Supporting Aspect Oriented Requirements Engineering for Large Documents

Arturo Zambrano[1], Julian Rousselot[2], Johan Fabry[3⋆], and Silvia Gordillo[1⋆⋆]

[1] LIFIA, Facultad de Informática
Universidad Nacional de La Plata, La Plata, Argentina
[arturo,gordillo]@lifia.info.unlp.edu.ar
[2] Facultad de Ingeniería y Ciencias Exactas,
Universidad Argentina de la Empresa,
Ciudad Autónoma de Buenos Aires, Argentina
jrousselot@uade.edu.ar
[3] PLEIAD Lab, Computer Science Department (DCC)
University of Chile, Santiago, Chile
jfabry@dcc.uchile.cl

Abstract. Performing Aspect Oriented Requirements Engineering for large documents is a hard task. This is due to the lack of tools that support demarcation and tracking of crosscutting concerns for requirements documents. In this paper we present the requirements and current implementation status of AORE Assistant, a tool aimed at helping the engineer to manage large documents with multiple crosscutting concerns. This documents are regulations, standard and technical specifications. Such tool should improve software development cycle by easing the location of crosscutting concerns in the documents and helping to avoid problems derived from neglected concerns interactions.

1 Introduction

Requirements engineering is a key task in the process of understanding how the system being built should behave. Complex systems involve myriads of requirements regarding multiple concerns, both functional and non-functional [13]. For some systems, sources of requirements are many, resulting in several – in many cases large – documents. Usually, requirements map straightforwardly to concerns of the application, allowing them to be cleanly encapsulated in different modules of the resulting system. There are however exceptions to this rule, take for example the requirement to log all actions of the system. The corresponding concern, logging, cannot be cleanly encapsulated in one module because it cuts across all other concerns of the system. In the requirements documents it is as if *crosscutting concerns* cut across these documents and across the structure of requirements. For such crosscutting concerns, tracking where these are and which requirements are affected by them is therefore a complex task.

⋆ Partially funded by FONDECYT project 1090083.
⋆⋆ Also CIC Pcia. Bs. As.

Aspect-Oriented Requirements Engineering (AORE) addresses the above requirements engineering problem that some requirements are hard, if not impossible, to isolate into separate modules. Also known as Early Aspects, AORE performs first-class modeling of these crosscutting concerns as aspects, identifying and characterizing their influence on other requirements in the system [9, 11]. These models enable to better identify and manage requirements conflicts, irrespective of the crosscutting nature of the requirement. Ideally, the result of this phase is to have a consistent model of the system early in the software development life-cycle.

The lack of a modularized structure for crosscutting concerns makes it difficult to follow them, specially when documents are many and large. Tools for AORE are needed for this particular regard. Many existing AORE tools are focused on helping in the application of specific AORE approaches [4, 5]. Even though navigation of crosscutting concerns might sound as a basic feature for any AORE tool, as far as we know, there is no support for such functionality.

Having no way of navigating information in a concern based manner poses a difficult scenario for the requirements engineer. He needs to deal with all this information in a non-modular way and cannot easily establish which concerns are present in a single requirement. This is essential for detecting possible interactions of cross-cutting concerns. Interactions between concerns are, in our experience, a source of difficult to identify and costly bugs.

In this paper we present the requirements and first advances in the development of a tool aimed at helping the engineer to manage large documents with multiple crosscutting concerns. The tool is inspired on the needs observed during the development of an industry project in the domain of Slot Machines (SM for short).

This paper is organized as follows: in Sect. 2 we present the domain that inspired us and some peculiarities regarding requirements sources in it. In Sect. 3 we list the requirements for a useful tool in handling crosscutting concerns in large requirements documents. In Sect. 4 we present relevant related work. Sect. 5 presents the current status of our tool for aspect oriented requirement engineering. Finally, we present in Sect. 6 some preliminary conclusions and future work.

2 Motivation

This work is inspired in real-world needs derived from a industry project where we worked for 6 years. For applications in this domain, which we will introduce in the next subsection, there are several requirements sources and lots of functional and non-functional interacting crosscutting concerns.

Our experience of working with these requirements is that the lack of concern track support leaded to costly bugs, detected only once the system was in a production environment. Providing adequate support for tracking such concerns should help to avoid these bugs, resulting in a better design and implementation during the software development life-cycle.

2.1 Slot Machine Domain

A slot machine (SM for short) is a gambling device. It has five reels which spin when a *play* button is pressed. An SM includes some means for entering money, which is mapped to credits. The player bets an amount of credits on each play, the SM randomly selects the displayed symbol for each reel, and pays the corresponding prize, if any. Credits can be extracted (called *cashout*) by different mechanisms such as coins, tickets or electronic transfers.

The SM game concept is developed by the game designers and its implementation must obey a set of regulations that control both hardware and software. The game concept is always a slot machine, what varies is the *skin* around the slots. The regulations that apply to slot machine can be divided in three main groups:

Government Regulations: Government regulations cover a broad spectrum of characteristics of gambling devices: payout, randomness, connectivity, shared prizes, etc. One example of these are the Nevada Regulations [10].

Standards: To ensure proper behavior of SMs, there are certification institutes that perform severals tests and quality checks on the SMs. The expected behavior of an SM is defined in documents called standards, one example is the GLI standard [7].

Technical Specifications: Some requirements are related to the SM connectivity with *reporting systems* (RS) and the underlying communication protocol. This is the case, for example, of the G2S [8] (Game to Server) protocol, an open standard for communication of the SM with a backend.

Requirements for the SM domain are therefore defined in different documents (regulations, standards, protocol specifications) written by different stakeholders, with diverse interests and backgrounds. This results in a large set of documents using multiple terms for describing the same object, action or event. In our experience documents size range from 90 pages (Nevada regulation [10]) to 1500 pages (G2S protocol specification [8]).

The large base of requirements sources, whose organization and structure is out of the control of the engineer makes it hard to reorganize the information so that requirements are grouped by concerns. Moreover, modifying the original documentation is unfeasible, as documents are under the control of their respective owners.

2.2 Concerns in the Slot Machine Domain

Several documents deal with almost the same concerns, but from a different perspective. Consider for example meters, which must be *updated* upon game actions (eg. a play) and are also the base for *accounting* and *reporting*. The first action (*update* after a play) could be stated in a regulation such as Nevada. The latter is part of the protocol specifications (when to *report* such meters). This means that elements from one concern are treated in different documents, according to the stakeholders' interest.

Additionally, regulations from different states and countries treat common concerns such as: metering, communication protocols, etc. We briefly explain some of them here.

- The *Game* concern includes the requirements related to the basic functionality of a SM, being able to perform a play, where some credits are bet by the player and he is rewarded according to an outcome which is randomly determined.
- The *Meters* concern refers to counters that must be maintained. They measure many aspects of the SM activity for auditing purposes – for example: the number of plays, total bet, total won, etc.
- *G2S* concern encompasses those requirements referred to the communication protocol used for monitoring SM behavior remotely. This concern establishes which, when, and how information must be reported. Part of the information is stored in meters.
- *Error Conditions* concern define how the SM must behave under certain circumstances, such us door open, stacker full, bill jam, reverse coin in, etc. This behavior often include to lock up the machine and require attendant intervention.
- *Game Recall* refers to a local audit functionality. Each game play must be stored along with the parameters and outcomes generated for it. This is useful for resolving disputes with the player. At least the last 10 plays must be stored.
- *Program Resumption* is analogous to a *persistence* non-functional requirement. In this domain there is set of data that needs to be recovered after a game reset or a power outage. This information includes: meters, SM status (such as error conditions), queues associated to communication protocols and game recall data.

The reader can think about the *Game* concern as being the base concern (using the jargon of AO community). All the others are crosscutting concerns which cut across *Game* and sometimes the other crosscutting concerns. Consider for example *Program Resumption* which cut across all the others.

3 Requirements for Tool Support for Managing Concerns

The scenario presented in Sect. 2 lead us to generate the following list of requirements for a tool that helps handling crosscutting concerns in several large requirements documents.

3.1 Importing Existing Documents

Requirements for SM come generally from external institutions, the only exception is game design document. The content of these documents is under the

control of the corresponding institution and requirements engineers are not allowed to edit them. Therefore, our tools need no authoring capabilities. Instead, regulations, standards and other documents shall be imported into the tool.

Most of these documents are delivered in PDF format or as MS Word documents. Both formats (as well as many others) can be easily converted to HTML. Therefore, we decided to take as input of our tool HTML documents containing the requirements. We decided for HTML with the aim of keeping the documents appearance as close as possible to the original one.

3.2 Demarcation

Input documents contain the requirements but no explicit information about what are the different concerns result from these requirements. While the requirements engineer works with the document he/she will identify concerns and where they apply.

So our tool must provide the ability to define concerns and to demarcate in the text of the requirements which parts of the document belongs to which concern.

3.3 Per Concern Navigation

Once the documents have been processed and information regarding existing concerns has been added, it is desirable to have navigation capabilities that allow us to explore only those requirements that belongs to a specific concern.

Consider the use case of an engineer that wants to know how to deal with performance constraints given the corresponding regulations and standards. In this case it is necessary to easily locate all those requirements which impose performance constraints or affect performance of the system.

3.4 Scalability

Having a lot of potentially big documents makes scalability of the tool a key point for it successful application. It is necessary to be able to see the big picture of requirements and its concerns. Presenting this information using graphical metaphors helps for quick understanding of how documents are composed and where to search specific information.

We require our tool to be able to provide high abstraction views of the documentation. The user should be able to easily relate this overall view with the specific details of the documents. Therefore it should be possible to zoom in on specific parts, revealing the actual text of the requirements.

3.5 Document Evolution

In the course of our project, every requirements document we used passed through several revisions, where their contents changed. Although at a slow

pace, as standards, regulations are subject to the approval of the respective organizations, they nonetheless evolve to cope with new scenarios, technologies and other needs.

It is then necessary to migrate concern information added from one version of a document to a newer revision, in order to avoid duplicated work between revisions. To achieve this objective it is necessary to keep concern related information from the original document, and for new versions locate those parts of the document that remain unchanged.

4 Related Work

4.1 Modeling Approaches

There are multiple approaches for aspect oriented requirement engineering [5, 6, 9]. We have analyzed some of them in [13]. A general characteristic of these approaches and their supporting tools (where available) is that they demand to manually introduce the requirements. Some of them work on XML, but there is no way of automatically convert requirement documents to the needed XML structures. Some approaches attempt to solve this by doing automatic analysis of textual information. These approaches brings all the complexity derived from analysis of natural language. Given our experience with requirement documents in an industrial case we doubt that this is feasible with current textual analysis technology. Instead, it is our opinion that requirements must be analyzed by engineers with knowledge in the field. The role of the tools is to help these engineers dealing with scalability problems and tracking crosscutting concerns.

4.2 Tools

There are many powerful tools for managing software requirements. One example is Enterprise Architect [2], a UML 2.1 analysis and design tool that covers software development from requirements gathering, through to the analysis stages, design models, testing and maintenance. Other recognized tools are Accompa [1] and Lighthouse [3], generally speaking they perform the same tasks EA does.

All these tools do a great job providing traceability and requirements management, but no one provides support for management, demarcation or visualization of concern related information.

5 Our tool: AORE Assistant

This paper reports on the ongoing work on our tool: AORE Assistant. The goal of AORE Assistant is to provide support for aspect oriented requirements engineering where a large set of requirements are stated, following the needs outlined in Sect. 3.

Development of our tool is based on Java Standard Edition, using a standard JTextPane component for HTML document rendering and Java Graphics2D library for rendering visual representation of concerns in the domain view. Concerns related information is handled separately and stored in XML documents, specifying all the associations between requirements and concerns.

5.1 Importing documents and demarcating concerns

Existing requirements documents must be converted to HTML and AORE Assistant works using those HTML files, maintaining the original formatting. Currently we use one HTML per original document so no hyperlinks are used, but our HTML renderer is capable of handling them anyway.

Demarcation is implemented and it is part of our textual requirements view as shown in Fig. 1, which we discuss next.

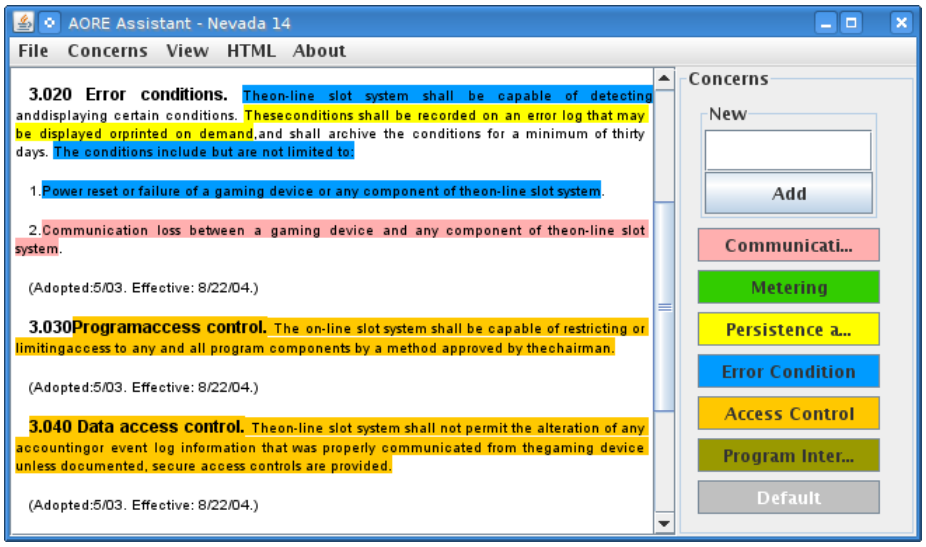

Fig. 1. Requirements View where concerns can be marked in the text.

5.2 Requirements View

In the Requirements view the engineer can read the actual text of document and, as he/she detects parts of the requirements as belonging one concern, he/she can select the text and mark it as part of the mentioned concern. Concern demarcation is performed during the exploration of requirements, and AORE Assistant supports this being done incrementally, by saving information from one session and loading it automatically in the next one. Also, the tool allows

to edit the list of concerns and the associated colors. Information regarding the concerns to which each part of the document belongs to is stored in a separate file, so that when requirements evolve the demarcation work is not lost.

As demarcation can be done even at word level, a paragraph dealing with multiple concerns will be colored according to all participating concerns colors. A multi-colored sentence helps the engineer to locate potential interacting concerns. Aspect or concern interactions [12] are usually neglected, and they are potential sources of complex bugs.

5.3 Domain View

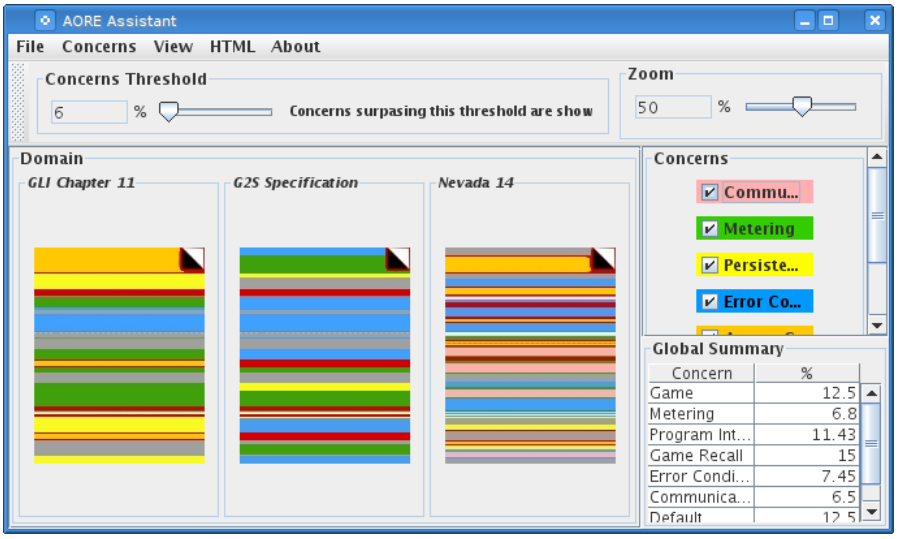

Fig. 2. Domain View showing the requirements documents for our domain.

The Domain View, shown in Fig. 2, presents the documents from a high abstraction perspective. Each document is represented by an icon colored according to the concerns treated in it. The surface covered is in direct relation with the amount of text devoted to this concern for the given document. Also the location of the colored surface is derived from the actual position of that concern in the requirements document. This view also summarizes some information regarding the concerns and how much they occupy in **all** the documents. The slider in the upper part allows to filter and shows only the concerns whose *size* surpasses the specified threshold for each document.

6 Conclusions and Future Work

In this work we have presented a particular domain where many large requirement documents cover several crosscutting functional and non-functional concerns. From our experience in this domain we listed requirements for a tool we believe is useful to the requirements engineer dealing with such scenario. We reported the current implemented features of our tool which include: importing and rendering of requirements documents, demarcation of concerns, high abstraction visualization of documents and their concerns.

As a preliminary validation, we have used this application in some of our real-world documents, even in its current prototype state, it proved to be useful to navigate quickly these large documents based on the concern information added by the engineer.

As future work we plan to implement all the mentioned requirements. After that, our objective is to extend it to assist the engineer in locating and keeping track of concern interactions. Interactions between concerns are, in our experience, a source of difficult to identify and costly bugs, so the final goal of our tool is to avoid such problems.

References

1. Accompa. http://www.accompa.com.
2. Enterprise architect. http://www.sparxsystems.com/.
3. Lighthouse. http://www.artifactsoftware.com/products/Requirements-Management.html.
4. E. Baniassad and S. Clarke. Theme: An approach for aspect-oriented analysis and design. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 158–167, Washington, DC, USA, 2004. IEEE Computer Society.
5. R. Chitchyan, A. Rashid, P. Rayson, and R. Waters. Semantics-based composition for aspect-oriented requirements engineering. In *AOSD '07: Proceedings of the 6th international conference on Aspect-oriented software development*, pages 36–48, New York, NY, USA, 2007. ACM.
6. S. Clarke and E. Baniassad. *Aspect-Oriented Analysis and Design. The Theme Approach*. Object Technology Series. Addison-Wesley, Boston, USA, 2005.
7. Gaming Laboratories International. *Gaming Devices in Casinos*, 2007. Available at: http://www.gaminglabs.com/.
8. Gaming Standard Association. *Game to Server (G2S) Protocol Specification*, 2008. Available at: http://www.gamingstandards.com/.
9. A. Moreira, A. Rashid, and J. Araujo. Multi-dimensional separation of concerns in requirements engineering. In *Proc. 13th IEEE International Conference on Requirements Engineering*, pages 285–296, 29 Aug.–2 Sept. 2005.
10. Nevada Gaming Commission. *Technical Standards For Gaming Devices And On-Line Slot Systems*, 2008. Available at: http://gaming.nv.gov/stats_regs.htm.
11. A. Rashid and A. Moreira. Domain models are NOT aspect free. In *ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems (MODELS06)*, volume 4199 of *Lecture Notes in Computer Science*, pages 155–169. Springer Verlag, October 2006.

12. F. Sanen, E. Truyen, B. D. Win, W. Joosen, N. Loughran, G. Coulson, A. Rashid, A. Nedos, A. Jackson, and S. Clarke. Study on interaction issues. Technical Report AOSD-Europe Deliverable D44, AOSD-Europe-KUL-7, Katholieke Universiteit Leuven, 28 February 2006 2006.
13. A. Zambrano, J. Fabry, G. Jacobson, and S. Gordillo. Expressing aspectual interactions in requirements engineering: experiences in the slot machine domain. In *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC 2010)*, pages 2161–2168. ACM Press, 2010.