Abstract—Today’s Information Systems include many components and computational processes such as Data Warehouses, Data Marts, ETL processes, Data Analysis tools, etc. An appropriate integration of these components requires the extensive use of metadata. However the representation, consolidation, management, and access to this metadata information can cause a major overhead in the system. This paper presents an XML based Metadata Repository that consists of a centralized repository built upon standard XML technologies such as XML Schema, XSL, XQuery and XLink, coupled with an appropriate storage that proved to be a flexible, and yet inexpensive, metadata solution. Moreover it provides the means for the metamodel extensibility and easy integration into any Information System. The feasibility of this solution is documented by the description and discussion of a prototype implementation currently deployed and running in operational state as part of a complex Information System.

Index Terms—Metadata, Repository, XML, Information Systems

I. INTRODUCTION

In the context of this article metadata (data about data) is understood as any information needed to develop and maintain an Information System [1]. In complex Information Systems, the use of metadata in system components and computational processes allows to obtain a better representation and documentation of their modelling and/or behaviour as well as a better integration between the components. In the last few years, while the Information Systems complexity increased, metadata started to gain importance and acceptance, which led to the increase of the standardization efforts taking place. Many standards were created, but most of them were tied to specific areas such as Data Warehousing, solving only specific problems. At some point there were no standards or solutions that could handle all types of metadata information. In order to fill in that gap, major software companies started developing enterprise-wide metadata solutions. For their price and complexity these proved to be good solutions to deal with the metadata information of large companies, but not to solve the integration issues of Information Systems whose domain is strict and the budget is limited.

Today’s Information Systems require a flexible, light weighted metadata solution, that is easily integrated in the system architecture. In this paper the design of an XML based Metadata Repository that fulfills the above mentioned requirements is presented and discussed. This solution is flexible and extensible, because it is able to handle any kind of metadata, yet it is inexpensive. It features interfaces for communication, advanced querying, and several management tools and can be easily integrated as part of the infrastructure of any Information System. As a proof-of-concept of this solution we present a prototype implementation that is currently running, fully integrated inside a complex Information System.

This paper is organized as follows. Section 2 introduces the Metadata problematic for Information Systems. Section 3 presents our XML based Metadata Repository solution, and in section 4 a case study is presented with a prototype implementation of the solution. Finally, in section 5 the final conclusions and future work are presented.

II. METADATA FOR INFORMATION SYSTEMS

In the scope of Information Systems, metadata information is usually divided into business metadata, and technical metadata. The first is about the domain of the Information System, while the second refers to technical details of the Information System itself. Within the system metadata will be used for two purposes: documentation and operations. Therefore another categorisation can be made regarding documentation metadata or operational metadata. The first group includes information regarding the subject and structure of the Information System exclusively for documentation purposes, such as database diagrams, types of data in the system, etc. Operational metadata information is required by the Information System to model its behaviour and to define its data loading mechanisms: for instance an ETL process can be metadata driven which means that it will make use of operational metadata. Inside each of these groups the metadata information can be categorized, mainly by its type of data, and also by the entity that will make use of it. This classification of metadata is presented in [13], along with others.

With the increasing importance of metadata in the last
years, many standardization efforts were made. Metadata standardization focused initially on defining a format for interchanging metadata between systems and within components of the same system. Standards such as the CDIF (CASE Data Interchange Format) allowed interchange of some metadata models between CDIF compliant tools. This was the first standard to emphasize similarity and exchange at the metadata level. Although it has been replaced with more influential standards, its design and execution schemes are still very common. With the upcoming of Data Warehousing, metadata gained its deserved importance and prestige. Standards such as OIM (Open Information Model) and CWM (Common Warehouse Model) [2] were created specifically for the Data Warehousing design, development, and maintenance. Other initiatives were taken by several groups such as the American National Standards Institute (ANSI) for defining a standard system geared at metadata sharing, or the Institute of Electrical and Electronics Engineers (IEEE) for defining a standard to support engineering-based industries.

The common point among these standards is that all of them have a very precise target. For instance OIM and CWM are aimed at Data Warehousing and consequently cannot be used with any other purpose. Most standards are not extensible, and cover only a small subset of metadata. Such standards are supported by metadata-driven tools that tie themselves to their standardized metadata structure and format. These tools are therefore not designed for extension or update purposes.

In order to overcome this gap, general-purpose metadata management tools were developed. Aiming at the enterprise-wide metadata management, these expensive “off-the-shelf” tools support extensibility natively and allow the representation of metadata in different abstraction levels. Usually they provide a repository and a tool for managing it, adding or removing metadata information and even importing or exporting data from other tools or repositories. To guarantee access to the metadata, applications can communicate with the repository using specialized APIs such as web-services or Remote Procedure Calls (RPC). As they are enterprise-wide tools they have advanced user access procedures, featuring user views for restricting access within metadata information, and advanced architectures that support distribution. Computer Associates’ is currently the leader in the general-purpose metadata management with their “Advantage Repository” solutions. Allen Systems Group², offers a multitude of metadata management packages, and Microsoft³ offers a metadata solution integrated with its SQL Server 2000.

Complex Information Systems require the use of a metadata solution. Specific metadata-driven tools usually have restrict purposes and do not support updatability or extensibility, being only usable in specific Information Systems. On the other hand, opting for a commercial general-purpose metadata management system rarely is the right solution, since these are too expensive and wide. Only a very complex enterprise-wide Information System would make use of the features of these systems.

In the next chapter we will present our approach, which is based on XML [3] (eXtensible Markup Language) for interchanging and management of metadata. Its architecture is flexible enough to comply with either predefined standards or specific metamodels, when appropriated. Consequently, it grants the possibility of defining a custom language and supplying standardized outputs.

III. XML BASED METADATA REPOSITORY

With the upcoming of XML all major standards committees started defining their own interchange standards using XML. XML has a great potential for defining metadata information, since it is a Meta-Markup Language.

Around XML there are a set of technologies for defining, validating and transforming XML, as is the case of the XML Schema [4] where it is possible to define a structure and data types for any XML document, and XSL [5] that allows transforming the data present in an XML into any other format: XML, HTML, etc. With the advance of the research in this area, other technologies emerged such as SchemaTron [6], which allows the creation of a set of rules to which an input XML document must obey, setting assertions on elements and values, and XLink [7] which can establish relationships between XML documents by using links in an analogy to HTML hyperlinks.

With its extension capabilities, XML can be used not only as interchange format, but also as a format for storing and managing metadata. This chapter proposes the creation of a general-purpose Metadata Repository system, using XML and its associated technologies.

Since we will make extensive use of XML, the use of an XML based repository such as eXist [8] for storing purposes should be considered. Besides storing, one of the main goals of our solution is to allow a powerful querying system. Most of the XML repositories available allow the use of XQuery [9] for obtaining data from the stored information. With XQuery, complex queries can be expressed by defining the result programmatically over source XML documents.

A. Information model

From the information point of view our repository must be able to store and manage metadata information of various types. Next figure (Fig 1) presents the model accordingly to this distinction and contains a hierarchy of distinct types of information and meta-information.

---

1 http://www.cai.com
2 http://www.asg.com
3 http://www.microsoft.com
The base of the hierarchy (“Data” level) is composed by the source of all metadata. In a metadata-driven system, metadata information refers to the system’s architecture, conceptual and operational models, computational processes or data itself.

In our repository each metadata unit is called an Instance and it is represented as an XML document, for providing the required flexibility. Considering each instance XML document as a resource, it is easy to relate instances by the use of XLink. These “XLInks” will be interpreted by the repository that will enforce referential integrity between instances. The “Metadata Level” contains all the instances and their relations.

The “Meta” level contains rules and definitions for the “Metadata” level. As metadata information is usually divided into several subjects, this level provides means for defining such subjects. Each subject (called a Concept in our repository) has its own definition, provided in the form of a language that the concept instances must follow. This language is specified using an XML Schema, with a root element that defines the concept language. Nevertheless the definition of a concept can go beyond a simple language definition, since for some cases it might be necessary to specify additional rules that would not be possible to specify using only the language definition. These “instance rules” can ensure that a value of a certain element is greater than the other, among other possibilities. For such cases, besides the XML Schema, a set of SchemaTron rules or XSL style sheets are provided for defining such rules.

Metadata information is generally divided into main groups. For organizing concepts through groups we provide the notion of Metamodels. For instance, concepts regarding the architecture of the system such as databases or computational processes should be included in an “Architecture” metamodel. In order to do so, metamodels can be defined through the use of XML Schema namespaces: a set of concepts that share the same namespace automatically belong to the same metamodel.

In the effort of normalization and integration of all metamodel concepts, they can share parts of a common language, e.g. for better control, all the concepts language definitions might have to start with a header element with the authoring information. For such purposes the “Meta-Meta” level defines “concept rules” to be applied to the concepts languages and definitions. Again SchemaTron can be used to express the rules by specifying assertions over the source language definitions. However XML Schemas, which are used to define concepts, have the possibility to express the same language in several ways, by allowing the use of complex types or references to other elements. This can pose problems for defining concept rules. The solution to this problem passes by transforming the source XML Schema, into its respective normalized XML representation [10], through the encapsulation of all the complex types and references, on which the rules can be expressed. Additionally a XML Schema template, which obeys to all the defined rules, can be specified, providing a good starting point for new concept definitions.

B. Operational model

From the operational point of view, the repository must provide means for accessing instances and metamodels. The next figure (Fig 2) presents a hierarchy of access methods to metadata and metamodels.

Extractors are the first access method to metadata Information. Extractors are specified using XQuery and can obtain customizable subsets of metadata information in an XML output based on source metadata. Transforms are visions of the information obtained by the Extractors, and can be specified using XSL style sheets or Java programs, for transforming the Extractor outputs into another format. Transforms can be used either for documentation purposes, by converting query outputs into HTML or PDF formats, or for exportation purposes, by transforming the outputs into a format compliant with some metadata standards.

Extractors and Transforms are available at the “Metadata”, “Meta” and “Meta-Meta” levels, where each level defines the type of source information. Therefore it is possible to extract and transform instances in the “Metadata” level, metamodels and concepts in the “Meta” level and concept language definitions and rules in the “Meta-Meta” level.

As an example, in an Information System where the database modelling is described as metadata, an Extractor can obtain this information for a certain database, which can be used to feed several Transforms: one for generating the SQL database creation script, another for generating a data dictionary in HTML or PDF, and finally one for generating a database table diagram.

For a cleaner integration of Operational and Information models, the Information model features a built-in system metamodel containing concepts for Extractors and Transforms. The definitions of Extractors and Transforms are saved as instances of these concepts.
C. Architecture model

The full architecture model of the repository is depicted in Fig. 3. Notice that the information and operational models are now depicted together, along with several components: Version Management, Caching, Access Control, Interfaces and External Applications. Each one of these components is detailed in the following paragraphs.

![Architecture model](image)

Fig 3. Architecture model.

D. Version Management

As part of an Information System, the metadata repository should be used in all the system development phases, from requirements phase to the test and deployment phases. Therefore the repository should provide version management for all the entities inside the information and operational models. This version management is achieved by creating different version numbers upon change of existing items.

From the operational point of view, version managing is simple, since an Extractor or Transform is identified by its name or identifier and the version number.

However, from the information point of view, changing a concept definition may invalidate all its instances. To deal with this problem two different solutions can be considered: the first is to be careful upon changing the concept, ensuring backward compatibility, i.e., by ensuring that the previous instances are still valid against the new concept schema. Another solution is to change the concept definition and provide a transformation (XSL Style sheet) to be applied on existent instances updating them in accordance to the new definition.

Instance versioning, much more common than Concept versioning, has to deal with instance relationships, and a notification mechanism will be provided. Tools for impact analysis based on instance changes will also be considered.

E. Caching

Caching mechanisms can be made available for speeding up the access to the repository. The most computational demanding operations are the retrieval of Extractors and Transforms, since they may deal with large amounts of information. Such operations can be speeded up by using a caching mechanism that stores the results of requested Extractors or Transforms. Upon request the repository will lookup the cache for a stored result and if this result exists, it is promptly returned, otherwise it will be produced, and the cache will be updated. The cache is cleaned every time the source definition of the corresponding Extractor or Transform changes, and every time the source information is changed, e.g. when an instance changes. The detection of such changes is performed by a mechanism that is continuously listening for change notifications coming from the Version Management component.

F. Access Control

As metadata will be externally accessed, access control mechanisms are required. The repository provides these mechanisms, by the use of roles. A role defines the type of allowed access to the repository, e.g.: an administrative role allows full control over the repository and full access to all concepts and instances with creation, read, update, and delete permissions; a more restrictive role could be defined that only allowed reading instances from a certain metamodel.

G. Interfaces

For external applications to make use of the repository, interfaces are available for both querying and managing purposes. All these interfaces interact with the repository passing through all the layers presented above (Version Management, Caching and Access Control).

The “XQuery Interface” allows the direct querying of the metamodels using XQuery, while the “Web Service Interface” provides operations for managing the whole repository.

H. External Applications

Management functionalities are provided by the Management Console application. This application communicates with the repository through the interfaces presented above. It also features multiple visualizations for concepts and their metamodels, instance visualization and edition [5], as well as navigation between related instances.

Importers are applications whose function is to import metadata in a specific standard, transforming it, if necessary, and loading it into the repository.

Specialized editors are targeted towards specific concepts. These provide better edition of the instances than the generic editor included in the Management Console. For the edition of instances the use of XML editors would require from the end-user a certain level of familiarity with the repository and knowledge of XML. However, specialized editors allow end-users to totally abstract from the metadata representation by offering an adequate, user-friendly environment for edition and visualization.

IV. CASE STUDY

This section presents a case study where a prototype implementation of our metadata solution was implemented. This implementation is integrated with a complex Information System, which is overviewed in the next sub section.

A. Space Environment Information System overview

The Space Environment Information System for Mission Control Purposes (SEIS) is a complex Decision Support
System, which is composed by a Data Warehouse, Data Marts, an Operational Data Store, ETL processes, and Data Analysis Tools. The system architecture is depicted in Fig 4. The purpose of this system is to provide the Spacecraft (S/C) operators with a set of tools for monitoring and analyzing Space Weather (S/W) conditions (related with radiation and particles emitted by the sun) that can affect the S/C lifetime and performance. A specifically designed ETL component downloads this data from the Internet, transforming and loading it into a Data Warehouse, as well as an Operational Data Store. Two client tools are available: one for real time monitoring purposes (Monitoring Tool) [11] and other for offline analysis purposes (Reporting & Analysis Tool) [12], which uses the Data Marts.

The main source of metadata in this system is Domain Metadata. SEIS domain is composed in its majority by S/C and S/W data. Data is divided into three categories: time-series (that compose the core volume of data), events and S/C auxiliary data (e.g. Orbit Positions, Ground Station Coverage, Revolution number).

The time-series refer to S/C or Ground Stations instruments readings through time, while the events data refer to the occurrence of something during a certain period of time (e.g. S/C radiation belts entry, S/C passage through the closer point to earth, Solar Flare explosion).

The Spacecrafts and the Ground Bases themselves are also defined in the domain metadata. System specific metadata such as configuration for extraction, transformation and loading processes, database table definitions, user settings and application configuration settings are part of the system’s Operational Metadata.

This system can be considered as metadata-driven since the metadata has been used not only for documentation purposes, but and mostly drives many computational processes such as (consider Fig 4 as reference):

- on the ETL part metadata information defines the External Data Sources and the data extraction and transformation needs, which are carried out by the UDET Engine;
- on the Forecasting Model, each of the models configurations and output definitions are defined as metadata information;
- the database models (conceptual and physical levels) are stored as metadata;
- the complex loading processes from the staging area to Data Warehouse and Operational Data Storage databases are guided by metadata;
- the Reporting and Analysis Tool, uses metadata for presenting the available parameters and events to the user, to display metadata concerning each parameter and to access the Data Marts.
B. Metadata Repository prototype implementation

This section presents a prototype implementation of the Metadata Repository in the scope of the SEIS project. This prototype implementation is based on a simpler model than the one presented in this paper, and there not containing all the features presented previously. It makes use of XML and XML Schema for representing and defining metadata types, allowing metamodel extensibility, and providing a simple but powerful querying mechanism. The storage architecture is a relational database. Its diagram is depicted in Fig 5.

In this diagram there are two main tables: Concepts and Instances. The Concepts table includes every concept in the repository, where the “XSDText” field holds the concept XML Schema. The Instances table contains all the instances in the repository, where the “XMLText” field contains the instance XML. As each concept contains a set of instances, each instance contains a foreign key to its parent concept in the Concepts table. Both Concepts and Instances tables include extra documentation fields extracted directly from the XML Schema and the XML document, which can be used on querying. The relations between instances are kept in the Relations table, which are automatically updated by the managing application.

Operationally, queries (called views in this implementation) are expressed as custom XML documents containing special tags with SQL queries to be performed over the instances table. Upon request the repository will execute those queries, returning their results in the XML document. Transforms are either XSL style sheets or executables that are able to transform queries results into something else. Queries are saved in the Views table and transforms in the Transforms table.

A management tool was developed for managing the whole repository. This tool is divided in two components: one for editing and visualizing metamodels (information model) (Fig 6) and other for editing and previewing queries and transforms (operational model). A web service was created to provide basic communication between SEIS applications and the Metadata Repository. This service offers methods for downloading the results of queries and transforms and for uploading instances in the repository.

Fig 5. Metadata Repository database diagram
C. Metadata Repository integration in SEIS

Thanks to the implementation presented above all the system components in the SEIS system are metadata-driven. The ETL component uses metadata to know where to download from, transform and load the data. The modelling of all the databases is done with metadata. This means that, for instance, the Data Warehouse can be created, populated and managed using metadata.

Client applications, namely the Reporting & Analysis Tool, are fully configurable via metadata, making use of it for display, navigation and information purposes of all the resources in the system. The Metadata Repository is fully integrated in the SEIS architecture, now in its operational state, containing 35 distinct concepts, ~2000 instances, featuring ~50 views (queries) and their transforms.

As a glimpse of the type of metadata saved in the repository, an instance of the Space Weather Parameter concept is presented (Fig 7). Only a partial view of this instance is shown, displaying a set of metadata information regarding the Anisotropy Ratio (ACE) parameter. This metadata will be used by the Data Warehouse, for populating the Space Weather Parameter dimension, and by the Reporting & Analysis Tool, for data slicing, visualization and search purposes. Notice in this figure the presence of two relations: One for the Spacecraft Sub System/Sensor which measured the parameter and other to the Data Service Provider which is providing the data for this parameter.

Fig 6: Management Tool
V. CONCLUSIONS AND FUTURE WORK

The Metadata Repository solution presented in this paper eases the task of managing and using metadata in an Information System. Thanks to the use of XML, this solution features metamodel extensibility and advanced querying capabilities, while being light weighted and easily integratable in any Information System, regardless of its domain or purpose. The presented case study shows the solution’s applicability. The implemented prototype was crucial in the SEIS developing, debugging and management phases since it integrates all metadata and allows the system components to be metadata-driven, taking full advantage of the defined metadata at all times.

As for future work, the solution presented in this paper shall be refined and implemented in upcoming Information Systems.
REFERENCES

1. A. Vaduva, K. R. Dittrich, "Metadata Management for Data Warehousing: Between Vision and Reality", University of Zurich, Department of Information Technology;


10. R. Queirós, P. Leal, “XESB (XML Editor Schema Based) - Um editor para as massas”.

