Reuse of Spatial Concerns Based on Aspectual Requirements Analysis Patterns

Sara Silva[1], João Araújo[1], Armanda Rodrigues[1], Matias Urbieta[2], Ana Moreira[1], Silvia Gordillo[2], Gustavo Rossi[2]
[1]CITI/FCT, Universidade Nova de Lisboa
Caparica, Portugal
sarapmsilva@gmail.com, ja@di.fct.unl.pt, arodrigues, amm@di.fct.unl.pt
[2]Lifia, Facultad de Informática. UNLP, Argentina
{matias.urbieta, gordillo, gustavo}@lifia.info.unlp.edu.ar

Abstract—**Web Geographic Information Systems (GIS) are systems composed by software, hardware, spatial data and computing operations, which aim to collect, model, store, share, retrieve, manipulate and display geographically referenced data. The development of online geospatial applications is currently on the rise, but this type of application often involves dealing with concerns (i.e., properties) which are inherently volatile, implying a considerable effort for system evolution. Nevertheless, geospatial concerns (e.g., temporarily blocked streets), although changeable, are reusable. However, lack of modularization in software artifacts (including system's models) can compromise reusability. In this context, the use of requirements analysis patterns, enriched with aspect-oriented modeling techniques, can support reusability and improve modularity. In this paper, we introduce requirements analysis patterns for geospatial concerns, to facilitate modularity in GIS Web applications. These patterns are generated from the domain analysis of Web GIS applications and described using a template which is supported by a comprehensive tool, enabling the completion of specific geospatial patterns.**

Keywords: Web GIS; Analysis Patterns; Spatial Concerns; Aspect-Oriented Modeling.

I. Introduction

According to [2], a Geographic Information System (GIS) is a computer system that supports the use and handling of geospatial data. GIS are mainly information systems, which aim to collect, model, store, share, retrieve, manipulate and display geographically referenced data. Web GIS involves the online availability of Geospatial data, with the associated tools.

Geospatial applications involve the temporary availability of spatial concerns (i.e., properties), inherently volatile, although recurring and, therefore, reusable (e.g., temporarily blocked streets). This implies a considerable need for maintenance, not only of the respective information structures, but also of its dynamic behavior. Also, the lack of modularization can compromise flexibility and lead to reuse problems.

Based on knowledge of the application domain, resulting from preliminary domain analysis, the identification of reusable (in our case, spatial) concerns (e.g., map adjustment according to temporal conditions, geographic interfaces) will facilitate Web GIS development. To help with this task at early stages of software development, requirements analysis patterns may be applied to spatial concerns. Analysis patterns [4] are reusable specifications, used at early stages of the development process. The reuse of these patterns and their instantiation in a particular Web GIS application analysis models will speed the development process.

We propose an approach to improve modularization when modeling Web GIS during the requirements specification. The aim of this work is thus to promote requirements modularity and reusability and hence the evolution of Web GIS applications. Nevertheless, the reuse of spatial concerns depends on the availability of appropriate modularization and composition mechanisms. It is important to identify, not only spatial concerns, but also other concerns which are related to these. Moreover, the transverse quality of these concerns must be taken into account, as relevant spatial concerns may crosscut various parts of a particular application.

Aspect-Oriented Software Development (AOSD) is characterized by allowing the identification, modularization and composition of crosscutting properties (or concerns) [3]. One property is said to be crosscutting if it is tangled with another property in a single module or if it is scattered in several system modules. Aspect-orientation is a software reuse paradigm and perfectly suits the specification of patterns' models, as it provides efficient mechanisms to reuse and compose pattern's models to a specific application. In this work we adopt the MATA (Modeling Aspects Using a Transformation Approach) [10], a technique for modeling and composition of patterns based on graph transformations.

In summary, the aim of this work is to create an aspect-oriented requirements analysis approach to model volatile but reusable concerns in Web GIS, specifically geospatial properties, based on analysis patterns. It is also important to define these patterns using an appropriate template for geospatial applications, whose solution models are specified using aspect-oriented principles, enabling the systematic reuse of spatial properties.

The remaining of this paper is organized as follows. Section 2 describes spatial concerns and the MATA approach. Section 3 presents the proposed requirements analysis pattern template. Section 4 applies the pattern to a spatial concern. Section 5 describes the tool support and section 6 discusses the evaluation of the approach. Section 7 depicts some related work. Section 8 draws some conclusions and points directions for future work.

II. Background

A. Spatial Concerns

The use of GIS implies the handling of large volumes of data which are visually integrated in a spatial framework and which, at the same time, need the availability of efficient and adapted data manipulation operations. The highly dimensional operation of GIS data involves the development of complex applications, where relevant concerns may crosscut various parts of a particular application.

The availability of map APIs, enabling the development of geospatial components for existing web services (e.g., Google, Flickr, Facebook, etc.) along with the popularization of the use of Global Positioning Services (GPS), has led to the growing numbers of Location-Based Applications on the Internet. The web context adds an additional difficulty to the development of GIS applications: requirements volatility [9]. These requirements are directly related to the spatial needs behind Web GIS users. They can be identified by the use of a spatial concerns catalogue, in association with the conception of an approach for modeling spatial concerns using aspects in Web GIS applications [7]. This catalogue is currently under development and the results of the work presented in this paper will be added to it.

Mainly, typical spatial concerns can be separated in five categories, described as follows.

Spatial Business Objects: To enhance applications by adding a spatial mapping and representation to business objects (e.g., a bus service management system can be improved by providing real-time bus locations).

Rich Spatial Data: Enriching a geographic object with additional (not geo-referenced) information (e.g. adding a video to a specific location on a map).

Spatially-Constrained Behavior: Change or modify the behavior of an object according to its actual location (e.g. pricing and taxing processes may change with objects' locations).

Map adjustments: To extend or restrict the available spatial information according to the application's constraints (e.g. certain parts of a map may be, according to temporal or permanent restrictions, unavailable or useless for specific operations).

Geographic Interfaces: To modify (or upgrade) the user interface of geographic objects. Though this is not strictly a spatial concern, it is clear that the previously described concerns may introduce changes in the application's user interface, specifically in the representation of geographic objects (e.g., to introduce a particular symbolization to make the user aware that a road cannot be used during a particular period of the day).

By addressing these types of (spatial) concerns we aim to identify the situations in which they may be present in web applications and to develop and approach to handle them early on in the development process, specifically during requirements specification.

B. MATA

The MATA [10] aspect-oriented modeling approach is based on UML, allowing aspects composition using, for example, class diagrams, sequence diagrams and state diagrams. Here we focus on MATA to model aspectual classes by using and adapting class diagrams. To specify aspectual classes, some stereotypes are used to define composition rules: <<create>>, which states that the element will be created in the base model; and <<delete>>, which states that the element will be deleted of the base model; <<context>>, which states that the element will not be affected by the other two stereotypes. Only <<create>> will be used in this paper.

Variables in MATA are prefixed by a vertical bar "|", meaning that "|X" will match any model element with the same type of X. After specifying both kinds of models, base and aspectual, a pattern matching is made between them. This means that the MATA tool tries to establish a connection between elements of each model, always respecting the composition rules defined in the aspectual model. The resulting composed model includes the elements of both models, according to the rules defined. MATA allows more composition combinations than other existing aspect-oriented modeling tools.

III. Aspectual Requirements Analysis Pattern Template

The template used to define requirements analysis patterns – Table I – is based on the one proposed in [8], but adopts MATA aspectual notation for the modeling part. The motivation to define and use a new pattern template instead of adopting an existing one (e.g., [4]) is that neither do the current templates support composition mechanisms nor do they provide more detailed models such as variability models [6]. Our analysis pattern template is depicted in Table I.

The description of a pattern requires a meaningful name. Then, it is necessary to describe what problem the proposed pattern seeks to solve and explain the context in which it commonly appears namely what kind of situations it is relevant to. After introducing the purpose of the pattern, its functional and non-functional (NF) requirements are described, followed by the specification of the dependencies between them and the actors that take action on the problem.

After specifying the requirements, the next step is to build the structural and behavioral models. The structural modeling consists of describing: which features should be present in the solution, represented in a feature diagram highlighting the variability of the pattern; and which classes will be needed to implement the solution, represented by a MATA (or aspectual) class diagram, which will identify concrete and variable classes and their relationships. Variable classes are the ones that need to be instantiated when composing with the base classes of an application. Therefore, the aspectual class diagram will describe the pattern structure, and the examples will show the composition of the base structure with the aspectual one.

The following step is behavioral modeling, describing the behavior that the solution must have, or what activities will be required to solve the problem. The behavioral modeling will be done through a sequence diagram, also supported by AOSD methodologies, to facilitate the modularization. More specifically, we used scenarios and aspectual composition, realized through the mechanisms offered by MATA [10]. Thus, the sequence diagram will describe the aspect that represents the pattern behavior, and the examples will show the composition of the base scenario with the aspectual scenario, as illustrated in the previous section. Once the modeling is completed, we describe the consequences of the pattern's application, i.e., its strengths and limitations.

Some events that would likely trigger the pattern are also specified. Some examples of applications of the pattern should also be provided. To describe these examples, features, class and sequence diagrams are configured for a particular application. The examples must include diagrams for the base and composed scenarios.

Also, an analysis is made in order to identify possible relationships between this pattern and other patterns, which results in a list of related patterns. At the end of this process, a detailed description of the pattern is provided, which can be reused in other applications.

TABLE I. ANALYSIS PATTERN TEMPLATE

Field			Description
Name			Pattern name identifier.
Problem			Describes the problem that the pattern intends to address.
Context			Describes the environment in which the problem and solution apply.
Requirements	Functional		List of functional requirements.
	Non-Functional		List of NF requirements.
	Dependencies		List of dependencies between functional and NF requirements.
	Actors		List of actors.
Events List			Identifies some examples of events that trigger the pattern.
Modeling	Struc ture	Feature Model	Shows optional and mandatory features of the pattern.
		Class Diagram	Shows the pattern's classes and relationships
	Beha vior	Sequence Diagram	Shows the dynamic behavior of the pattern.
Consequences			Advantages and disadvantages of the pattern's application.
Examples	Features Diagram		Examples using these models that illustrate the context of the pattern, as it is applied and any necessary amendments to the initial context.
	Class Diagram		
	Sequence Diagram		
List of related patterns.			List of already defined patterns related to the proposed one.

IV. PATTERN DEFINITION

One of the aims of this work is to identify spatial properties in Web GIS applications, described using analysis patterns. The Geo-reference Entity pattern was one pattern identified in this work, and it is presented below. The NF requirements (and the respective dependencies) plus the behavioral models are not shown here due to lack of space.

Name: Geo-referenced Entity

Problem: Many web applications were not designed with the intention of providing geospatial characteristics. However, to support spatial behavior, some of the entities composing these applications will need geo-referencing.

Context: This problem applies to systems that involved, initially, no geospatial features, but which can benefit from them. For example, an application for a bus company that offers bus routes may be improved by providing users with buses' locations in real time. The application will also benefit from the availability of more accurate and timely information, which may involve bus routes' geographic mapping. The assumptions taken are that the applications' entities are either static (do not move in space, for example, a garage) or mobile (may move during the execution of the application, for example, a bus or a person). A static entity is associated with a location, while a mobile entity may be associated with several locations, during the execution of the application. It is also assumed that a mobile entity holds a main location (which can be home, an office, a garage) but, as it moves, it may become associated with a secondary location. This secondary location may be obtained from the analysis of the entity's schedule and requested from static entities, referenced in the schedule. This means that a bus may be located at a bus stop, which is a static location, or on route to the next stop and, at this time, its location is the street it is traversing, which is also a static location.

Requirements

<u>*Functional*</u>

1. Geo-reference an application's entity.
2. Check if the entity is static or mobile:
   2.1. If static, obtain the geographic reference data.
   2.2. If mobile obtain system time and entity's schedule:
      2.2.1. Obtain the details of the entity's location.

<u>*Actors*</u>: User; Entities; System/Application.

Modeling<u>*: Structural*</u>

<u>Feature Model:</u> Fig. 1 illustrates a feature model [6] for locating an entity without georeferencing. This model shows the variable and mandatory features associate to the pattern.

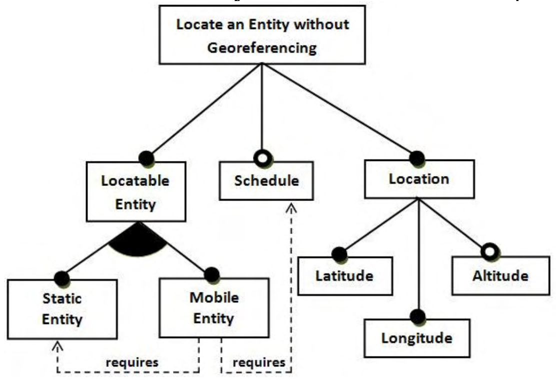


Figure 1. Feature Model for Locate Entity

There must be a *Locatable Entity* which can be either a *Static Entity* or a *Mobile Entity*. There must also, necessarily, be a *Location*, and this has a *Longitude* and *Latitude* and may or may not have an *Altitude*. Optionally, we can still have a *Schedule*. When we have a *Mobile Entity,* there must be at least one *Static Entity* and a *Schedule*. Besides specifying the variability of the pattern, this model can be used to help defining the class diagram as some features can be mapped to domain classes.

Class Diagram: In the diagram in Fig. 2 we have six classes: *|Interface*, *|Control*, *|Locatable Entity, Schedule*, *|Main Location* and *|Secondary Location*. The class *|Interface* is the intermediary between the user and *|Control*, and the latter (the *|Control* class) is responsible for checking and request the necessary data. The *|Control* class communicates with the *|Locatable Entity* class. The *|Locatable Entity* has a single *Schedule*, one *|Main Location* and can have several *|Secondary Location*. One *Schedule* only belongs to one *Locatable Entity*. Either one *|Main Location* as well as a *|Secondary Location* can be of several *|Locatable Entity*. Note that we defined those classes as variables (except Schedule).

Note that the classes *|Interface*, *|Control*, and *|Locatable Entity* must be matched against classes in the base class diagram (in a particular application), for composition purposes. The classes *Schedule*, *|Main Location* and *|Secondary Location* comprehend the new classes that will be added to the base class diagram (that is why they and their related associations are defined with the stereotype <<create>>). These classes must be instantiated with concrete classes. The instantiation will be shown in the *Examples* section.

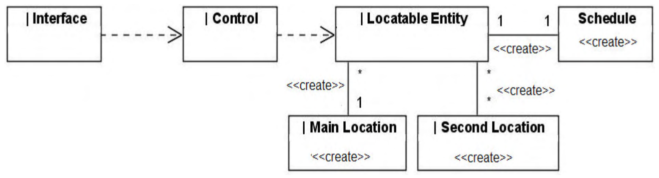


Figure 2. Aspectual Class Diagram for locating an entity

Consequences: The application of this pattern will enable the spatial location of an application's entity, allowing that this may be presented visually, for example, in a map.

Events List: Possible events are:

1. Fire breaks out in a building and you must find all the occupants therein to ensure that everyone leaves the building;
2. A bus company needs to inform its users, in real time, of the location of a given bus.

Examples: Here we show the actual application of the pattern models described above. This pattern can be applied to static entities, i.e. entities that have always the same location and in this case, this is simply stored in a database. Moreover, the pattern can also be applied to mobile entities, i.e. entities that change location according to certain characteristics, such as an individual or a vehicle.

As an example, let us consider the application of this pattern to the CLIP system (http://clip.unl.pt), a real information system of courses and schedules of all students, lecturers and other employees at Universidade Nova de Lisboa.

In this system, individuals (e.g. lecturers and students) and classes are considered mobile entities, while rooms and offices are static. Let us look at an example of locating a mobile entity.

Consider that an emergency happens. Consequently, we need to convene a meeting of lecturers, but their locations must be also resolved. Thus we want to ***Locate Spatially*** the ***Lecturer entity*** – that is the application of the pattern. Below we instantiate the feature model and the class diagram for this particular situation.

Feature Model: Fig. 3 has the necessary characteristics to locate a Lecturer. As you can see, you need an object of class *Lecturer*, its *Schedule* and its *Location*, and the latter must be composed of a *Latitude* and *Longitude*, and may also include an *Altitude*. The Lecturer may also have an *Office* and/or a *Room*. This means that, during working hours, when a *Lecturer* is not teaching, his location should be his *Office*. If s/he is teaching, the location should be taken from the *Schedule* and in this case, it should be a *Room.*

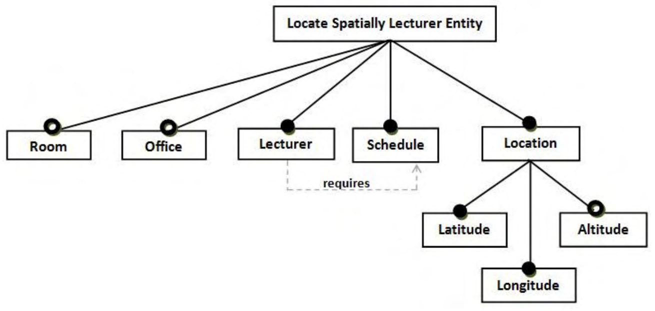


Figure 3. Feature Diagram for locating a lecturer

Class Diagram: Fig. 4 illustrates the (base) class diagram containing only the classes related to the objects relevant to convene a meeting of lecturers. Thus, it will need *CLIP Interface*, *CLIP Control*, *Lecturer* and *Meeting* classes.

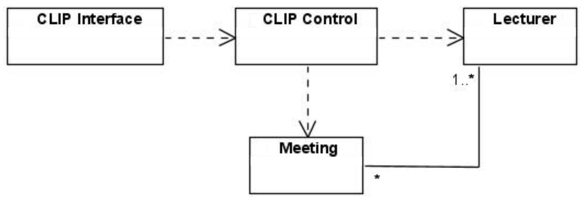


Figure 4. Base Class Diagram for convening a lecturers' meeting

Note that the aspectual class diagram in Fig. 2 complements this one by adding the classes necessary to locate the lecturers.

Fig. 5 shows the class diagram that is the composition of the base class diagram with the aspectual class diagram shown in Fig. 2. So we have the aspect's classes instantiated and composed with the base classes. We thus instantiated aspect classes *|Interface* with *CLIP Interface*, *|Control* with *CLIP Control*, *|Locatable Entity* with *Lecturer*, *|Main Location* with *Office* and *|Secondary Location* with *Class Room*.

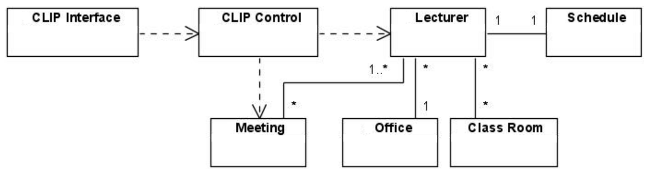


Figure 5. Composed Class Diagram

Related Patterns: Add temporal availability for spatial entities, Presentation of spatial entities; Adjust the state of entities (Add/Remove all or part of spatial entities).

V. PATTERN TOOL

The *PatternTool* tool was created to support this template, allowing the creation of patterns using the Eclipse environment. Fig. 6 presents the generic view of the tool. The pallet of the tool is presented on the left hand side, where all the fields needed to create a pattern can be added. The fields are placed on the pallet in the order wherein they must be placed in the template. The colors help to understand when a field should be placed within another.

The Pattern Template field should be the first one to be clicked, as it starts the creation of a new pattern. The hierarchy of the fields of the template is represented by a color scheme. For example, the fields to be filled within this Pattern Template are icons with dark green color, such as the Pattern's Name field, Pattern's Requirements Analysis, among others. When a field is placed within a field with a Dark Green icon, its icon is Light Green, and finally, when a field must be placed inside a field with a Light Green icon, its icon color will be an even lighter green, as is the case of the Features Diagram icon. To edit feature, class and sequence diagrams, after inserting its compartment in the template, an additional editor can be opened, which can be used to build the diagram.

VI. EVALUATION

Here we discuss the evaluation for both the Pattern Tool and the pattern description. We selected 15 subjects, all students from our Faculty's MSc on Informatics Engineering. Only 2 had already finished the course and were working in industry. All of them had courses on analysis models. About 85% had a course on GIS.

The tool evaluation consisted of a set of questions about language expressivity and syntax, tool usability and the satisfaction level of the users. The pattern evaluation involved questions about simplicity and clarity of the description and the relevance of the pattern and of each of its attributes.

A. Pattern Tool Evaluation

The evaluation results were positive: the users considered the tool very useful and intuitive, easy to use and understand. The results of the tool evaluation are discussed below.

1. How easily were the concepts identified in the tool? The aim was to assess the quality of the concepts representation. Concerning the representation, 10 out of 15 users thought it was "very easy" to identify the concepts, while the remaining 5 considered it "easy".

2. What is your overall impression of the tool? The aim was to evaluate whether users liked to use the tool. In general, results were positive as 12 out of 15 users thought the tool was "good", two rated it "very good" and one of them considered it "average".

3. Do you consider that it was easy to migrate the pattern from paper to the tool? The aim here was to evaluate whether users felt lost while moving the pattern from paper to the Pattern Tool. The results showed that 11 out of 15 users considered the migration "easy" and 4 others considered it "very easy".

4. Do you consider the tool useful? The aim was to evaluate whether users felt that the tool was useful. The results were positive. They said that no other tool enabled the creation of the pattern with such detail.

B. Pattern Description Evaluation

The results obtained were positive: the users felt that the pattern described is quite relevant and useful; the description was clear and simple, applicable to various areas and easy to reuse. This evaluation is discussed below.

1. How do you rate the relevance of the pattern? The aim was to evaluate whether users believed that the pattern was important and could bring benefits for other applications. Results showed that the pattern was considered important and useful, as only 3 out of 15 users found relevance "moderate", 11 considered it "high" and 1, "very high".

2. Do you think this pattern can be reused in various applications in different areas? The aim was to evaluate whether users found the pattern well described and generic enough to be applied to various areas and applications. Results showed that all users found this to be true.

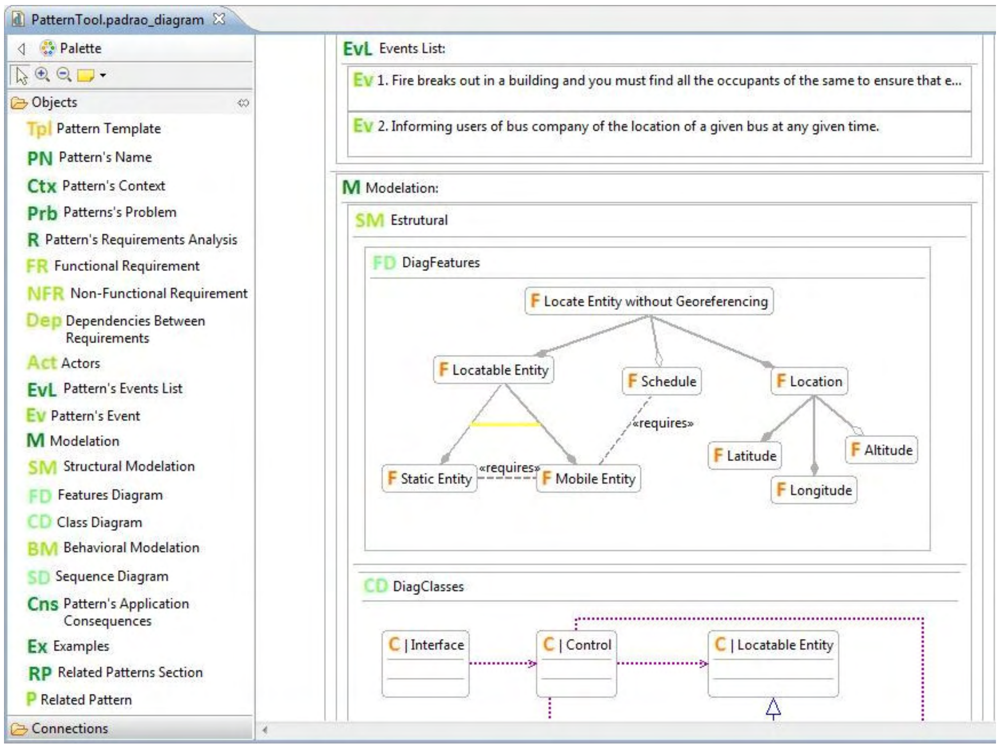
3. Do you think that GIS applications can benefit from this pattern? All users believed that the pattern can bring benefits when used in GIS applications.

4. How do you rate the clarity and simplicity of the definition of the pattern? The aim was to verify that the definition of the pattern was written in a simple and clear way. The answers of users were very positive, since 13 out of 15 users classified clarity and simplicity in the pattern as "good" and the remaining two thought it was "very good".

C. Evaluation Threats

Having an evaluation in the industrial/business environment would give a different perspective to the evaluation of the tool and the pattern description. However, the users who evaluated both the tool and the description of the pattern knew the most recent technologies, which does not always happen in a business environment.

The evaluation was performed with only 15 users, but although the statistical significance is reduced, the results are indicative of the acceptance of the approach evaluated.


Ev 1. Fire breaks out in a building and you must find all the occupants of the same to ensure that e...

Ev 2. Informing users of bus company of the location of a given bus at any given time.

Figure 6. Generic view of the PatternTool

VII. Related Work

Oliveira et al. [7] presented an AOSD approach to identify, modularize and compose crosscutting concerns, more precisely spatial concerns, in Web GIS applications. The identification of requirements is achieved through a thorough knowledge of the Web GIS domain, obtained by domain analysis techniques. Nevertheless, it did not consider reusing requirements analysis patterns.

Gordillo et al. [5] developed a technique for modeling object-oriented GIS where, from the basic geographic model, spatial characteristics are added to each object in a dynamic and transparent way. This technique relied on object-oriented methodologies to obtain reusable, modular and modifiable software, as well as objects that encapsulate knowledge. This work focused on design and did not address modularization of crosscutting concerns.

In [1] a catalogue of common functionalities for defining a basic Web GIS application is proposed. However, the description of the functionalities is not as detailed as the approach presented in this paper.

VIII. Conclusion

In this paper, Web GIS applications were examined to identify some of their volatile but reusable spatial concerns, since these applications are characterized by the constant change of requirements. Thus, an aspectual analysis pattern template was defined and applied to the modeling of spatial concerns patterns. One of the advantages of the template is the use of the MATA notation, which provides efficient mechanisms for modeling and composing static and behavioral elements of a pattern. Some work is still under development, which includes the analysis of additional patterns and tool improvement.

Acknowledgment

We want to thank, for the financial support for this work, CITI – PEst-OE/EEI/UI0527/2011, CITI/FCT/UNL.

References

[1] Digital Earth Summit on Geoinformatics: Tools for Global Change Research. Int. Journal of Digital Earth, 1:1, 2008, pp. 171-173.

[2] S. Dragicevic, "The Potential of Web-based GIS", Journal of Geographic Systems, Springer, vol. 6, 2004, pp. 79-81.

[3] E. Filman, T. Elrad, S. Clarke and M. Aksit, "Aspect-Oriented Software Development", Addison-Wesley, 2005.

[4] M. Fowler, "Analysis Patterns - Reusable Object Models", Addison Wesley, 1997.

[5] S. Gordillo, F. Balaguer, C. Mostaccio, F. Neves, "Developing GIS Applications with Objects: A Design Patterns Approach", GeoInformática, vol. 3:1, 1999, pp. 7-32.

[6] K. Kang, S. Cohen, J. Hess, W. Nowak, S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report: CMU/SEI-90-TR-021, Pittsburgh, USA, 1990.

[7] A. Oliveira, M. Urbieta, J. Araújo, A. Rodrigues, A. Moreira, S. Gordillo, G. Rossi, "Improving the Quality of Web-GIS Modularity Using Aspects", QUATIC, Portugal, 2010, pp. 132-141

[8] M. Pantoquilho, R. Raminhos and J. Araújo, "Analysis Patterns Specifications: Filling the Gaps", Viking PLoP, Norway, 2003.

[9] I. Sommerville, Software Enfgineering, Addison Wesley, 2010.

[10] J. Whittle, P. Jayaraman, A. Elkhodary, A. Moreira, J. Araújo, MATA: A Unified Approach for Composing UML Aspect Models Based on Graph Transformation, Transactions on AOSD, vol. 6, 2009, pp. 191-237.