a certain structure in a relational DB; and the same basic information (size, date, etc) must be available for all datasets.

Still, the OR is incomplete without the MR, since the ORDB is very schematic and can definitely not represent the kind of provenance detail that a RDF-based database can store. The OR keeps only a backup (in XML metadata files).

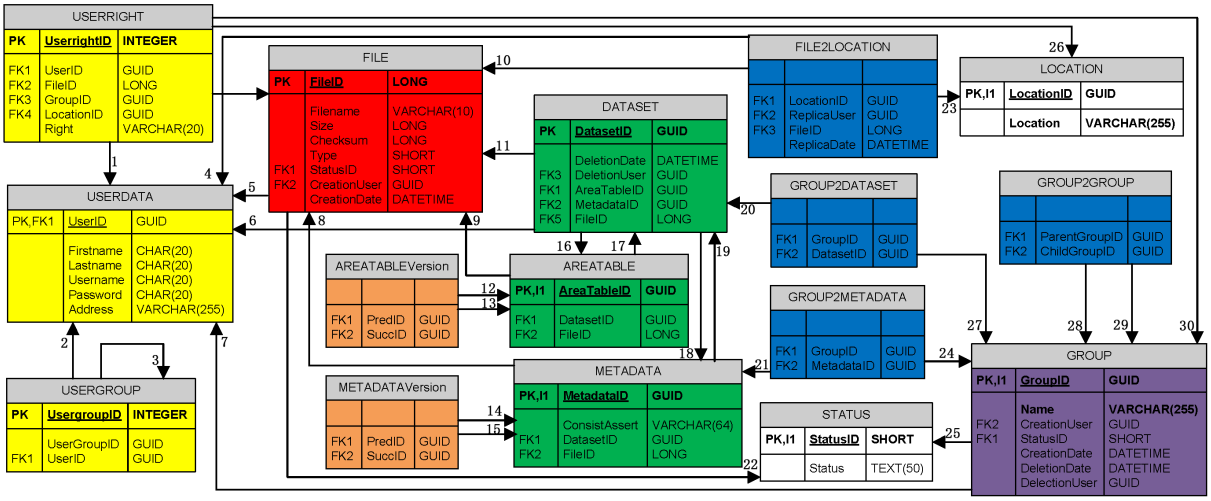

Figure 5: *The database schema of the ORDB. The purpose is color-coded: the central File table (red), information about the files (green), link tables (blue), dataset grouping (violet), revision management (orange), user management (yellow), miscellaneous (white). Arrows indicate references explained in the text, "PK" is primary key and "FK" foreign key.*

4.3. ORDB: The OR DataBase

The OR database scheme shown in Fig. 5 maybe gives the best and most concrete idea of the implementation; therefore we present it in some detail. For better overview, the tables are color-coded according to their purpose. The blue group has a special role, as its purpose is interlinking, so it is discussed within the other groups.

The File section (red, green). The OR can store three types of files (green tables): Datasets, Metadata and AreaTables. Each of them is associated with exactly one File, so arrows 8, 9, 11 represent 1:1 relations. While the latter two are XML files, Datasets are binary files. As DD2 specifies, each Dataset is tied to exactly one Metadata file (arrow 18) and (DD17) at most one AreaTable file (16). Each File can reside on many different repository nodes, which is accounted for in the File2Location table. The Location table contains exactly one entry for each repository node (with AFS file server) so both arrows 10 and 23 are 1:n.

Versioning information (orange). DD6 and DD17 require Metadata and Areatables to be subject to versioning: While the Dataset references only the most recent version, the Version tables realize a doubly linked list that makes it possible to revert to previous information in case of errors. Accordingly, the arrow pairs 12/13, 14/15 are 1:1 relations (except for the first and the last).

Poly-hierarchical grouping (violet). A group acts like a virtual directory (DD15). The members of a group are stored in the tables Group2Dataset and Group2Group. Concerning the latter, note that a Group can have any number of parents. The Group structure resembles more a general graph than a directory tree; in principle, it can even have cycles, although there is no real use for them. So unlike with file system directories, the parent directory ("..") depends on the context. Navigation always requires maintaining a full path, and *any* Group can be used as entry point (root node) of the path. The purpose of forming this particular group structure is motivated by the Group2Metadata relation.

Groups are lightweight and can be used as *labeling mechanism*: There can be groups for all statues, for the Gothic period, for lengthy objects, etc. The group label is described in detail by the metadata attached to each group. - Note that grouping can also help dealing with, e.g., *ZIP-archives*, which are files containing a set of other files. It is possible to create a Group of datasets that all point to the same archive file, the group metadata containing the zipfile information. So the grouping functionality can potentially overlap with purpose of the semantic network. Yet again, the argument is that searching in a relational DB is faster.

User and rights management (yellow). A design decision on the implementation level is that only Files are subject to user rights management, simply not to overcomplicate it. Access rights must be checked once at ingest time and then later on for each and every retrieval (i.e., file download). For simplicity, the UserData table can stand both for individual persons and for user groups. Similar to Group2Group, the UserGroup table can express a poly-hierarchy; usually it should reflect the hierarchy of a participating organization.

The Status table. As prescribed by DD2, dataset and metadata must be ingested together. However, in case of transmis-

sion errors it may occur that File fields like Checksum and Size, or information in the metadata, is inconsistent with the dataset. The type of detected inconsistency is described in the Status table. Since the integrity of a group depends on that of its members, the Group also refers to a status.

5. Communication API

The OR is implemented in Java. Despite the complexity of maintaining the database and the distributed storage consistent, the whole OR system can be operated using only a very small set of ≈ 19 functions (plus a few administrative functions needed for technical reasons), the so-called *OR-API*. They are issued using SOAP calls to the OR webservice. For simplicity, all functions have the return value Result containing a flag for success or error, and a Java Object reference as a hook to return any collection of objects.

User identification (login/passwort) and session management are handled centrally by the RI webservice. The following functions are members of a class ORSession that is instantiated in the OR service for each user session. The ORSession constructor receives a *session ticket*, containing the user ID and a session UUID. The session ticket is sent for identification whenever the RI calls the OR (using SOAP); within the OR it is used to identify the right ORSession object. So the session ticket is not part of the signatures of the ORSession functions. Functions for user and access rights management are omitted here, they are straightforward.

Ingestion of datasets and metadata

```
Result createDataset  (GUID datasetID, Dataset struct);
Result setMetadata    (GUID datasetID, Metadata struct);
Result ingestDataset  (GUID datasetID, File path);
Result ingestMetadata (GUID datasetID, File path);
```

Ingestion is a two-stage process: First the database entries are created (Fig. 5), then the files are transmitted. Unlike createDataset, the setMetadata function can be called more than once, to update the metadata attached to a dataset. The ingest functions initiate the actual file transfer from client to server. The built-in CRC mechanism verifies the success of the transmission, which is vital to maintain consistency. To some extent, the MD5 sum can be used to detect duplicate binary datasets. Once the dataset is ingested, it is immutable (DD4), so calling ingestDataset again makes no sense. The path of ingestMetadata points to the XML file.

Retrieval: Accessing datasets and metadata

```
Result queryField      (GUID datasetID, String field );
Result retrieveDataset (GUID datasetID, File path);
Result retrieveMetadata (GUID datasetID, File path);
```

These functions retrieve information about repository datasets. The field string parameter describes the database field relative to the chosen Dataset using a path expression: FileID returns the FileID of the Dataset, while MetaFileID.FileID that of the (current) metadata file. Using AreaTableID.FileID.CreationUser the person is identified who added the last area. retrieveDataset is *the* central routine to retrieve complete datasets with data file, metadata, areatable. It operates asynchronously, i.e., a callback is triggered when the files are available on the client side. The path must be a directory. Metadata can also be retrieved separately.

Poly-hierarchical grouping

```
Result createGroup      (GUID groupID, GroupNode struct);
Result addToGroup       (GUID nodeID, GUID groupID);
Result removeFromGroup (GUID nodeID, GUID groupID);
```

Any user can create a new empty dataset group (DD15) and add to it any number of datasets and sub-groups, e.g., sequences of RAW photos of a statue. This powerful feature is the key to structuring workflows, and to a lightweight paradata reporting scheme. Note that a nodeID can equally refer to a Dataset or a Group. Group metadata are set using the same setMetadata function as for datasets.

Queries

```
Result query       (SQLStatement sql);
Result setAccess   (GUID fileID, Location id,
                    GUID userID, String right );
Result checkAccess (GUID fileID, Location id,
                    GUID userID, String right );
```

As any other relational DB the ORDB can be queried using SQL statements. The Result typically contains a Table (list of records), e.g., a list of UUID(s) of queried Files or Datasets. Note that the RI does not provide any function to change the database directly; this can only be done by database administrators (using the ORSession service functions that are not listed here). Four rights exist: *Ingest, Retrieve, Update, Delete*. Since binary datasets are never updated (DD4) the UPDATE right just means the user is allowed to update metadata.

Replica management

```
Result createReplica (GUID datasetID, GUID locationID);
Result deleteReplica (GUID datasetID, GUID locationID);
Result eraseDataFile (GUID datasetID);
```

Creating a replica triggers the creation of a physical copy in a remote location (provided the user has the right to do so). The user and date of a replica creation are logged in File2Location. Note that it may well be the case that no user in the target location has the right to access the file (backup scenario). On the other hand, even if a user in another location has the right to access a file, it remains inaccessible unless he has also access to the node (i.e., he be-

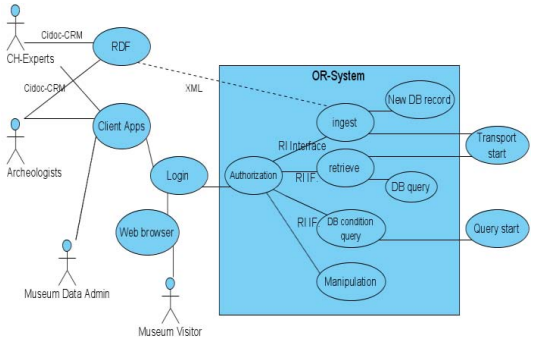

Figure 6: *Use case diagram for ingestion, retrieve and query. The ovals represent operations, the rectangle symbolizes the OR system*

longs to the node UserGroup). Deletion of the last remaining replica using deleteReplica will fail. Authorized users, however, can erase a binary dataset completely from the whole ORFS; only its metadata are not removed (DD5). The dataset is physically removed from *all* repository nodes, and the File2Location table (replica list) is cleared for it. The Status of the File is set to *deleted*, the DeletionDate and DeletionUser are set appropriately. Replica deletion may be logged as well, if necessary (future work).

Area tables

```
Result setAreaTable    (GUID datasetID, File path);
Result getAreaTable    (GUID datasetID, File path);
```

As specified by DD15 and DD16, areas in a dataset can be defined by any user with *Retrieve* permission. For simplicity, they are stored as a single file (also in XML). Users typically retrieve the area table, add an area, and ingest it again. The XML file is structured such that it permits conflict-free merging of areas defined by different users. This merging is done transparently by setAreaTable, so it may be that the XML file given as parameter is not identical to the one that will be entered into the ORFS.

6. Use Cases

The functionality of the OR system is illustrated by the use case diagram in Fig. 6. The OR is designed as a component of an integrated repository infrastructure and can be entirely operated with the concise API from the previous section. The obvious main use cases are uploading/ingesting digital artefacts, downloading/retrieving the digital artefacts, and querying the database to obtain meta information.

6.1. Ingestion of datasets

Suppose the user has a number of valuable data files like raw measurement data, semi-finished models, or a finished master model. Each file (e.g., seq-12_22.jpg) on the harddisk is complemented by an XML provenance metadata file that complies to the CIDOC-CRM [CDG*05] with a similar filename residing in an invisible sub-directory, e.g., .metadata/seq-12_22.jpg.metadata.

The user connects to the RI through stand-alone *client software*, identifies with username and password, and obtains a *session ticket* from the repository server. The user also needs to have rights for a location, i.e., a repository node. On this node, a (hidden) AFS/Kerberos ticket is obtained to access the AFS server. The AFS ticket is also stored in the ORSession class on the OR (middle red box in Fig. 3).

The client software has an integrated browser for the local file system to select all the files to ingest. When ingestion is started, a SOAP call triggers the data transmission (metadata and dataset) directly to the chosen node (Data Transfer in Fig. 3). Data are stored on the node in a temporary folder where they are checked for integrity (CRC, MD5), and the metadata are sent to the MR for validation. If positive, the metadata is processed and entered into the central semantic network (MR), and the datasets are transferred from the temporary folder to the node's AFS by calling the ingestDataset and ingestMetadata functions of the OR webservice (called OR-DT-API in Fig. 3) of the node. Finally, the central ORDB is updated by calling createDataset and setMetadata, here the OR-DT-API webservice of the node calls the OR-API. In case the transfer process was not finished properly, the process will be reset and the ORDB remains unchanged; the user receives a notification to re-send the corrupted datasets, or to repair the metadata.

6.2. Retrieval processing flow

From the user perspective, the retrieval process starts in a similar way as ingestion. The main difference is that now the browser shows the Group hierarchy of the ORDB server instead of the local file system. Both the ORDB and the semantic network can be queried (using SQL and SPARQL, respectively) to obtain the list of data to be processed. The user may also choose to create groups to collect datasets according to his requirements.

When the user decides to download the chosen data to the local harddisk, first the access rights are validated; retrieval is denied if either the user does not have sufficient access rights, or if no replica is on a node accessible to the user. If neither applies, the client software triggers the server function retrieveDataset using SOAP. When the transfer is finished the built-in transfer integrity check verifies the transfer process. In case of failure, the user will get notice.

6.3. Query processing flow

There are two main query scenarios: (1) The user has a list of dataset UUIDs and wishes to obtain further information and

metadata. Then he can use the client software to obtain file size, ingestion date etc. by issuing the queryField function on the node's OR webservice. (2) The user does not know the UUID but has some attributes of the desired datasets. If this is the case, the user again communicates first with the RI to obtain a session ticket, and then triggers through the RI the query() function of the OR-API. The known attribute values are converted to a conditional clause in an SQL SELECT statement. As a result of the query the ORDB returns the dataset IDs to the user who can then retrieve the binary data for local processing according to the chosen workflow.

7. Conclusion and Future Work

We believe we have laid out sound foundations for a reliable and useful object repository for Cultural Heritage. To our experience, almost all archeologists, curators, conservators etc. share the same grief about chaotic and hardly manageable digital content creation workflows. Our project could be a step forward for creating better models in shorter time. So, much has been achieved, but also much remains to be done. Many question are not yet answered.

We are very eager to provide our system to practitioners for obtaining qualified feedback on whether its usefulness really outweights the administrative overhead. Also open is whether the architecture scales up to very large user bases; we anticipate a slight bottleneck when too many users contact the central repository server. We have not yet carried out any stress tests with many parallel requests for large volume data, or for a large number of DB queries per second.

More fundamental is the question to what extent paradata needs to be captured in order to be useful. Although our system does not prescribe any particular granularity of storing intermediate results, we anticipate that many of them are in fact redundant, even if they are obtained by manual work.

Also challenging will be to define the details of the lightweight reporting scheme. Even though many data remain the same (same author, same device, etc) over the duration of an acquisition campaign, it may still be that the exceptions that occur with almost every acquisition may be too tedious to describe in practice. We have to explore generative approaches for creating flexible, parameterized metadata templates that can be instantiated procedurally.

And finally, we would like to know whether our system can easily be used with other more domain specific semantic databases. Although the OR design does not depend on CIDOC-CRM, it was developed with the CRM in mind. If our system can be re-targeted to work with different metadata repositories, this would be a strong clue that we have found a sufficiently general object repository infrastructure.

Acknowledgements

We gratefully acknowledge the funding from the Seventh Framework Programme of the European Commission (FP7/2007-2013) under grant agreement no. 231809 (IP project "3D-COFORM").

References

[Ado08] ADOBE SYSTEMS INCORPORATED: ISO 32000-1:2008 - portable document format – part 1: Pdf 1.7, 2008. 3

[AFS] AFS:. www.openafs.org. OpenAFS: Open source implementation of the Andrew File System. 6

[BKHS09] BERNDT R., KROTTMAIER H., HAVEMANN S., SCHRECK T.: The probado-framework: Content-based queries for non-textual documents. In *ELPUB 2009: 13th International Conference on Electronic Publishing* (2009), p. 17. 3

[CDG*05] CROFTS N., DOERR M., GILL T., STEAD S., STIFF M.: *Definition of the CIDOC Conceptual Reference Model*, version 4.2 ed. CIDOC Documentation Standards Working Group, June 2005. Also ISO/PRF 21127, available from cidoc.ics.forth.gr. 5, 9

[CIG*03] COSMAS J., ITEGAKI T., GREEN D., JOSEPH N., GOOL L. V., ZALESNY A., VANRINTEL D., LEBERL F., GRABNER M., SCHINDLER K., KARNER K., GERVAUTZ M., HYNST S., WAELKENS M., VERGAUWEN M., POLLEFEYS M., CORNELIS K., VEREENOOGHE T., SABLATNIG R., KAMPEL M., AXELL P., MEYNS E.: Providing Multimedia Tools for Recording, Reconstruction, Visualisation and Database Storage/Access of Archaeological Excavations . In *Proc. VAST 2003* (Brighton, United Kingdom, 2003), Arnold D., Chalmers A., Niccolucci F., (Eds.), Eurographics Association, pp. 165–174. 3

[fed] www.fedora-commons.org. Fedora Repository Project: General purpose, open source digital object repository system. 3

[git] GIT:. git-scm.com. git - the fast version control system. 3

[HF07] HAVEMANN S., FELLNER D. W.: Seven research challenges of generalized 3d documents. *IEEE Computer Graphics and Applications 27*, 3 (2007), 70–76. (special issue on 3D documents). 5

[HKM*88] HOWARD J. H., KAZAR M. L., MENEES S. G., NICHOLS D. A., SATYANARAYANAN M., SIDEBOTHAM R. N., WEST M. J.: Scale and performance in a distributed file system. *ACM Trans. Comput. Syst. 6*, 1 (1988), 51–81. 6

[ISO96] ISO: Information technology – open systems interconnection – remote procedure call (rpc), 1996. ISO/IEC 11578:1996. 4

[ISO08] ISO: Information technology – open systems interconnection – generation and registration of universally unique identifiers (uuids) and their use as asn.1 object identifier components, Dec 2008. ISO/IEC 9834-8:2008. 4

[KW95] KAHN R., WILENSKY R.: *A framework for distributed digital object services.* Tech. rep., Corporation for National Research Initiatives, 1995. 4

[Lon06] LONDON CHARTER INITIATIVE (HUGH DENARD): The london charter, June 2006. www.londoncharter.org. 1

[MSC*86] MORRIS J. H., SATYANARAYANAN M., CONNER M. H., HOWARD J. H., ROSENTHAL D. S., SMITH F. D.: Andrew: a distributed personal computing environment. *Commun. ACM 29*, 3 (1986), 184–201. 6

[SVN] SVN: Subversion revision control system. subversion.apache.org. 3

[SZW09] SCHEIBLAUER C., ZIMMERMANN N., WIMMER M.: Interactive domitilla catacomb exploration. In *Proc. VAST 2009* (Sept. 2009), Eurographics Association, pp. 65–72. 2