EXTRACTION AND TRANSFORMATION OF DATA FROM SEMI-STRUCTURED TEXT FILES USING A DECLARATIVE APPROACH

R. Raminhos
UNINOVA – Desenvolvimento de Novas Tecnologias
Quinta da Torre, 2829-516 Caparica, Portugal

J. Moura-Pires
CENTRIA/FCT
Quinta da Torre, 2829-516 Caparica, Portugal

Keywords: ETD, ETL, IL, Declarative Language, Semi-Structured Text Files.

Abstract: The World Wide Web is a major source of textual information, with a human-readable semi-structured format, referring to multiple domains, some of them highly complex. Traditional ETL approaches following the development of specific source code for each data source and based on multiple domain / computer-science experts interactions, become an inadequate solution, time consuming and prone to error. This paper presents a novel approach to ETL, based on its decomposition in two phases: ETD (Extraction, Transformation and Data Delivery) and IL (Integration and Loading). The ETD proposal is supported by a declarative language for expressing ETD statements and a graphical application for interacting with the domain expert. When applying ETD mainly domain expertise is required, while computer-science expertise will be centred in the IL phase, linking the processed data to target system models, enabling a clearer separation of concerns. This paper presents how ETD has been integrated, tested and validated in a space domain project, currently operational at the European Space Agency for the Galileo Mission.

1 INTRODUCTION

ETL stands for "Extraction, Transformation and Loading" of data from a data source to a normalized data target, usually applied to the data warehousing / integration domains (Caserta and Kimball, 2004). In the "Extraction" phase, relevant data is identified and extracted from a data source. Since source data is usually not in a normalized format, it is required to "Transform" this data, either using arithmetic, date conversion or string operations. Finally, in the "Loading" phase the already converted data is loaded into a target system model, usually a staging area database. Considering that data may be complex depending on the domain it refers to, a domain expert is usually required during the "Extraction" and "Transformation" phases in order to identify which data is relevant and how it must be transformed in order to be correctly manipulated. In contrast, the "Loading" phase involves computer-science expertise, closely related with the target data model. In order to provide a clear separation of concerns (Dijkstra, 1972) between these two types of actions, this paper presents a different approach to ETL. The known ETL paradigm can be split into: domain ETD operations (Extraction, Transformation and Data Delivery) which require domain expertise, and IL (Integration and Loading) that require computer science operations, such that ETL = ETD + IL. By differentiating domain from computer-science operations, the development time required for an ETL solution is reduced and the overall data quality is improved by a close validation of domain data performed by a domain-expert (instead of a computer-science expert). The ETD solution proposed in this paper has been implemented and validated in a real-world application, currently operational as part of a space domain decision support system.

This paper is organized in seven sections: The current section describes the ETL problem domain

Raminhos R. and Moura-Pires J. (2007).
EXTRACTION AND TRANSFORMATION OF DATA FROM SEMI-STRUCTURED TEXT FILES USING A DECLARATIVE APPROACH.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 199-205
DOI: 10.5220/0002364201990205

and motivates for the novel ETD approach. Section 2 describes the requirements for the definition of ETD. This solution is supported by a declarative language (Section 3) and a graphical application (Section 4). Section 5 describes the integration of ETD's language and application in a complete data processing implementation, while Section 6 describes how this implementation has been successfully applied in the construction of an operational system. Finally, Section 7 draws some conclusions and presents future work.

2 REDEFINING ETL

The ETL problematic is becoming progressively less specific to the traditional data warehousing / integration domains and is being extended to the processing of textual data. The World Wide Web appears as a major source of textual data, referring to multiple domains, some of them highly complex. These files, containing textual data follow a human-readable semi-structured format. The term "semi-structured" refers to the capability to organize and present information, highlighting the different types of data available in a file (e.g. descriptive metadata area, informative header, remarks associated to the data area, numeric values or final remarks).

A traditional ETL approach, e.g. (Caserta and Kimball, 2004), follows the development of specific source code for each data source. Such approach is not the most appropriated, specially in the context of retrieving data from the World Wide Web due to the huge quantity of text files that follow heterogeneous format / presentation rules. Since the data present in the text files is closely related to the domain it refers to, it is fundamental to involve a domain-expert (usually without programming skills) in the selection, extraction and preparation of the relevant data present in the text. Thus, the classical ETL process follows a three-phase iterative procedure: (*i*) the domain expert identifies the relevant data and a set of procedures to be implemented by a computer-science expert; (*ii*) the computer-science expert codifies this knowledge and (*iii*) the solution is presented to the domain-expert for validation. This approach has several drawbacks. First, the time required for the correct processing of one file increases dramatically depending on the domain and data complexity present in the file. According to the number of interactions / corrections to the initial file processing solution, the overall time for setting up a single file may increase substantially. Second, since the logic definition for the file processing is performed by an individual outside the domain it is common that wrong assumptions are performed (e.g. data types / validation rules) that may not be detected by the domain expert during the validation phase and thus propagated to an operational environment. Third, by representing the extraction and transformation knowledge using a declarative language instead of making it hard-coded in the source code, makes this knowledge easily auditable by external domain-experts and shareable with the scientific community. Fourth, since knowledge is represented in a computable way, external analysis programs may derive some metrics regarding the language expressiveness, for future use in the language refinement and improvement.

Considering these drawbacks, this paper proposes a clear separation of ETL in ETD and IL parts. In order to accomplish ETD a set of high-level requirements has been defined: (*i*) all ETD procedures and definitions shall be represented using a high-level declarative language; (*ii*) source-code development shall not be required; (*iii*) a graphical application shall be available, making the use of the declarative language, transparent to the end user; (*iv*) data quality mechanisms shall be available incrementally in all ETD steps; (*v*) the solution shall focus mainly on the Extraction and Transformation steps. A generic interface shall be proposed for data delivery since the Loading process is highly coupled with the target database / application that shall receive the data; (*vi*) after performing the ETD specifications for the base file, its generality shall be tested with a set of samples of the same type.

For accomplishing such requirements, a solution based on a declarative language and a graphical application is presented in the next two sections.

3 THE FFD DECLARATIVE LANGUAGE

The proposed ETD solution relies on declarative definitions that identify which operations are required during the ETD process, instead of specifically implement this logic at code level. These declarative definitions are stored in ETD scripts named File Format Definition (FFD). FFD contents are directly dependent on the text file format, such that a one-to-one association exists between a text file and a FFD. XML Schema and XML technologies have been selected for the definition of the declarative language and FFD instances, respectively, since they are highly known World Wide Consortium (W3C) standards for which exist computational efficient tools. The FFD language is divided into eight distinct sections:

General Information: Identification metadata as name, description and authoring.
Versioning Control: Text files formats may change without any kind of previous notification. To resolve this issue, each FFD contains a pair of date values defining a time frame in which the FFD is valid.
Processing: Thread priority to be applied during the ETD process.
Sectioning: The first step for the extraction of data consists in the partition of the text file into non-overlapping sections that identify different areas of data within the text file. Generally these sections are easily identified since they usually share some common property (e.g. all line starting with a given prefix or following a start / end delimiter condition). Each section is identified by a name and can be either *delimited* (where two boundary conditions determine the beginning and end of the section) or *contiguous* (defined by a common property, shared by a contiguous set of text lines). Delimited sections can be defined through absolute conditions as "file start", "file end", "line number" or the first line that "starts", "contains" or "ends" a given string pattern. Besides absolute conditions, delimited sections can also be defined using relative conditions as "Start section after previous section end" or "End section before next section start". Contiguous sections can be defined through one of three conditions: group of lines that "start", "contain" or "end" a given string pattern. A simple BNF grammar is present in Figure 1 for the proposed sectioning scheme.

```
Sectioning-Spec -> (Section-Spec)*
Section-Spec -> (Contiguous | Delimited)
Contiguous -> LinesStartingWith(pattern) |
              LinesContaining(pattern)|
              LinesEndingWith(pattern)
Delimited -> (Start, End)
Start, End -> Relative |
              Line(number) |
              LineStartingWith(pattern) |
          LineContaining(pattern) |
          LineEndingWith(pattern)
Relative -> AfterPreviousSectionEnd |
           BeforeNextSectionStart
```

Figure 1: A BNF grammar for sectioning.

A set of validation rules can also be imposed to each section in order to detect (as early as possible) a change in the file's format: if the section can be optional, minimum and / or maximum number of lines present in the section, existence of a given pattern in the section start, middle or end.
Fields: Contains the definition for all fields that can be extracted from the contents of a section. Two types of fields are available "single fields" and "table fields". Single fields refer to individual values present in a text file. These can be captured in one of two ways: specifying prefix and / or suffix values or through a regular expression. Table fields contain one or more table columns, which can be defined through fix-width length, a regular expression or by specifying a column delimiter character that separates the columns. Both single and tabular fields can be defined from the start of the section or given a specific offset of lines within the section.

Data quality mechanisms are available for both types of fields. To each single value or table column it is possible to associate a data type, a set of validation rules (e.g. minimum / maximum numeric value or text length) and a missing value representation (usual in scientific data files).

Independent from the type of section and field definition, both object specifications are represented internally as regular expressions. This representation is transparent to the end-user that only requires knowledge on a set of gestures for interacting with the graphical application. Using regular expressions increases substantially the text processing performance due to their pattern matching capabilities and efficient supporting libraries.
Transformations: Contains a set of transformation pipelines (i.e. a directed set of computational elements handled in sequence), to be executed, transforming raw data into a suitable format for data delivery. Two types of transformations are available: column / single field and table oriented. In the first case the transformation will affect only one table column / single value (e.g. append a string to the end of each value of a selected column). In the second case the transformation will affect multiple table columns (e.g. deleting a set of rows from a set of table column given a matching criteria).

Each transformation pipeline can be mapped to a direct acyclic graph. Each start node from the pipeline refers to an extracted field, while the remaining nodes represent transformation operations. Connections between transformation nodes represent that an output of a source transformation node is being used as input by a target transformation node. Since the required transformations are closely related with the domain and structure in which data is presented, new transformations can be inserted as needed, according to a plugin architecture. Some examples of transformations are: *AppendConstant*, *CreateDate*, *DateConvert*, *DeleteStringRows*, *Distribute*, *Duplicate*, *GetElement*, *Join*, *Map*, and *Split*.
Data Delivery: A Data Delivery consists in a XML file containing as contents some descriptive metadata header and a data area organized in tabular format. In order to specify a data delivery, two types

of information must be available: (*i*) **Structural Metadata:** Structure definition of the data delivery XML file. Specifies multiple pairs (*field name, data type*) to be used in the data delivery's interpretation phase in the IL layer. The order in which data is delivered is also specified as metadata; (*ii*) **Data References:** The actual data values (extracted fields or transformation outputs) to be delivered.

Data Delivery acts as a generic interface between ETD and IL components. Depending on the specific file format, data contents and the amount of data to be delivered, during runtime a single data delivery may be split into multiple packages in a transparent way, due to performance issues.

For each text file and associated FFD, an ETD engine shall perform the following sequence of operations: (*i*) section splitting; (*ii*) field extraction; (*iii*) field transformation and (*iv*) data delivery.

4 FFD EDITOR

The FFD Editor is a graphical application for the creation, edition and debug of FFDs. With this graphical interface the FFD XML language is made transparent to the domain-expert. The creation of a new FFD is based on annotations over an example text file, following four main phases (that may be iterative if required): Extraction, Transformation, Data Delivery and Validation.

4.1 Extraction

Specific user interaction schemes – gestures – applied to the sectioning and field definition enable a simple interactive way of human-machine communication. All file sections are initially created by a "default section creation" gesture, where the user selects a set of text lines that are marked as a section. A "default section" is delimited by nature and has its start and end delimiters following a "line number" condition based on the first and last line selected by the user. Dragging the section boundaries interactively, it is possible to link a section boundary either to a "file start / end" or "section start after previous section end / section end before next section start" relative conditions. Sectioning can also be performed via specific wizards for defining pattern dependent conditions like "line starting / containing / ending with a pattern string" or "lines starting / containing / ending with a pattern string", for delimited and contiguous sections respectively. During the wizard, the user may express a set of validation rules that the defined sections must comply in order to be valid. Based on the sectioning type (delimited or contiguous), different types of symbols are presented to the user in order to identify pictorially the section's characteristics. Independent from the section type, clicking on the section symbol will highlight the corresponding text at the base text file. Figure 2 presents the sectioning symbols for a delimited upper boundary section: *a)* line number; *b)* relative to previous section; *c)* relative to file start; *d)* line starting with a string pattern; *e)* containing a string pattern and *f)* ending with a string pattern. Symbols are also applied to contiguous sectioning: *g)* lines starting with a string pattern; *h)* containing a string pattern and *i)* ending a string pattern.

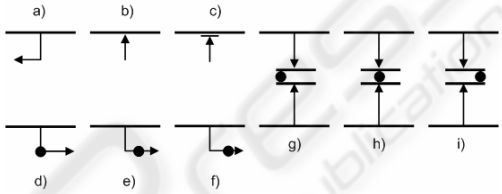

Figure 2: Example of delimited and contiguous sectioning symbols.

The definition of fields, either single value or tabular, is initiated by a "default field creation" gesture and completed via a specific wizard, since field definition is usually more complex than sectioning definition. For each single value or table column the user may define a data type, a missing value representation and a set of validation rules (e.g. "Maximum value" or "Minimum value").

Internally, all gestures are codified as regular expressions (some of them quite complex), before performing the file sectioning and extraction steps. Regular expressions enable a good performance, due to their pattern matching capabilities that can be applied to the entire text, when compared to traditional string operations that would require breaking the text into a set of lines and deal with each line individually.

4.2 Transformation

The transformation area (Figure 3) follows a classical approach based on possible multiple transformation pipelines that are represented as graphs (2). Each graph node represents a transformation that is part of an internal library, depicted in the transformations toolbar (1). Having selected a specific transformation in a graph it is possible to verify its correctness by visual inspection of its data inputs and outputs (3).

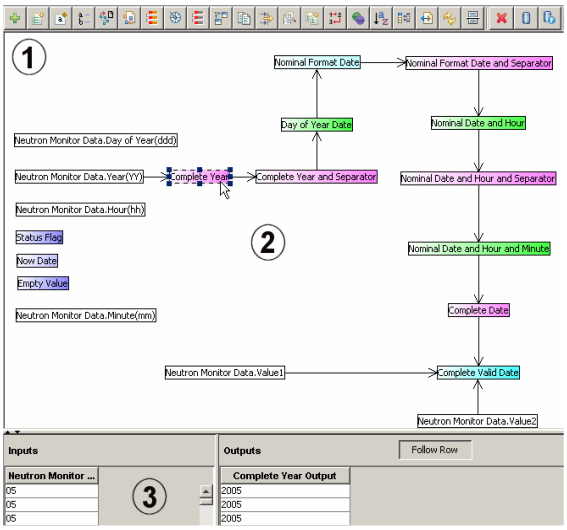

Figure 3: FFD Editor's "Transformation" step.

Transformations require specific "tuning" metadata defined by the user (e.g. for an *appendConstant* the user must define the constant to be appended and if this shall be placed as a prefix or suffix). For example, in Figure 3 an *appendConstant* transformation has been selected, appending to an extracted field the "20" value as prefix, forming a four digit year value that is placed in the "Complete Year Output" column. Since transformations are highly dependent on the domain data structure, these have been implemented as independent plugins, making the transformation library easily expanded. Associated to each transformation plugin, exists a descriptive metadata file where inputs and outputs data types can be defined enabling some data quality control during the transformation step.

4.3 Data Delivery

The first step in defining a data delivery consists in selecting the parameters to which the data delivery refers. Depending on the parameters nature, a structure for the data delivery must be defined in the form of (*field name*, *data type*) pairs, where the user will drag-and-drop either references to extracted fields or transformation outputs. Figure 4 depicts the "Data Delivery" panel: (*i*) **Extract Tree:** A tree with all the extracted fields in the "Extract" step; (*ii*) **Visible Outputs Tree:** A tree with all the transformation outputs in the "Transform" step; (*iii*) **Data Delivery Tree:** A tree with all the created data deliveries; (*iv*) **Toolbar:** Creation, edition and saving operations for a data delivery; (*v*) **Template Area:** A tabular template for the data delivery definition. While the left column contains the column name for the data delivery field the right column receives a data reference "drag-and-drop" by the user; (*vi*) **Preview Area:** Contains a tabular preview of the data to be delivered.

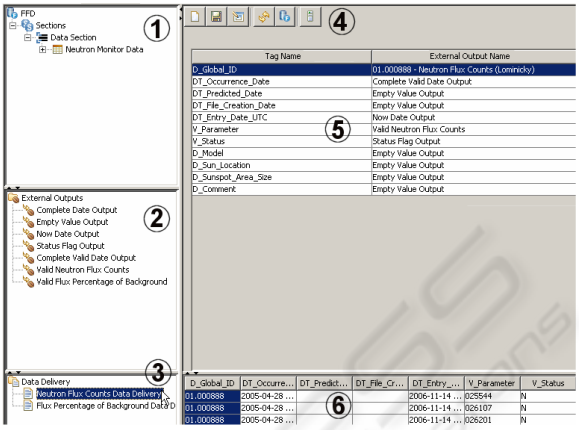

Figure 4: FFD Editor's "Data Delivery" step.

4.4 Validation

As a final step the user should apply the FFD definition to a set of other files in order to verify if the definition is general enough. If an error is detected, then the user can correct the FFD (iteratively if required), analyzing the impact of the change over the ETD steps. Otherwise, the FFD can be saved locally into the file system or uploaded directly to a Metadata Repository.

5 DATA PROCESSING MODULE

In this section, a general architecture for a Data Processing Module (DPM) is proposed that strongly supports the declarative assertions present in the FFD language (Figure 5). Although the declarative language and FFD Editor are the key technologies presented in this work, in order to have an operational data processing solution, two other software components must be introduced: (*i*) **File Retriever (FR) engine:** responsible for the acquisition of data files from external data service providers and (*ii*) **DPM Console:** A graphical tool that enables the monitoring and control of download and processing actions. The application enables to manage the downloaded files and checking all logging information. The FR and ETD engines execute continuously and are responsible for the download / ETD chain. All metadata required by the FR or ETD engines is stored in a centralized Metadata Repository (Ferreira and Moura-Pires, 2007).

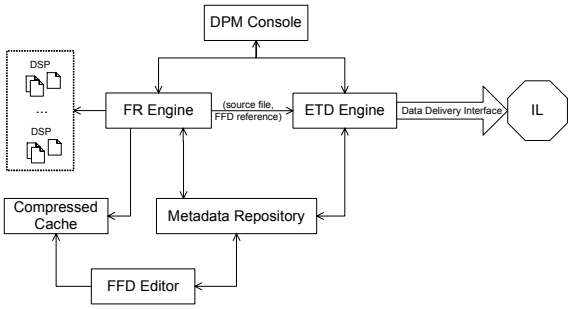

Figure 5: Data Processing Module Architecture.

The FR Engine downloads text files according to a schedule and places them in a compressed directory for backup purposes. For each Data Service Provider, the user can specify the type of connection for reaching the Data Service Provider. Four types of connections are available: HTTP, FTP, Web Service or Database (through a JDBC connector). Metadata is required depending on each connection specifics (e.g. "username", "password" or "database name").

Each Data Service Provider hosts multiple data files, commonly known as Provided Files. For these files the user can define the source file path / SQL query / or web service arguments (depending if the related Data Service Provider has a HTTP or FTP, database or web service connection, respectively). For each Provided File it is also possible to specify the path for the target file to be stored in the local cache and which type of schedule shall be used for retrieving the file: by user request; from a specific set of dates; every X seconds; at a specific (minute), (hour / minute), (day / hour / minute) or (month / day / hour / minute) tuple. Further, each Provided File has a reference to the FFD associated for processing that type of file. Finally, each Provided File may have a set of relations to a pool of dedicated ETD engines for processing. If no ETD engine instance is specified, text files shall be processed in a round-robin fashion through all the ETD engine instances that have been declared in a Metadata Repository.

The ETD Engine is responsible for the actual ETD processing. After download, FR sends to the ETD engine the text file contents and a FFD reference to be applied. If an error is detected (e.g. a file format has changed causing a data type violation), the system administrator is notified by email, receiving in attach the file that caused the error and the FFD that raised the exception. After processing the file, the ETD engine delivers all data to the IL layer using the generic Data Delivery interface, where all processed data is delivered in XML format.

Finally, the FFD Editor application enables the creation, debug and test of FFDs, as well as their submission to a supporting Metadata Repository, for posterior use by the ETD engine.

Both the ETD engine and FFD Editor component share the same code for processing each provided file. This way the results accomplished while using the FFD Editor application will be the same during the processing phase with the ETD engine. In order to improve scalability and performance it is possible to have multiple ETD engines that serve one or more FR engines. With this file-partitioning scheme, load balancing is attained during processing.

Further details and screenshots can be found at http://centria.di.fct.unl.pt/~jmp/DMP.

6 SESS CASE STUDY

The space domain term "Space Weather" (S/W) (Daily, 2002; Schmieder, Vincent et al., 2002) can be defined as the combination of conditions on the sun, solar wind, magnetosphere, ionosphere and thermosphere. Space Weather, affects not only Earth's environment, but specially all Spacecraft (S/C) systems orbiting the planet. The degradation of solar panels is an example of a S/W effect.

The integration of both near real time and historical S/W and S/C data for analysis, is fundamental in the decision-making process during critical Spacecraft control periods and in order to extend the mission life-time to its maximum. Analysis of the current solar activity together with the internal S/C sensors measures may force / prevent the execution of manoeuvres in order to protect S/C equipments or even human lives.

The Space Environment Support System (SESS) (ESA, 2006) is a multi-mission decision support system, capable of providing near real-time monitoring (Moura-Pires, Pantoquilho et al., 2004) and visualization, in addition to historical analysis (Pantoquilho, Viana et al., 2005) of S/W and S/C data, events and alarms. The main goal of the system is to provide S/C and S/W data integration. The Data Processing Module is responsible for the download and ETD processing for each text file containing scientific data made available by the different public and / or private data service providers. Communication with the data service providers is performed via well-know protocols like HTTP and FTP or through a Web-Service interface. After processed, data is delivered to the Data Integration Module that comprises two databases: an Operational Data Storage for the parameters values within the last five days (sliding-window) and a

Data Warehouse for storing historical data. Data present in the Operational Data Storage can be monitored through the Monitoring Tool while historical data can be analysed through the Reporting and Analysis Tool. Crosscutting to all system components is a Metadata Repository (Ferreira and Moura-Pires, 2007) that contains all domain and technical metadata.

In the scope of the SESS project a set of 63 Provided Files have been selected from a total of 10 different Data Service Providers. For each Provided File a specific FFD has been created with the FFDE tool. At nominal operational execution DPM downloads around 4000 files per day representing an average storage load of 140 MB of text per day (average of 35KB per text file). The performance of the ETD engine based on FFDs declarative language presented very good results, considering the 5 minute "near real-time" requirement imposed by ESA for performing all ETD tasks. Processing times range from a 1 second average for small files to 30 seconds average for 3.5 MB telemetry files. The overall average per file processing is around 2 seconds.

7 CONCLUSIONS AND FUTURE WORK

This paper presented a new approach to the ETL problem, based on an ETD declarative language. This novel approach supports a clear separation of concerns between the Extraction and Transformation steps (domain related) from the Loading step (technical related). A new Data Delivery interface is proposed for abstracting all details regarding the target application that will receive the processed data. A graphical application is proposed for interacting with the domain expert in order to define an ETD script based on graphical interaction, making the supporting language transparent to the end user. The data processing solution has been implemented, tested and validated in the scope of the SESS system and is currently operational for the Galileo Mission. Feedback from the end users has been positive regarding user interactivity, language expressiveness, scalability and performance for the ETD solution.

As future work we intend to keep improving the user interaction with the FFD Editor application and explore two new features. First, enable the FFD Editor application for proposing a possible transformation pipeline (P) based on a user declaration of an output result (O) - including the field value and data quality metrics like data type and validation rules – given the available input set (I) as the defined transformation library (L), such that P(I, L) = O. Second evaluate the practical impact of including a filtering cache for dealing with repeated input data prior to the ETD process execution (e.g. a duplicated text line or database record). With the current solution all data filtering must be performed either by the source data service provider, that guaranties by itself no data repetition, or by processing all data (even repeated) and perform the data filtering at the IL layer, resulting in unnecessary processing.

REFERENCES

Caserta, J. and R. Kimball (2004). The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data, John Wiley & Sons.

Daily, E. (2002). Space Weather: A Brief Review. SOLSPA: The Second Solar Cycle and Space Weather Euroconference, Napoli, Italy.

Dijkstra, E. (1972). Notes on Structured Programming. A. Press.

ESA. (2006). "Space Environment Support System for Telecom/Navigation Missions (SESS)." from http://telecom.esa.int/telecom/www/object/index.cfm?fobjectid=20470.

Ferreira, R. and J. Moura-Pires (2007). Extensible Metadata Repository for Information Systems and Enterprise Applications. ICEIS 2007 - 9th International Conference on Enterprise Information Systems, Funchal, Portugal.

Moura-Pires, J., M. Pantoquilho, et al. (2004). Space Environment Information System for Mission Control Purposes: Real-Time Monitoring and Inference of Spacecrafy Status. 2004 IEEE Multiconference on CCA/ISIC/CACSD, Taipei, Taiwan.

Pantoquilho, M., N. Viana, et al. (2005). SEIS: a decision support system for optimizing spacecraft operations strategies. IEEE Aerospace Conference, Montana, USA.

Schmieder, B., B. Vincent, et al. (2002). Climate and Weather of the Sun Earth System: CAWSES. SOLSPA: The Second Solar Cycle and Space Weather Euroconference, Napoli, Italy.