On Preserving the Intelligibility of Digital Objects through Dependency Management

Yannis Tzitzikas

Institute of Computer Science (ICS)

Foundation for Research and Technology – Hellas (FORTH)
P.O. Box 1385, GR-711 10 Heraklion, Crete, Greece
EMail: tzitzik@ics.forth.gr

ABSTRACT

Digital information has to be preserved not only against hardware and software technology changes, but also against changes in the knowledge of the community. Apart from the bits we have to preserve the information and for this purpose we have to preserve the intelligibility of digital objects. This paper approaches this notion by introducing an abstract model of *modules* and *dependencies*. Community knowledge is formalized over the same model, by introducing the notion of *profile*. A preservation information system can use this notion as a gnomon for deciding *representation information adequacy* (during input) and *intelligibility* (during output). Subsequently some general dependency management services for identifying and filling intelligibility gaps during input and output are described and analysed.

Keywords: Digital Preservation, OAIS, Intelligibility, Dependency Management, Semantic Web.

INTRODUCTION

The preservation of digital information is an important requirement of the modern society. Digital information has to be preserved not only against hardware and software technology changes, but also against changes in the knowledge of the community. According to the OAIS Reference Model [1], metadata are distinguished to various broad categories. One very important (for preservation purposes) category of metadata is named *Representation Information (*RI) which aims at enabling the conversion of a collection of bits to something meaningful and useful. In brief, the RI of a digital object should comprise information about the Structure, the Semantics and the needed Algorithms for interpreting and managing a digital object (see Figure 1).

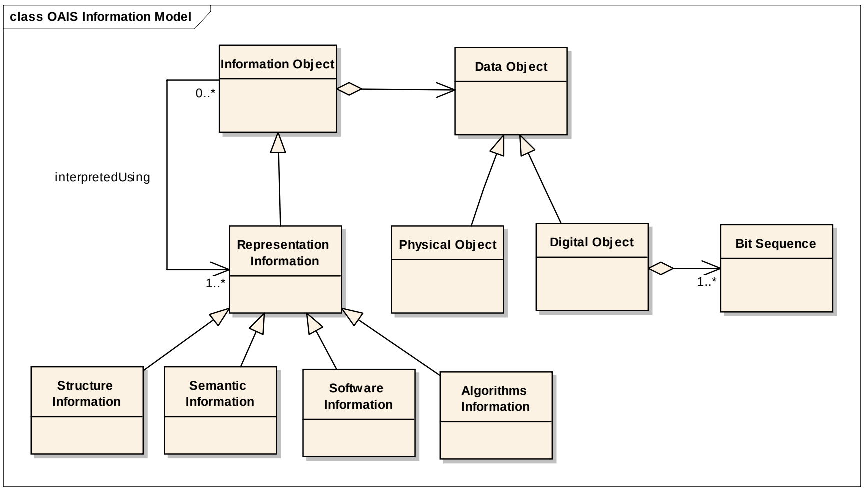


Figure 1 The Information Model of OAIS

In order to abstract from the various domain-specific and time-varying details, in this paper we model the RI requirements as *dependencies*. This view is very general and can capture a plethora of cases. Subsequently, we identify a set of core services for managing dependencies. These services aim at identifying the *knowledge gaps* (missing RI), and at computing and proposing ways to fill these gaps. These services can be used during both importing and exporting information (to and from a preservation information system). As different users or communities of users (consumers or providers), have different characteristics (in terms of RI), we introduce the notion of DC (Designated Community) *profile*. Subsequently, we describe protocols (interaction schemes) that could be used in the communication between a preservation information system and information consumers and providers.

In addition, a preservation system should be able to tackle the fact that everything changes over time (the world, our knowledge about the world, the uses of information/ knowledge). This means that preservation is an endless process. To capture and meet the related requirements we introduce the notion of *converters* (or translators) and we describe and analyze their role.

This work can be exploited for building advanced preservation information systems and registries. This research is being done in the context of the ongoing EU project CASPAR (FP6-2005-IST-033572)[1] whose objective is to build a pioneering framework to support the end-to-end preservation lifecycle for scientific, artistic and cultural information.

MODELING INDELIBILITY THROUGH DEPENDENCIES

In order to abstract from the various domain-specific and time-varying details, we introduce the general notions of Module and Dependency. We adopt a quite broad definition. A module could be a piece of software/hardware, a knowledge model expressed explicitly and formally (e.g. an Ontology), a knowledge model not expressed explicitly (e.g. Greek Language). The only constraint is that modules need to have a unique identity. Concerning dependencies, a module t depends on t', written t>t', if t requires t'. Broadly speaking, the meaning of a dependency t > t' is that t cannot function/be understood/managed without the existence of t'. So we model the RI requirements of the OAIS information model as dependencies between modules. Some examples are illustrated in Figure 2. For instance, the intelligibility of a digital object README.txt depends on the availability of text editors and knowledge of the English language. The rest examples come from the various testbeds of the CASPAR project (e.g. FITS files are used by astronomers). Dependencies also exist between formally expressed knowledge. For instance, the left part of Figure 3 sketches a number of Semantic Web (SW) schemas (recall that a SW schema may reuse or extend elements coming from different schemas) and the right part shows the corresponding dependency graph.

Concerning community knowledge, an actor or community u can be characterized by a ***profile*** Tu that contains those modules that are assumed to be available/known to u (i.e. Tu $\subseteq$T). For example, if u is an artificial agent, then Tu may include the software/hardware modules available to it. If u is a human, then Tu may include modules that correspond to implicit or tacit knowledge. Note that if preservation is done for a particular Designated Community (DC), we may call these *DC profiles*. One can easily guess that what the dependencies really are, strongly depends on the DC and on its purposes.

Now we introduce an assumption, namely the *unique module assumption* (UMA), which is very useful for both theoretical and practical reasons. According to UMA each module is uniquely identified by its name and its dependencies are always the same (in practice we may adopt a more relaxed assumption of the form: different modules have different identities).

If T denotes the set of all modules, then the dependency relation > is actually a binary relation over T. We shall use Nr(t) to denote the direct dependencies of t, i.e. Nr(t)={t' | t > t'}. We shall use $>^+$ to

[1] Work partially supported by European Community under the Information Society Technologies (IST) programme of the 6th FP for RTD - project CASPAR contract IST-033572. The authors are solely responsible for the content of this paper. It does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein.

denote the transitive closure of >, and $>^*$ to denote the reflexive and transitive closure of >. Analogously, we can define $Nr^+(t) = \{ t' \mid t >^+ t' \}$ and $Nr^*(t) = \{ t' \mid t >^* t' \}$.

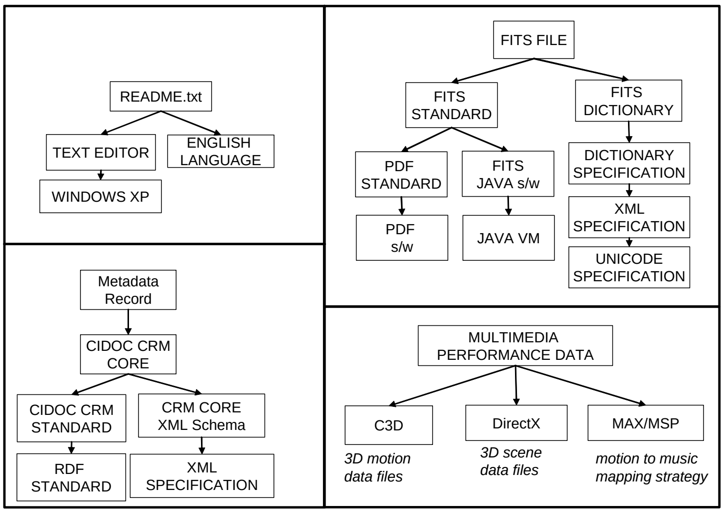


Figure 2 Examples of modules and dependencies

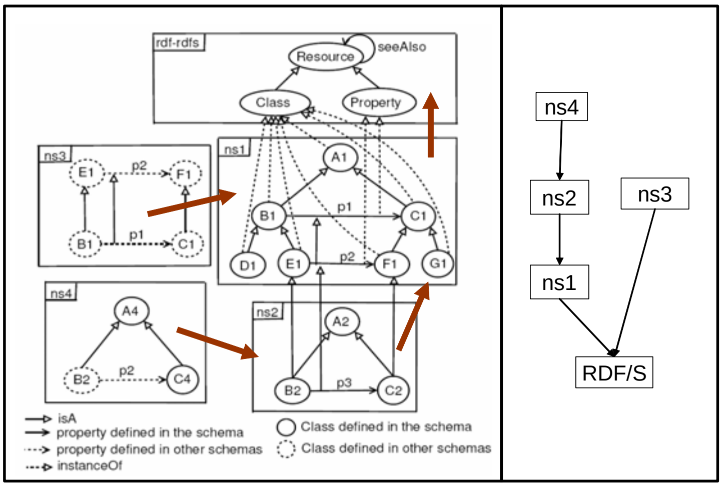


Figure 3 Dependencies between RDF schemas

In order to formalize the notion of intelligibility we introduce the notion of *closure*. The closure of a module t is defined as $C(t) = Nr^*(t)$. The closure of a set of modules S (where $S \subseteq T$) is defined as $C(S) = \cup \{ C(t) \mid t \in S \}$. As Tu is a set of modules, we can define its closure as C(Tu).

Recall that $Nr^+(t)$ is the set of all dependencies of t, i.e. $Nr^+(t)=C(t)-t$. We may denote this by $C^+(t)$, so it is actually the set of all (direct or indirect) dependencies of t. Figure 4 shows a dependency graph, and the profile Tu of an actor u.

It follows easily that u can understand a module t if and only if $C^+(t) \subseteq C(Tu)$. In the running example u can understand ty but he cannot understand tx.

We can now define the *intelligibility gap* as the smallest set of extra modules that u needs to have in order to understand a module t. We can denote this by **Gap(t,u)** and it follows easily that $Gap(t,u) = C^+(t)-C(Tu)$ where "-" denotes set difference. In our example $Gap(ty,u)= \emptyset$ while $Gap(tx,u)= \{t1, t2, t4, t5\}$. Notice that due to UMA it is implied that u can also understand t7 and t8 even if they are not elements of Tu. In addition, and due to UMA, we can decide whether a module is intelligible by inspecting only the direct dependencies of t. In particular it holds : $C^+(t) \subseteq C(Tu) \Leftrightarrow \max(C^+(t)) \subseteq C(Tu)$. In our example $\max(C^+(ty))) = t3 \in C(Tu)$, while $\max(C^+(tx))=t1 \notin C(Tu)$.

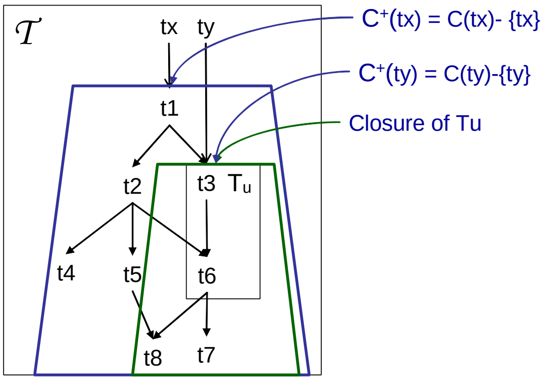


Figure 4 Dependency graph, profile and closure

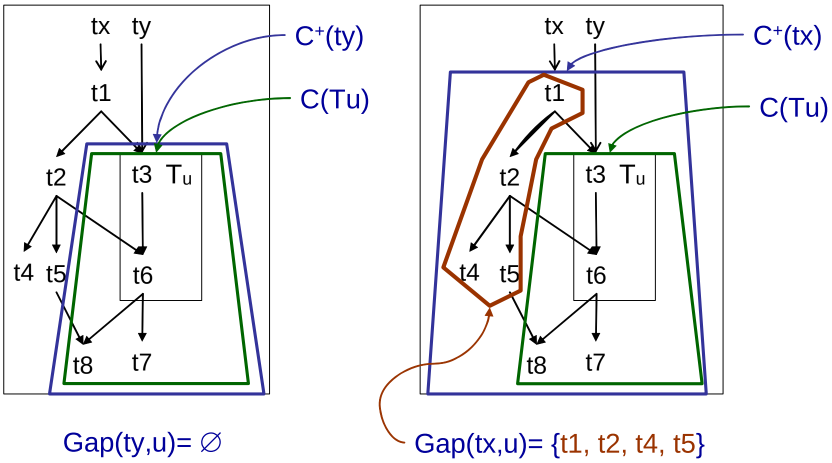


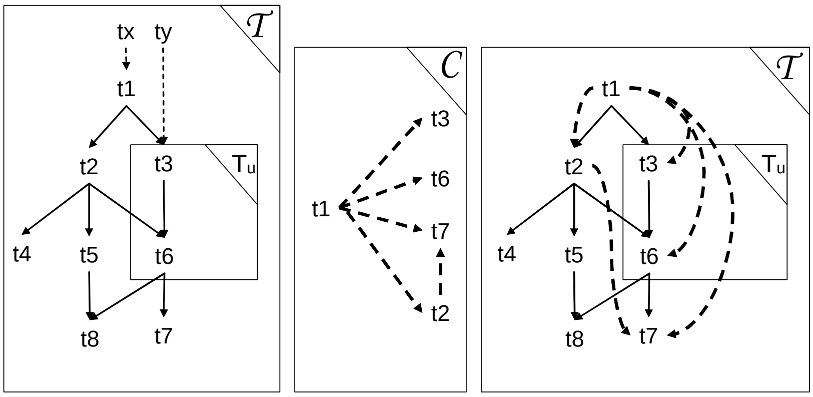
Figure 5 Example of Gaps

According to the previous discussion, an intelligibility gap can be filled by getting the missing modules. This means that if we want to preserve a digital object t for a community with profile Tu then we need to get and store only Gap(t,u). Analogously, if we want to deliver an object t to an actor with profile Tu, then the only extra modules that we should deliver to him in order to return him something intelligible, is the set Gap(t,u).

To summarize, the above formalism allowed us to provide specific answers to the following very important questions: (a) how much RI should we create? (b) how this depends on the knowledge of the designated community? (c) what automation can he have?

However, a preservation system should also be able to tackle the fact that everything changes over time. This means that changes in the dependencies should trigger intelligibility checks and may requests for further actions. In addition, it should be remarked that the notion of conversion (or transformation), which is a common practice for achieving interoperability in information systems, could also be exploited for the problem at hand. Specifically, for any module t we can use In(t) to denote the set of all inputs that t can recognize (valid well-formed inputs). For example a module called PDFReader can read all files with type PDF. From this point of view a converter from t to t' can be seen as a function f: In(t) → In(t'). We can model the available converters by a graph C = (T,→) where the edges denote converters. As there may be more than one converters from t to t', the graph is actually a multigraph. In particular, an edge t → t' represents a converter from module t to a module t'. For example, if Nr(o)=t and there is a converter t → t' then if we apply on o the converter t → t', we can get a new digital object o' such that Nr(o') = t'. Suppose an object o and a user u, such that Gap(o,u) ≠ ∅. For understanding o, the user instead of having to find and install the missing required modules, he could apply a number of converters so that to transform o to an object that he can understand with the modules already available to him. Specifically, such a conversion is possible (i.e. C contains the needed converters) if from every element of Gap(o,u) there is a path t →…→ t' in C where t' ∈ Tu (more precisely, t' ∈ C(Tu)). The problem of deciding whether this is possible reduces to the problem of REACHABILITY in directed graphs, i.e. from every t ∈ Gap (o,u) we want to reach at least one element of Tu. The plain REACHABILITY algorithm can return the shortest path (corresponding to the minimum number of needed conversions) for each element in O(|E|) where |E| the number of the edges. An example of converters is shown in Figure 6 where converters are denoted with dashed arrows. For instance, t1 which is not intelligible by u, can be transformed to an intelligible object if we apply to it the converter t1→ t3, or t1→t6, or t1→t7, or the sequence of converters t1→t2→t7. An alternative formal definition of the notion of conversion and emulation is given in [3]. However, we should not ignore that converters are themselves modules, therefore we could have a core model like the one shown in Figure 7.

We should also point out that our view encompasses preservation approaches that are based on virtualization. For instance, the UVC (Universal Virtual Computer) approach [5] also requires RI (as the specification of a UVC has to recorded and it may change over time). So we could say that the adoption of virtualization approaches just reduces the number of dependencies that we may have (however it does not vanish them). In the same spirit, MDA (Model Driven Architecture) and OMG standards aim at reducing the dependencies (instead of having a specification of an information system that depends on a number of different technologies, MDA proposes a specification that depends only on OMG standards).



.

Figure 6 Extending the model with converters

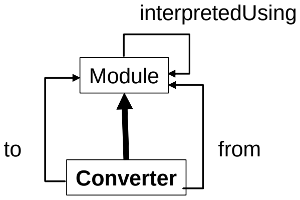


Figure 7 Modules and Converters

INDELIBILITY-AWARE PROCESSES

A Preservation Information System could adopt the notion of profile in order to support intelligibility-aware services. For instance, it could adopt the following policies: (a) the input (e.g. data objects to be archived) should be intelligible by the system, and (b) the output (e.g. returned answers) should be intelligible by the recipients. The notion of profile could be used as gnomon in these policies. Figure 8 illustrates some basic steps of these processes. They include the steps of selecting a profile, identifying the gap and filling the gap. Moreover, the impacts of changes have to be identified and the involved parties should be notified. The latter is part of the ongoing curation process. The impacts of changes on the modules and their dependencies are discussed in [3]. We should remark that these processes correspond to the elements of the functional model of OAIS

Another important remark is that identification and completion of an intelligibility gap could be done either in one step, or gradually. The former is in many cases difficult. For instance consider the example of Figure 3. The compute the closure of n4 one has to be able to parse an RDF file; otherwise he could deduce only ns4 > RDF/S (from the extension of the file). However, a gradual approach would allow one to first fill the gap of RDF/S (e.g. to obtain an RDF/S parser). After this step, he will be able to extract the dependency ns4> ns2 and then try to obtain ns2.

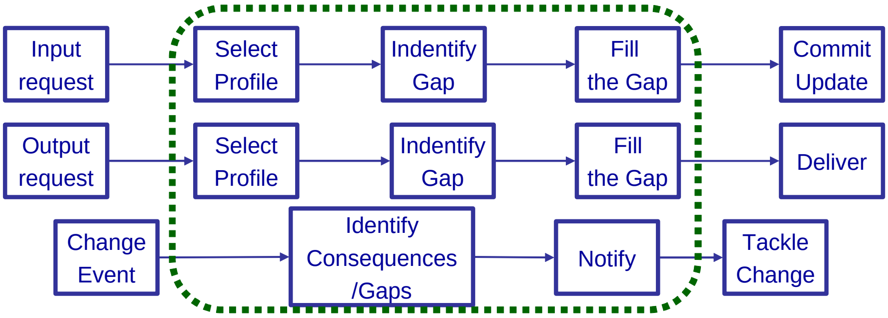


Figure 8 Intelligibility-aware services

Technical architecture

There are several options for implementing the above framework and the related services. One approach is to adopt Semantic Web technologies. Figure 9 illustrates a possible architecture of SW schemas and data. The upper level comprises 3 small SW schemas. The basic dependency management services could need to know only these schemas. The bottom layer shows an indicative instantiation of the above schemas. Between these two layers a number of other schemas can be defined that specialize/refine the elements of the upper schemas. For instance, the notion of module can be specialized (we could have a typology/taxonomy of modules). Furthermore the property class interpretedUsing can be specialized (the new specialized property class could have as domain and range subclasses of Module). One benefit of adopting Semantic Web technologies and an architecture of schemas like this, is that we can build a preservation system that needs to know only the upper schemas. To be more specific, this means that the queries may be formulated in terms of these schemas. However the same queries will function correctly even if the data level instantiates specializations of the above schemas. This is due to the semantics of specialization.

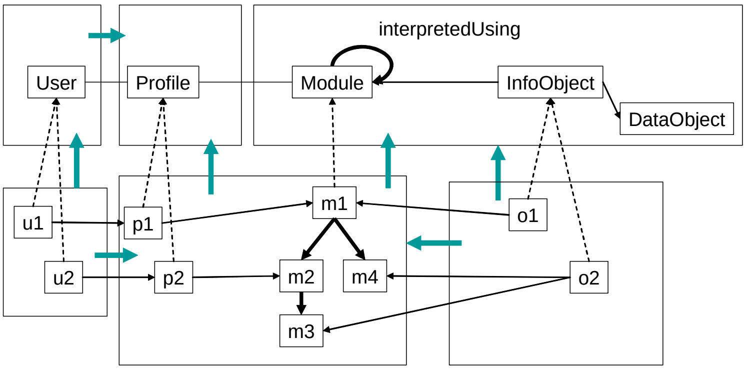


Figure 9 Architecture of Semantic Web Models

CONCLUSION

The preservation of intelligibility is an important requirement for digital preservation. In this paper we formalized this notion on the basis of modules and dependencies. Recall that dependencies are ubiquitous and dependency management is an important requirement that is subject of research in several (old and new emerged) areas, from software engineering to ontology engineering. Subsequently we formalized the notion of community knowledge in the form of profiles and defined intelligibility and intelligibility gaps. The notion of intelligibility gap is very important as it provides specific answers to questions of the form: what we should be record/deliver to achieve the intelligibility of digital objects by a specific community? Based on these notions we sketched a number of intelligibility-aware processes.

Recall that the preservation of the intelligibility of digital objects requires a generalization (or abstraction) able to capture also non software modules (e.g. explicit or implicit domain knowledge). A modern preservation system should be generic, i.e. able to preserve heterogeneous digital objects which may have different interpretation of the notion of dependency. The dependency relations should be specializable and configurable (e.g. it should be possible to associate different semantics to them). Focus should be given on finding, recording and curation of dependencies. For example, the makefile of an application program is not complete for preservation purposes. The preservation system should also describe the environment in which the application program (and the make file) will run. Recall the four worlds of an information system (Subject World, System World, Usage World, Development World) as identified by Mylopoulos [4]. Finally, the provision of notification services for risks of loosing information (e.g. obsolescence detection services) is important.

A proof-of-concept prototype based on Semantic Web technologies is under development. The benefits of adopting Semantic Web languages, for the problem at hand, is that although the core dependency management services need to know only a very small core ontology (defining the abstract notion of module and dependency), we can refine (specialize) the notion of module and dependency, and in this way we can capture application/domain-specific preservation requirements.

REFERENCES

[1] –OAIS: Open Archival Information System, International Organization For Standardization, ISO 14721:2003, http://public.ccsds.org/publications/archive/650x0b1.pdf (version of 11 June 2007)

[2] – Y. Tzitzikas: Dependency Management for the Preservation of Digital Information. 18th International Conference on Database and Expert Systems Applications, DEXA’2007, Regensburg, Germany, September 2007

[3] – Y. Tzitzikas, G. Flouris: Mind the (Intelligibility) Gap. 11th European Conference on Research and Advanced Technology for Digital Libraries, ECDL’2007, Budapest, Hungary, September 2007

[4] – J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis: Telos: Representing Knowledge about Information Systems, ACM Transactions on Information Systems, 8(4), 1990.

[5] - R.A. Lorie: Long term preservation of digital information, Proceedings of the 1st ACM/IEEE-CS joint conference on Digital Libraries, 346--352, 2001.