ASPECT-ORIENTED ANALYSIS APPLIED TO THE SPACE DOMAIN

André Marques, Ricardo Raminhos, Ricardo Ferreira, Rita Ribeiro, Sérgio Agostinho
UNINOVA – Instituto de Desenvolvimento de Novas Tecnologias 2829-516 Caparica, Portugal

João Araújo, Ana Moreira
CITI/Dep. de Informática Universidade Nova de Lisboa 2829-516 Caparica, Portugal

Keywords: Early Aspects, XML technologies, metadata, analysis, specification, space domain.

Abstract: This paper presents an aspect metadata approach, which has been developed in the scope of the "Aspect Specification for the Space Domain" project. This approach is based on XML and XML Schema technologies, enabling a rigorous knowledge representation. The proposed approach has been applied to a real complex system, the "Space Environment Support System", enabling a comparison and evaluation between the proposed approach and the "traditional" requirements analysis methods used during the development of the original version of the system. This paper presents a full description of both the identified metadata concepts and their relationships. The metadata concepts and associated instances have been stored in a Metadata Repository that provides simple navigation facilities between concepts. The Metadata Repository also enables the automatic generation of documentation.

1 INTRODUCTION

Separation of concerns refers to the ability of identifying, modularizing and possibly reusing system concerns (e.g., functionalities and global properties) in a software program. Crosscutting concerns are usually identified at the final stages of software development, namely at the implementation level using aspect-oriented techniques. However, it is good practice to identify the system's crosscutting behaviour as soon as possible, addressing possible conflicts between concerns as early as the requirements phase, not postponing this task to the latter stages of development.

Different approaches have been proposed for the representation of concerns structure ((A. Rashid, 2003), (Grundy, 1992), (Lamsweerde, 2001)), including metadata (R. Ferreira, 2005). This work proposes a refinement to the "early aspects" structural metadata (R. Ferreira, 2005), using XML and XML Schema technologies that provide a rigorous representation. A full description of the identified metadata concepts is presented, as well as the relations between them. The approach is applied to a real case study in the space domain, the "Space Environment Support System", currently operational at the *European Space Technology Centre (ESA/ESTEC)*.

Within the scope of "early aspects" a multidimensional approach (I. Brito, 2004) has been used for the identification of crosscutting concerns. This multidimensional approach proved to be a better way of identifying user needs, when compared with the traditional bi-dimensional approaches based on a dominant decomposition using viewpoints, use cases or goals (e.g. (A. Rashid, 2003)).

This paper is organized as follows. Section 2 presents the ASSD project. Section 3 presents the concepts specification. Section 4 introduces the "Space Environment Support System" case study and identifies the advantages and drawbacks in applying the proposed methodology and concepts to a complex operational system. Section 5 discusses some related work. Finally, Section 6 provides an overview evaluation of the current proposal and finishes by drawing some conclusions.

2 ASPECT SPECIFICATION FOR THE SPACE DOMAIN (ASSD)

Various programming paradigms have increased the modularity of software, but there are still some

Marques A., Raminhos R., Ferreira R., Ribeiro R., Agostinho S., Araújo J. and Moreira A. (2007).
ASPECT-ORIENTED ANALYSIS APPLIED TO THE SPACE DOMAIN.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 71-79
DOI: 10.5220/0002385800710079

properties that classical software development methods are unable to modularize. These properties cut across several base (code) modules, producing scattered and tangled code that is difficult to maintain and evolve. This is called a crosscutting concern. A match point is composed by a set of concerns that need to be composed together. One of the concerns plays the role of base concern on which the behaviour of the remaining concerns needs to be weaved.

The main objective of the ASSD project is to apply an Aspect-Oriented methodology[1] to the ground segment of space domain projects, at the early stages of software development. Taking as a case study an already (successfully) deployed project using a traditional requirements methodology, the consortium addressed how "aspects" could be employed in order to reduce system complexity, development time and improving maintenance and evolution tasks. The development of a "meta-aspect" repository architecture was also proposed, for storing the aspects specification identified in the case study, intended for reuse in further projects. For further validation, a second application shall be tested by the industrial partners of the project.

The ASSD project used and adapted two main technologies, namely the Metadata Repository (see Section 2.1) for storage purposes and the Aspect-Oriented Requirements Analysis method presented in (I. Brito, 2004),(I. Brito, 2006), for the early stages of software development.

The advantages of a repository for storing metadata were already assessed in the scope of the Space Environment Information System for Mission Control Purposes (SEIS) project (M. Pantoquilho, 2004). However, the hybrid combination of the two technologies (aspect methodology and metadata repository) as proposed in the ASSD project was never attempted.

2.1 Metadata Repository

The Metadata Repository is a by-product of the SEIS project developed for ESA and currently operational at ESOC. The SEIS project, like many other large projects, is composed of many sub-systems, where every module has a need for sharing metadata. The Metadata Repository is used as a technology infrastructure where all metadata is stored in a structured way, and accessible to other system components.

[1] Methodology enabling modularization of crosscutting concerns.

The Metadata Repository enables the management of the metadata in a coherent way. Coherency, and consistency, is achieved due to the absence of replication and available versioning support. To increase the flexibility of the metadata structure, XML Schema documents are used to define and validate different types of metadata information.

By including all ASSD metadata in a specialized Metadata Repository, the validation effort (both structural and referential) is performed at the server side and made transparent to the client applications. Further, the Repository enables metadata reutilization (especially at the concern level), navigation and traceability between concepts. Finally, the Metadata Repository is able to automatically generate documentation, handling ASSD metadata in order to create a set of outputs according to the proposed ASSD methodology.

A specific terminology has been used to identify each type of information as a layer. Layer **M0** represents the **objects** to be described using metadata in the above layer. These objects are part of the host system and are not to be stored in the Metadata Repository (e.g., a record in a database table). Layer **M1** represents metadata information about M0. In the Metadata Repository context, this information is called instances and is stored as XML documents (e.g., "security", "concern"). Layer **M2** defines the format and rules for each type of metadata to be stored in the M1 layer. These are called **concepts**. A concept is a definition of a structure to specify a type of metadata and is represented as an XML Schema (e.g. the "concern" concept). This paper focuses on the metadata level M2 for the definition of concepts. Instances (M1 level) are handled by the Metadata Repository while applying the proposed ASSD Aspect-Oriented methodology.

3 CONCEPT SPECIFICATION

This section introduces the concepts required for the representation and logic structure of early aspects. Seven main structures are proposed: *System, Concern, Stakeholder, Decomposition Node, Stakeholder Requirement, System Requirement* and *Test Case*. All concepts are defined through templates with three columns: attribute name, cardinality and description.

Considering the authors' previous work, System, Concern and Stakeholder structures have been extended with new fields, and system related information has been gathered in the system concept (in order to make Concern and Stakeholder independent). The remaining concepts are proposed for the first time in this paper.

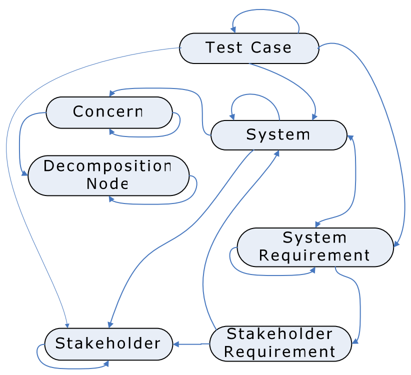

Figure 1: Concept's main relations.

Figure 1 depicts the main relations for the proposed concepts. Each arrow represents how the relation scheme is defined as its navigability. Analysing the relations some concepts are independent from the others, such as *DecompositionNode*, Stakeholder and *Concern*. The *System* concept, which represents a collection of interacting and interrelated elements, links together concepts like *Concern*, *Stakeholder*, *StakeholderRequirement* and *SystemRequirement*. This way, system dependent information is stored in the main *System* concept. All references linking the *System* concept to the independent concepts are intended to promote instance reusability.

The following section presents the concepts required for structuring all the information regarding the Aspect Specification analysis. Such specification is important individually for each concern and as a whole for the definition of a supporting infrastructure for early-aspects storage.

3.1 System

The structural representation for the *System* template is depicted in Table 1, describing the system in terms of the context in which it is applicable and defining the consequences resulting from its usage.

Table 2 presents an instance (i.e., an example) of the System concept regarding a simple log component.

Table 1: The System template.

Attribute name	#	Description
Name	1	The *System* name.
Description	1	Short description of the intended behaviour of the system.
Aliases	0..N	Additional names (synonyms) that also identify the system.
URLs	0..N	List of URL references valuables in the system understanding.
Parent reference	0..1	Points to the parent system of this system, enabling hierarchy between systems.
Context	1	Description of the environment in which the problem and solution occur and for which the solution is desirable.
Known uses	0..N	Presents a set of "real world" applications that implement the system, providing some "warranty" over its quality.
Motivations	1..N	Description of problematic situations (examples).
Similar systems	0..N	List of similar systems, enabling a navigation mechanism between them.
Stakeholders	1..N	List of stakeholders involved in the system that have a direct or indirect influence in the system.
Concerns	1..N	List of concerns involved in the problem modelling.
Keywords	0..N	List of keywords of this system for searching purposes.
Extension	0..1	Additional extensions to the definition of the system.

Table 2: The LOG's System concept instantiation.

Attribute name	Instantiation		
Name	Log		
Description	The Log component registers all the task activities from SESS components.		
Aliases	Logging	Log	
URLs	None.		
Parent reference	Data Processing Module		
Context	While Data Processing Module applications are executing, actions of these applications are registered by the LOG component.		
Known uses	Information Systems		
	Critical Systems		
	Banking Systems		
Motivations	Data traceability.		
	Need to show data in simple and structured way.		
Similar systems	*None.*		
Stakeholders	**Name**	**Role**	
	Developer	Developer	
	Administrator	Administrator	
Concerns	**Concern Name**	**Stakeholder**	**Stakeholder Priority**
	Logging	Administrator	Very Important
		Developer	Important
	Persistence	Administrator	Very Important

		Developer	Important
Keywords	Log	Catalogue	Event
Extension	*None.*		

The *Stakeholder* attribute (the *Stakeholder* concept is defined in Table 5) contains a list of stakeholders involved in the system that have a direct or indirect influence (e.g. "Administrator", "Application", "Client", "Company"). Since the ASSD project will provide a client tool allowing the possibility of generating UML use cases, there is still the possibility to define if a system's stakeholder is an actor or not, i.e., if a given Stakeholder will be part of a UML use cases diagram. The *Concern* attribute (the *Concern* concept is defined in (Table 3) contains the list of concerns that compose the system. For each concern, a list of needed or requested concerns is available. A list of stakeholders is associated to each *Concern*, where each stakeholder has a priority towards the concern: "Very Important", "Important", "Medium", "Low", "Very Low" or "Don't Care". Since a concern results from a system requirement, a list of system requirements (the System Requirement concept is defined in (Table 8) that need this concern can be defined. A concern can be decomposed through a decomposition node and for each decomposition, a value of this analysis is given: "Satisficed[2]", "Weakly satisficed", "Undecided", "Weakly denied", "Denied", or "Conflict" (L. Chung, 2000). Finally, and for extensibility purposes, a *Mapping* of each concern to an artefact can be made at a later stage of software development.

3.2 Concern

Table 3 presents the template for describing a concern, while Table 4 presents an instance example. The *Classification* attribute identifies the concern according to its type, e. g. functional or non-functional (Sommerville, 2004): "Delivery", "Efficiency", "Ethical", "External", "Functional", "Implementation", "Interoperability", "Legislative", "Organisational", "Performance", "Portability", "Privacy", "Product", "Reliability", "Safety", "Space", "Standards", or "Usability".

Table 3: The Concern template.

Attribute name	#	Description
Concern name	1	The name of the concern.
Description	1	Short description of the intended behaviour of the concern.
Parent reference	0..1	Concern related with the current one. This reference points to the parent concern of this concern, permitting hierarchy between concerns.
Sources	0..4	States the concerns origins: stakeholder, concern, catalogue and/or system.
Classification	1	This attribute helps the selection of the most appropriate approach to specify this concern.
Contributions	0..N	List of concerns that are affected positively/negatively[3] by this concern.
Similar concerns	0..N	List of similar concerns, enabling a navigation mechanism between them, not necessarily from the same *System*.
Decomposition node	0..1	Reference to the root of the Decomposition Node of the current concern.
Keywords	0..N	List of keywords of this concern for searching purposes.
Extensions	0..N	Additional extensions to the definition.

[2] Satisfice: decide on and pursue a course of action satisfying the minimum requirements to achieve a goal.

[3] Negative contributions are analysed to identify potential conflicts between concerns

Table 4: The Configurability's Concern instance.

Attribute name	Description	
Concern name	Configurability	
Description	The capability of defining the configuration and behaviour of an application.	
Parent reference	*None.*	
Sources	Stakeholder	
Classification	Implementation	
Contributions	**Concern Name**	**Contribution Value**
	Client Portability	+
	Modularity	+
	Server Portability	+
Similar concerns	*None.*	
Decomposition node	*None.*	
Keywords	Configuration, Extensibility.	
Extensions	*None.*	

3.3 Stakeholder

Table 5 presents the template for describing a stakeholder.

Table 5: The Stakeholder template.

Attribute name	#	Description
Stakeholder name	1	The name of the *Stakeholder*'s instance.
Description	1	Short description of the intended behaviour of the Stakeholder.
Parent reference	0..1	Stakeholder related with the current one. This reference points to the parent stakeholder of this stakeholder, allowing a hierarchy between stakeholders.
Contact information	0..1	General stakeholder information: address(es), email(s), phone and fax numbers..
Keywords	0..N	List of keywords of this concern for searching purposes.

3.4 Decomposition Node

Table 6 presents an example of the *Decomposition Node* concept. The *Is operationalization* attribute determines if the current decomposition provides operations, processes, data representation, structuring, or specific constraint. That is, it informs if it provides a concrete functional mechanism to accomplish the decomposition or not. This is known in the literature as *operationalization* (L. Chung, 2000). The *Decomposition* contributions attribute refers to the nodes that contribute with a given value ("Make", "Help", "Unknown", "Hurt" or "Break" (L. Chung, 2000)) to the current *Decomposition*; yet a justification for the choice of the *Decomposition Contributions'* value can be applied to this. The *Decomposition operator* attribute describes which logical operator (i.e., "AND", "OR") decomposes the current node. A justification can be applied to support or deny the way target components are selected.

Table 6: A Decomposition Node example.

Attribute name	Description	
Name	Encryption	
Description	The capability to make the contents of a message or file unintelligible to anyone not authorized to read it.	
Is operationalization	False	
Decomposition contributions	*None.*	
Decomposition operator	**Or decomposition**	
	RSA	3DES
Keywords	Security	Cryptography

3.5 Stakeholder & System Requirement

The Stakeholder Requirement template is presented in Table 7. The *Classification* attribute allows to choose from an enumeration of values (Sommerville, 2004): "Delivery", "Efficiency", "Ethical", "Functional", "Implementation", "Space", "Interoperability", "Performance", "Safety", "Portability", "Privacy", "Reliability", "Standards", or "Usability", whereas the *System Reference* attribute is mandatory in the way that it directly points to the system that owns the current *Stakeholder Requirement*. The *Priority* attribute defines the priority of the current Stakeholder *Requirement* in the scope of the *System*, which is referred in the *System Reference* attribute. The MoSCoW rules (Stapleton, 1997) were used for this purpose.

Table 7: The Stakeholder Requirement template.

Attribute name	#	Description
Name	1	The *Stakeholder Requirement* name.
Description	1	Short description of the intended behaviour of the Stakeholder Requirement.
Source stakeholder	1	The stakeholder that is responsible for elicitation of this requirement.
Classification	1	Classification of the requirement.
System reference	1	The system that is referenced by the current stakeholder requirement.
Priority	1	Priority of the *Stakeholder Requirement.*
Keywords	0..N	List of keywords for searching purposes.

Table 8 presents the template for describing a System Requirement. The *Stakeholder Requirement* attribute holds a list of all stakeholder requirements that complies with the current *System Requirement*. The *System Reference* attribute is mandatory in the way that it directly points to the system that owns the current *System Requirement*. The *Type* attribute describes the category of the *System Requirement* that can be either “Private”, “Internal” or “Public”.

Table 8: The System Requirement template.

Attribute name	#	Description
Name	1	The *System Requirement* name.
Description	1	Short description on the intended behaviour of the System Requirement.
Parent reference	0..1	*System Requirement* related with this one. This reference points to the parent System Requirement of this System Requirement, enabling hierarchical structure between System Requirements.
Stakeholder requirement	0..N	List of *Stakeholder Requirements* that complies with the current requirement.
System reference	1	Reference to the system that owns the current system requirement.
Type	1	Type of the requirement.
Classification	1	Classification of the requirement.
Priority	1	Priority of the *System Requirement*.
Keywords	0..N	List of keywords of this *System Requirement* for searching purposes.

3.6 Test Case Concept

The *Test Case* description is presented in Table 9.

Table 9: The Test Case template.

Attribute name	#	Description		
Name	1	The name of the *Test Case*.		
Description	1	Short description of the intended behaviour of the test case.		
Expected tester profile	1	The profile of the tester that shall perform the test (e.g. “Administrator”, “User”).		
Test method	1	Specifies the test method to be applied: “Testing”, “Code Inspection” or “Review”.		
System reference	1	System to which this test case refers to.		
Objective	1	**Test Objective Description**		Description of the test case objective.
		Tested Requirements		A set of references for the tested requirements.
Test dependencies	0..N	References to the tests that shall be executed before this test case.		
Test details	0..N	**Pre-requisites**		Requisites required to perform the test.
		Inputs		Data required to perform the test.
		Procedure		Description of the test to be performed.
		Expected Results	**Outputs**	Generated outputs description.
			Pass Criterion	Condition for the test to be successful.
Test case executions	0..N	**Test Campaign**		Type of test[4].
		System Version		System version that was tested.
		Tester	**Name**	Tester name.
			Profile	Tester profile.
		Result		Test Result
Keywords	0..N	List of keywords of this test case for searching purposes.		

4 CASE STUDY APPLICATION

The main objective of the Space Environment Support System (SESS) system is to provide accurate real-time information about the ongoing Space Weather (combination of conditions on the sun, solar wind, magnetosphere, ionosphere and thermosphere) conditions and spacecraft onboard measurements along with predictions for supporting the decision-making process. Within this system the *Data Processing Module* is a critical component. This module is responsible for all file retrieval, parameter extraction and further transformations applied to all identified data, and validation mechanisms, ensuring that all real-time availability constraints are met.

In the scope of the ASSD project both the methodology and defined concepts were applied to this module. For this purpose, an independent team performed the Aspect-Oriented analysis for the proposed case studies, that were later validated by the developer team of the actual SESS project.

4.1 The Data Processing Module

The design of the data processing module was inspired on a typical ETL architecture, but following a completely different paradigm of implementation. Instead of creating specific code for downloading and extracting information for each input file, a declarative language (based on XML technology) and an engine for processing ETL scripts (named *File Format Definitions*) were created. The *Data Processing Module* components and interactions are depicted in Figure 2: (*i*) the *File Retriever* (FR) Engine acquires near real-time data from multiple

[4] I.e. “Unit Tests”, “Integration Tests”, “System Tests”, and “Acceptance Tests”.

external sources in the Internet holding scientific data, based on a set of schedulers, and using a set of well-known protocols like HTTP, FTP or Web Service invocation. For each downloaded file, a local copy is placed in a File Cache for backup purposes and the file is sent for processing by the FET component; (*ii*) *the File Extractor and Transformer* (FET) Engine applies a *File Format Definition* to each input file, extracting all relevant information and delivering it to the *Data Delivery* interface (data entry point for the *Data Integration Module*); (*iii*) all *File Format Definition* files are gener *r* graph t file a ll inforr o c s c

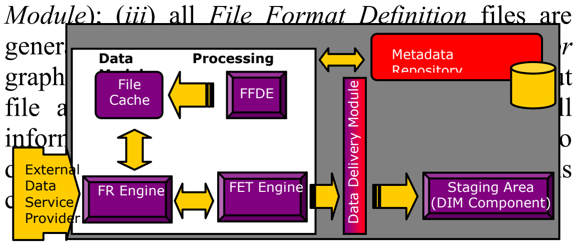

Figure 2: The Data Processing Module architecture.

4.2 Analysis / Discussion of the Results

An analysis of the DPM component focusing on the newly identified concepts has been performed for the main DPM's components: *File Retriever*, *File Format Definition Editor* and *File Extractor and Transformer*. Instances for these proposed concepts have been inserted in the Metadata Repository describing all DPM components. All this information was then manipulated by the Metadata Repository (via the XSLT language) applying the methodology proposed in ASSD project and the resulting outputs have been formatted to HTML. From these outputs, relevant information and characteristics (in both textual and graphical format) can be extracted: (*i*) the Concerns contributions (i.e., positive, negative, inexistent) and relations are represented pictorially in a tabular format, which enables a high-level validation of the analysis correctness; (*ii*) conflicts between concerns are automatically identified and an explanation facility is available for determining the reason of the conflict as the stakeholders that must be contacted to unblock the conflicting situation. Many conflicts are thus resolved at the early steps of the software development instead during the implementation phase; (*iii*) concerns conflicts are resolved through prioritization that will be followed during the implementation phase; (*iv*) crosscutting functionalities are detected for each module, enabling a better strategy for their future implementation.

Comparing the proposed aspect-oriented methodology with the traditional requirement analysis (supported by UML 2.0 diagrams) that was applied initially to the SESS system, several advantages have been identified: (*i*) almost all the conflicts that were found during the implementation of SESS have been identified at the early stages of the software lifecycle using the aspect-oriented paradigm. Since these conflicts were only detected during implementation, some functionalities had to be refactored since the initial implementation collided with other system functionalities that were found to be more important. The adjustment of these functionalities conducted to an unnecessary waste of resources; (*ii*) the initial analysis of SESS although complete, offered some level of ambiguity in some system functionalities, namely in their prioritization. Using the aspect-oriented approach a better prioritization was accomplished between system concerns. This feature enabled the developer to focus on the development tasks, and not to suspend them in order to solve analysis ambiguities; (*iii*) by applying the aspect-oriented analysis, stakeholders had a more active participation in the project, due to their direct involvement in the definition of priorities for the system functionalities. Using the herein proposed concepts and methodology, there is a higher participation and inter-activity in the project between stakeholders, facilitating the communication among them and perception of the project for all; (*iv*) due to the feature of automatic identification of crosscutting behaviour, the developer may "think ahead", prior to the start of the actual implementation, the best way to implement the crosscutting behaviour, instead of refactoring this behaviour at later development stages.

In the following two subsections a set of representative examples of these features are presented for the File Retriever and File Extractor and Transformer components.

4.2.1 File Retriever

Regarding the analysis of the *File Retriever* component, the most significant negative contributions are due to the *Bandwidth Usage*, *CPU Usage*, *Response Time* and *Space Performance* concerns. These negative contributions appears for the *Bandwidth Usage* and *CPU Usage* concerns since the *File Retriever* system intends to minimize the use of network bandwidth (as much as possible, since this component downloads multiple input files, sometimes simultaneously) and the *CPU Usage* since most performed operations are I/O bound and not CPU bound. *Space* Performance also appears with many negative contributions due to the existence of the *File Retriever's* cache, where copies of all downloaded files are stored for backup purposes. Finally, *Response Time* have multiple negative contributions since the duration for the input file download must be minimized as possible in order to accomplish the real-time constraint that data must not take more than five minutes from the moment the download starts (FR responsibility) until data it made available to the user in the client tools.

Four concerns have been identified as the most required within the *File Retriever* system, namely: *Configurability*, *Bandwidth Usage*, *Data Recovery* and *Error Notification*.

The *Configurability* concern is required by many FR's operations since all download activity is configured, or based, on metadata (e.g. the sites for the data service providers, URLs, port numbers, protocols, path and names of the files to download). Bandwidth Usage is also fairly required since the main goal of FR is to download data from external sites. Finally, *Data Recovery* and *Error Notification* are also important since failed download activities must be reported to the system administrator and to the scheduler threads that may try to recover the lost data.

Two main clusters of conflicts have been identified for the system: (i) Bandwidth Usage versus Data Format Interoperability / Data Delivery / Data Recovery / Transfer Data Using a Web Service / Transfer Data Using FTP Protocol and Transfer Data Using HTTP Protocol: all download and data delivery activities affect the network bandwidth resource; (ii) CPU Usage versus Data Processing: although minimum, Data Processing is required for managing the scheduling of operations, thus requiring the use of the CPU resource.

The following concerns have been identified as crosscutting, as they interfere with multiple system functionalities: *Backup, Bandwidth Usage, Configurability, Data Delivery, Data Format Interoperability, Data Recovery, Error Notification, Persistence, Scheduling, Transfer Data Using a Web Service, Transfer Data Using FTP Protocol* and *Transfer Data Using HTTP Protocol*.

4.2.2 File Extractor and Transformer

Regarding the analysis of the *File Extractor and Transformer* component, most significant negative contributions are related with *Bandwidth Usage* and *CPU Usage*. This pattern confirms the performed analysis since the FET system is a major resource user, thus affecting negatively these two concerns.

Five concerns have been identified as the most required within the *File Extractor and Transformer* system, namely: *Configurability*, *Data Format Interoperability*, *Data Recovery*, *Error Notification* and *Validity*.

These concerns are required very often since they are strongly coupled to the *Validity* concern that is an important issue for the FET system. The FET system must, on the one hand, process all input files, determining if the extracted data is valid or if the input file format has changed. When detecting these abnormal cases, the system must not be compromised nor compromise other processing requests (*Data Recovery*) and the system administrator must be advised as soon as possible that some input files are not being processed (*Error Notification*). The FET component requires both internal configuration (ETL script configuration for each particular input file) and parameterization (regarding the *Data Delivery Web Service*), which justifies the strong need for the *Configurability* concern. Finally, the FET component needs to interchange data with data delivery interfaces and with the FR component, so data communication normalization is required (*Data Format Interoperability*).

Regarding conflicts, these have been identified in four match points: *Computation in Parallel*, *Data Delivery*, *Data Processing and Transfer in Parallel*. From these the most representative clusters are: (*i*) *Bandwidth Usage* versus *Data Delivery / Data Format Interoperability / Data Recovery / Transfer in Parallel*: *Data Delivery* usually refers to a considerable amount of data requiring high *Bandwidth Usage*. That is also affected by interoperability constraints, possible recovery actions and simultaneous transfer connection; (*ii*) *Data Recovery* versus *Response Time* / *Parallelism*: *Data Recovery* slows down *Response Time* while Parallelism makes the system prone to error (due to concurrent execution flows).

The following concerns have been identified as crosscutting (as they affect multiple system functionalities): *Bandwidth Usage*, *Configurability*, *Data Delivery*, *Data Format Interoperability*, *Data Recovery*, *Error Notification*, *Transfer Data Using a Web Service* and *Validity*.

5 RELATED WORK

Some approaches have been proposed related with the work presented in this paper. In (I. Brito, 2006) an initial approach for the representation of a *Concern* template is provided. However, this approach is limited to the *Concern* concept (e.g. no requirement information is available) and the navigation between concepts is limited. On the other hand, the work described in (Grundy, 2000) refers a repository for storing aspect components. However, the repository is not structured and merely associates code files to a component name. Searching capabilities are restricted and the solution is specific to the domain and problem.

The authors previous work (R. Ferreira, 2005) in this domain provided a better expressiveness in terms of available concepts and relations between them, comparing with the existing work. However, in a more rigid evaluation of these concepts' modularization some shortcomings have been identified: (*i*) only the concepts closely related with the representation of systems and concerns were identified. Concepts regarding the representation of requirements and test cases were missing; (*ii*) almost all concepts could not be reused in other case studies, since metadata specific to each case study was being stored at the concept level (e.g. Concern) instead at a case study level (i.e. System); (*iii*) only one case study had been applied to the proposed specification, resulting in non- independent and generic concepts.

6 CONCLUSIONS

This paper presented a refinement to early-aspects specification. For this purpose a set of concepts have been introduced and validated using a set of real case studies. Applying the proposed approach, a better traceability regarding the decisions took from requirement elicitation to testing was achieved by the inclusion of *Stakeholder Requirement* and *System Requirement* concepts. Further, *Concerns* can be decomposed via the *Decomposition Node* concept and tested with the *Test Case* concept. In this way more information is available to describe systems. Another major improvement is the concept independence from the system information, increasing instance reusability. In order to solve the problem of the "tangled information", metadata previously present at *Concern* or *Stakeholder* concept that were system dependent had been moved to the *System* concept. This allowed the reuse of stakeholder and concern instances, allowing in a special way the creation of a *Concern* library (enhancing normalization).

All concepts have been validated in a thorough way with different sets of case studies that acted as independent training and testing sets. This methodology was applied to the *Data Processing Module* (partially presented herein) of a real application on the Space domain.

As future work, a structural improvement has to be performed in the concern attribute at the system concept, since the stakeholders' priorities for the concerns need to be prioritized not generally at the system level but particularly at the match point level (I. Brito, 2004). This results, as explained in (I. Brito, 2006), that a same stakeholder may have different interests on the same concern depending on the selected system.

REFERENCES

A. Rashid, A. Moreira, J. Araújo 2003 *Modularisation And Composition Of Aspectual Requirements*. ACM Press, International Conference on Aspect-Oriented Software Development (AOSD 2003), ACM Press, Boston, USA

J. Grundy 1992 *Aspect-Oriented Software Engineering – a Use Case Driven Approach*. Addison-Wesley,

J. Grundy 2000 *Storage and retrieval of Software Components using Aspects*. Australian Computer Science Conference, Canberra, Australia

I. Brito, A. Moreira 2006 *Aspect-Oriented Requirements Analysis (Internal document)*.

I. Brito, A. Moreira 2004 *Integrating the NFR framework in a RE model*. Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design, workshop of the 3rd International Conference on Aspect-Oriented Software Development, Lancaster, UK, 22-26 March 2004

L. Chung, B. A. Nixon, E. Yu, J. Mylopoulos, *Non-Functional Requirements In Software Engineering*. 2000, Kluwer Academic Publishers

A. Lamsweerde 2001 *Goal-Oriented Requirements Engineering: A Guided Tour*. 5th Int'l Symp. On RE, IEEE CS Press,

M. Pantoquilho, N. Viana, R. Ferreira, J. Moura Pires, A. Donati, A. Baumgartner, F. Di Marco, L. Peñin, T. Hormigo 2004 *SEIS:A Decision Support System For Optimizing Spacecraft Operations Strategies*.

R. Ferreira, R. Raminhos, A. Moreira 2005 *Metadata Driven Aspect Specification*. Workshop on Aspect-Oriented Modeling, MoDELS/UML 2005, Jamaica

I. Sommerville, *Software Engineering*. 7th ed. International Computer Science Series. 2004, Addison-Wesley

J. Stapleton, *Dynamic Systems Development Method: A Framework For Business Centered Development*. 1997, Addison-Wesley