Early Aspect Identification from Use Cases using NLP and WSD Techniques

Alejandro Rago
Fac. Cs. Exactas, UNICEN University - Argentine
Campus Universitario
Paraje Arroyo Seco
+54 2293 439681/2
arago@alumnos.exa.unicen.edu.ar

Esteban S. Abait, Claudia Marcos
ISISTAN Research Institute,
Fac. Cs. Exactas,
UNICEN University, Argentine
Campus Universitario
Paraje Arroyo Seco
+54 2293 439681/2 - ext. 26
{eabait, cmarcos}@exa.unicen.edu.ar

Andrés Diaz-Pace
Software Engineering Institute,
Carnegie Mellon University
4500 Fifth Ave., Pittsburgh
PA, 15232 – USA
+1 412 268 9565
adiaz@sei.cmu.edu

ABSTRACT
In this article, we present a semi-automated approach for identifying candidate early aspects in requirements specifications. This approach aims at improving the precision of the aspect identification process in use cases, and also solving some problems of existing aspect mining techniques caused by the vagueness and ambiguity of text in natural language. To do so, we apply a combination of text analysis techniques such as: natural language processing (NLP) and word sense disambiguation (WSD). As a result, our approach is able to generate a graph of candidate concerns that crosscut the use cases, as well as a ranking of these concerns according to their importance. The developer then selects which concerns are relevant for his/her domain. Although there are still some challenges, we argue that this approach can be easily integrated into a UML development methodology, leading to improved requirements elicitation.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/Specifications – *Methodologies (e.g., object-oriented, structured)*

General Terms
Design, Documentation, Languages.

Keywords
Early aspect identification, semantic aspect analysis

1. INTRODUCTION
Accomplishing a good separation of concerns [1] since early development stages, such as requirements elicitation and architectural design, is crucial [2] because many concerns have far-reaching effects on design and implementation decisions. A poor concern identification usually precludes developers from reasoning about their effects on the system (or on other concerns), and consequently affects the final quality of the software product. For example, a security concern that is inadvertently skipped in a use case can lead to wrong implementation, in which adding security mechanisms late in the development can be really hard. Over the last years, it has been argued [2] that the identification of the so-called *early aspects* can significantly help developers to analyze and plan for design tradeoffs early in the lifecycle. In this article, we describe a semi-automated approach to mine candidate early aspects from requirements specifications more effectively.

Other researchers have previously approached the management of early aspects in systematic ways. For example: Theme/Doc [6,7] exposes the relationships between the components of a system; EA-Miner [14,15] permits a quick identification of system-related aspects from unstructured requirements; Rosenhainer et al. [16] use Information Retrieval techniques to detect possible concerns in requirements; among others. Related to concern identification, Shepherd et al. [17,18] propose a structure called Action-Oriented Identifier Graph (AOIG) that gives supports for refactoring object-oriented code into aspects. All these approaches have shown interesting results with respect to parts of the process of identifying early aspects. However, these approaches still present drawbacks and challenges. A first drawback is the low precision when mining candidate aspects, due to the intrinsic difficulty of analyzing text in natural language. A second drawback is a weak syntactic analysis of the text, and the lack of semantic analysis. Some challenges include: integration with software development methodologies and scalability of the analysis techniques.

In this context, we argue that the combination of techniques such as Natural Language Processing (NPL) [12] and Word Sense Disambiguation (WSD) [13] has a great potential for boosting the precision of algorithms for discovering early aspects. We describe here an aspect mining approach that takes a use case specification as input, and then is able to perform syntactic and semantic analyses of this input in order to identify potential concerns of the system. The concerns are captured by an extended AOIG, and they can refer either to functional or quality-attribute (extra-functional) system properties. We have also developed tool support for the approach, so as to minimize the developer's work when looking for early aspects in the AOIG. As the tool proposes

EA'09, March 3, 2009, Charlottesville, VA, USA.

candidate aspects, the developer can filter out those aspects that are relevant to her/his system. The output of the approach is an extended use case specification that includes those aspects selected by the developer and their relationships to the use cases. The main contribution of this work is the incorporation of word sense awareness in an aspect identification process, as well as a workflow of activities developed to alleviate the problems caused by ambiguities and synonyms in requirements in natural text.

The rest of this paper is organized into 4 sections. Section 2 describes the main characteristics of an aspect mining approach, which drove the development of our approach. Section 3 presents the activities of the aspect identification process, based on text analysis techniques. Section 4 shows preliminary results obtained with the tool that supports the approach. Finally, Section 5 concludes the paper and discusses future work.

2. CHARACTERISTICS FOR AN ASPECT MINING APPROACH

From a study of approaches from the literature, and based on our own experience, we believe that a tractable approach for identifying early aspects from software artifacts should be:

1. *Aware of documentation structure.* The use of templates [10] for software documentation (e.g., UML use case template) is a standard practice in many development methodologies. Thus, the way in which textual information is captured and organized provides clues about candidate aspects.
2. *Transparent to the developer.* Several aspect mining approaches, even when providing tool support, assume that the developer is willing to provide as much information as need for the analysis of early aspects. This limits the practicality of the approaches. Instead, the analysis should take whatever software documents are available, and require as little information from the developer as needed.
3. *Driven by semantic analysis.* Although the ambiguity problems coming from the use of natural language are unavoidable, these problems can be alleviated in particular domains by combining lexical, syntactic and semantic analyses of structured information. Of course, the analysis techniques to be applied here will be more complex.

Let's consider a motivating example. We have a system that can generate workflows of tasks that can be executed by users. An analyst has elicited three use cases for this system from the stakeholders, as depicted in Figure 1. In addition, the analyst has documented some supplementary requirements that apply to the use cases. Before proceeding with the system design, the developers are interested in early aspects they should be aware of. However, eliciting such information directly from the stakeholders is a tedious and error-prone activity. A more efficient solution is to have a tool assisting the analyst to process the available use cases and extract a list of candidate early aspects. This way, (s)he can quickly confront those early aspects with the stakeholders and pass the relevant aspects to the developers.

Interestingly, several text analysis techniques to deal with the situation illustrated by the example have been developed for other domains (e.g., web search). Despite the availability of these techniques, their application to analyzing latent information in software artifacts is still a topic of research. In particular, we have investigated a number of text analysis techniques that can be added to an aspect mining tool, so as to take into account the three characteristics stated above. The proposed techniques include:

- Use an NLP tool to perform a lexical and syntactic analysis of the text of the use cases. This tool will tag certain words of each sentence, indicating if they are verbs, nouns, etc.
- Perform a WSD analysis of each tagged word, which will determine the meaning of the word according to its context.
- Perform an analysis of word groups based on semantic dictionaries [4], in order to generate clusters of words having similar semantic meaning. This processing also includes the use of thesaurus and stemming algorithms.

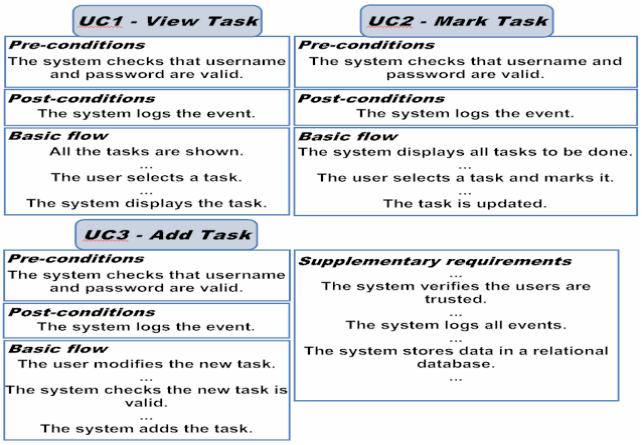

Figure 1 – Sample use cases

3. PROPOSED APPROACH

A common technique for identifying aspects in textual specifications is to search for verbs (actions) that indicate crosscutting behavior [7, 15, 16]. For instance, action verbs such as 'verify', 'check', 'log', or 'update' often give clues about crosscutting behaviors. A better technique is that of looking (if possible) at the objects to which these verbs affect [17]. These objects are called direct objects (DOs). The key idea here is that an action applied to different objects can lead to different behaviors (even when using the same verb). We can then use combinations of verbs and objects to find specific crosscutting concerns, which would be otherwise overlooked when analyzing single verbs. For instance, the sentences "The system checks that username and password are valid" (precondition of UC1 in Figure 1) and "The system checks the new task is valid" (basic flow of UC3 in Figure 1) use the same verb but refer to different behaviors. The first verb-object pair is about validation of user access to the system, while the second is about consistency of task data entered into the system.

Along this line, we propose a semantic analysis of verb-object pairs in specifications expressed in natural language, based on the AOIG proposed in [18]. There are cases in which the text in natural language does not include information about direct objects, so individual verbs must be also considered in the AOIG. In addition, verbs can be grouped in clusters according to their semantic meaning. We have extended the AOIG technique to

support these two considerations. Figure 2 shows a possible AOIG built from our use cases. In our AOIG, there are four types of nodes: (i) nodes that correspond to verbs, (ii) nodes that correspond to objects, (iii) nodes that correspond to verb clusters, and (iv) nodes that correspond to object clusters. The arcs in the graph are used to navigate the nodes, that is, from a verb to a direct object (and vice versa), from a verb to a verb cluster, or from an object to an object cluster. The dashed lines stand for verbs without a corresponding direct object. As we will explain later, the nodes of the AOIG are linked to text of the use cases.

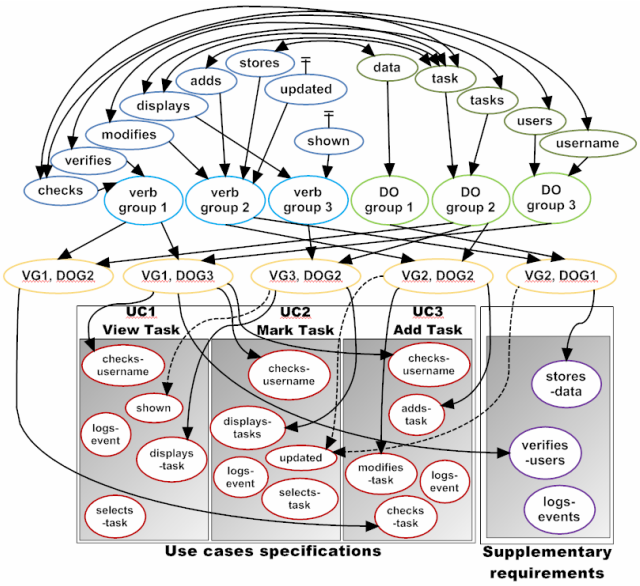

Figure 2 – Capturing concerns with an extended AIOG

UML is used as the modeling support for our approach. In particular, the use-case notation [8] of UML provides a structure for textual specifications, and we can benefit from that structure when looking for potential crosscutting concerns. However, this approach can be applied to any requirements specification document. A standard use-case template [10] is structured around five sections, namely: basic flow, alternative flow, precondition, postcondition, and special requirements. The last section collects all the requirements (such as quality-attribute issues) on the use case that are not considered in the above sections, but that should be taken care of during design or implementation. Use cases are often accompanied by a supplementary specification for system requirements that are not noted in the use-case template [10].

Assuming a use case in which all its sections are filled in with text, we extract verb-object pairs from the text and categorize the pairs either as functional or quality-attribute concerns. A (potential) functional concern is detected by looking at the sections 'basic flow', 'alternative flow', 'precondition' and 'postcondition'. A (potential) quality-attribute concern is the result of a verb-object pair detected in the 'special requirements' section of the template. Also, a quality-attribute concern may come from verb-object pairs in the supplementary requirements. For instance, in Figure 2, there is an 'accessing' concern tagged as a quality-attribute aspect, because one of its verbs ('verifies') is present in the supplementary requirements (verifies-users).

The processing of use cases to build an extended AOIG comprises two phases: (i) tagging of use-case text using NLP, and (ii) semantic analysis of tagged words using WSD. In the first phase, NLP begins tagging the basic and alternative flows, preconditions and postconditions of each use case. After that, the 'special requirements' section as well as any supplementary information are tagged. In the second phase, semantic dictionaries [4] are used to perform the semantic disambiguation of words. These dictionaries give support to the creation of clusters in our AOIG. Each cluster node is a consequence of navigating the semantic relationships between terms of a dictionary. By clustering related verbs and objects after the semantic disambiguation of words, we can tackle some problems caused by synonyms, ambiguity or vagueness in natural language. For example, in Figure 2, two verb nodes referring to verifications ('checks' and 'verifies') are clustered in node 'verb group 1';four verbs associated with persistency ('adds', 'stores', 'updates', and 'modifies') go to 'verb group 2', and so on. The clustering is similar for direct objects.

Once the AOIG is built, the graph needs to be traversed in order to select nodes that point to two or more use cases. We refer to this type of nodes as "hint node". The hint nodes are the ones that actually lead to candidate early aspects. In Figure 2, note that hint nodes act like "behavior" separators: the node (VG1,DOG2) refers to a particular behavior, while the node (VG1,DOG3) refers to a different one. Remember also that nodes are already classified either as functional or quality-attribute concerns. The traversal of the AOIG returns a list with a ranking of hint nodes that showed crosscutting behavior. Then, the list is presented to the analyst, who can select and refine the concerns into aspects that (s)he judges relevant for the domain. Filters can be applied here to further prune the list, based on analyst's criteria. For instance, (s)he can apply a filter based on a specific direct object, e.g. "user", and get only those aspects related to user issues. A similar filtering can be done for specific verbs, or other words.

4. PROCESS AND TECHNIQUES USED

As outlined above, our approach processes a set of use cases, then constructs an AOIG, and finally generates candidate early aspects out of the graph. In more detail, the aspect identification process involves three main blocks, as depicted in Figure 3.

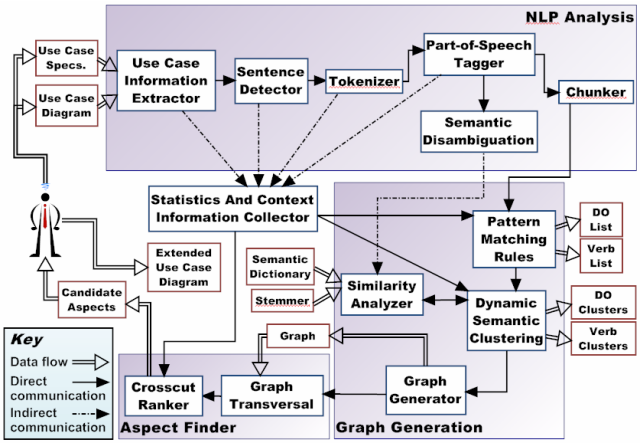

Figure 3 – Steps of the Aspect Identification Process

The *NLP Analysis* block performs lexical, syntactic and semantic analyses of the textual sentences used in each use case. In addition, this block collects statistics and data about the placement of words in the sentences. The tasks involved in NLP

Analysis include: information extraction, detection of sentences, detection of words inside the sentences, word tagging using part-of-speech (POS) techniques, semantic disambiguation of words, and word grouping according to syntactic criteria. An NLP tool implements all these tasks, except for the semantic disambiguation task that is implemented by an algorithm known as "Maximum Relatedness Disambiguation" (MRD) [13]. In this work, we used some Java-based NLP toolkits provided by OpenNLP [3], which implement infrastructure for common NLP components.

The purpose of disambiguation is to determine the intended sense of a particular word (with many potential meanings) when used in particular context. The MRD algorithm basically chooses the word sense that maximizes the semantic relationships of that word with other words within a limited context window. To do so, there are different metrics to quantify semantic relationships among words. In our work, we modified the original algorithm so as to consider all the words in the text, and we also set the context window of a word to the whole sentence that contains that word. The word definitions are taken from the WordNet semantic dictionary [4]. This is a database that contains information about nouns, verbs, adjectives and adverbs. WordNet is structured in terms of sets of synonyms (called synsets). Each synset can be seen as the representation of a concept or sense. WordNet also connects the concepts by means of different relationships (hypernyms, hiponyms, troponyms, etc.). Therefore, the user of WordNet (like our approach) has a network of concepts available, in which related concepts are identified by computing a "distance" metric between them. Two common metrics in the algorithm are Lesk Gloss Overalp Measure (LGOM) and Lesk Extended Glos Overlap Measure (LEGOM) [11]. LGOM is easy to compute but not very effective, while LEGOM is more effective than LGOM but its computation takes more time. For instance, let's assume that we want to find the correct sense for the word 'project' in the sentence "The user adds tasks to the project". According to WordNet, the glosses for senses of 'project' can be: 'project$_1$' project$_2$', and the context word has two senses: 'task$_1$' and 'task$_2$', as depicted in Figure 4. The algorithm counts words co-occurring in all combination of senses, and picks those senses that maximize the count. Here, 'project$_1$' and 'task$_2$' are the senses chosen.

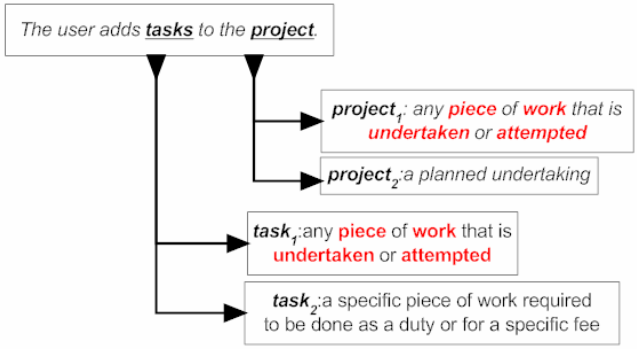

Figure 4 – Using the Lesk distance metric with WordNet

Regarding the *Graph Generation* block, it consists of four tasks: rules for pattern matching, similarity analysis for verbs/objects, dynamic clustering of verbs/objects, and construction of the AOIG itself. By means of pattern matching, pairs of verbs and direct objects or single verbs are detected in each sentence (remember that all the words were tagged during NLP analysis). After that, we rely on the results of the disambiguation algorithm and WordNet for identifying similar verbs and objects. Given two concepts (either a pair of verbs or a pair of objects), they are compared according to the number of terms shared by the concepts. For verbs, the hierarchies of troponyms and hypernyms of WordNet are used to span the words represented by the verbs. For objects (nouns), the hierarchies of hypernyms and hyponyms of WordNet are used to span the words represented by the nouns. This spanning process is graphically exemplified in Figure 5. As the text of the use cases is being analyzed, their verbs and objects are added to different groups (clusters). At last, all the verbs, objects and clusters are mapped to corresponding AOIG nodes, and integrated into a graph like the one shown in Figure 2.

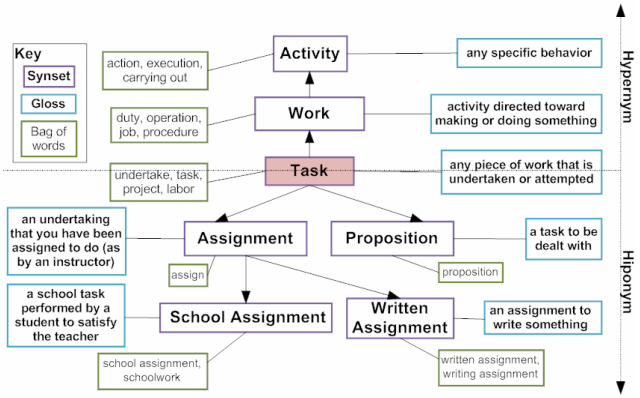

Figure 5 - Expanding the context of a word

The *Aspect Finder* block takes the outputs for the two previous blocks and performs two tasks on the AOIG: graph traversal and ranking of candidate early aspects. When traversing the graph, the hint nodes that affect more than N use cases (the threshold N is configured in the tool) are marked for further analysis. The fact that a hint node affects several use cases is an evidence of a crosscutting concern. The most representative verb of a hint node is consequently suggested as candidate concern. For instance, hint nodes (VG1,DOG3), (VG3,DOG2) and (VG2,DOG2) are proposed as candidate concerns. Based on the source of the target verb (i.e., the sections of the use case template), node (VG1,DOG3) is categorized as a quality-attribute concern, whereas nodes (VG3,DOG2) and (VG2,DOG2) are categorized as functional concerns. A heuristic ranks the concerns detected in the AOIG on the basis of parameters such as: number of crosscut use cases per concern, total crosscutting count per verb cluster, verb cluster occurrence per concern, object cluster occurrence per concern, and functional/quality-attribute property of the concern.

5. PRELIMINARY EVALUATION

As a proof-of-concept, we have evaluated the techniques of our approach in two case-studies. The first case-study was a student management system (SMS) whose requirements specification consisted of 3 use-cases and 3 specification pages. The second case was a conference evaluation system (CES), involving 9 use-cases and 15 specification pages. In order to identify early aspects, we performed three experiments with different tools. In the first experiment, we applied the Aspect Extractor Tool (AET) [9], while in the remaining experiments with applied the techniques described in this work. AET applies stop words and

stemming algorithms, and it basically suggests candidate aspects when a verb is shared by two or more use cases. Our technique was exercised with two configurations of the clustering algorithm, setting parameters for minimum (T1) and maximal (T2) effectiveness. Table 1 summarizes the numbers collected from the experiments. The row 'True Positives' refers to those identified aspects that were considered as relevant ones by the user. The row 'False Positives' refers to identified aspects that were discarded by the user. The row 'False Negatives' are those aspects that should have been identified but were missed by the technique. We took typical measures such as Precision, Recall and F-measure [5]. We also measured the time consumed (speed) by the aspect identification process for each experiment.

The results were encouraging, as it can be seen from Table 1, with the exception of speed. We observed an increase in recall, when moving from the first to the third experiments. This was due to two factors: the identification of concrete functional aspects, and the correct discovery of relationships between aspects and use cases. The functional aspects primarily came from specific functionality scattered across the use cases, generally caused by bad modularization of functionality. We also noted that both the disambiguation and clustering techniques had a better performance with larger and complete use-case specifications, presumably because of the information amount in these specifications. We observed that the use of NLP and WSD requires almost no extra information manually entered by the analyst. Thus, the aspect identification can be automated to a larger degree than in existing approaches.

	SMS			CES		
	AET	T1	T2	AET	T1	T2
True Positives	5	6	5	6	13	18
False Positives	5	3	3	9	18	22
False Negative	3	2	3	14	7	2
Recall	0,62	0,75	0,62	0,3	0,65	0,9
Precision	0,5	0,67	0,62	0,4	0,42	0,45
F-Measure	0,56	0,71	0,62	0,34	0,51	0,6
Time (speed)	2s.	30s.	1min.	5s.	60min.	90min.

Table 1 – Results of several aspect identification experiments

Nonetheless, the main drawback of the proposed technique is still its processing time. The time consumed by the analysis of concerns took more than expected. In general, this performance can be admissible or not, depending on the precision/ recall levels needed by the analyst and on the sizes of the use-case specifications. From the differences between the second and third experiments, we see that the more we adjust the parameters for the dynamic clustering, the better results we get regarding recall and small false negatives. However, this improvement comes at the cost of performance. Along this line, we have started to investigate possible optimizations and filters for the aspect identification process described in Section 4.

6. CONCLUSION

In this paper, we have discussed an approach for the identification of candidate early aspects from requirements specifications in the form of use cases. A novel characteristic of this approach is the semantic analysis of textual requirements via NLP and WSD techniques. Our semi-automated analysis is centered on the relationships among terms in use cases (e.g., verbs, direct objects) that often hint crosscutting behaviors. We believe that these techniques can solve issues related to synonyms, vagueness and ambiguity in text, as reported by other aspect mining approaches.

Future areas of improvement for this approach include: how to enhance word clustering, how to add domain-specific filters for reducing the aspect list, and how to implement strategies for increasing the performance of our prototype, among others. In addition, we are planning to evaluate this technique against other techniques, based on a number of case-studies.

7. REFERENCES

[1] *AOSD Net*. 2008. DOI= http://aosd.net/

[2] *Early Aspects*. 2008. DOI= http://www.early-aspects.net/

[3] *OpenNLP*. 2008 DOI= http://sourceforge.net/projects/opennlp

[4] WordNet. 2008 DOI= http://wordnet.princeton.edu/

[5] Baeza-Yates, R. and B. Ribeiro-Neto. 1999 *Modern Information Retrieval*. Addison-Wesley (Wokingham, UK, 1999)

[6] Baniassad, E., Clarke, S. 2004 *Finding Aspects in Requirements with Theme/Doc*. In Workshop on EA, in conjunction with AOSD Conference. (Lancaster, UK, 2004).

[7] Baniassad, E., Clarke, S. 2004 *Theme: An Approach for Aspect-Oriented Analysis and Design*. In Proc. of ICSE'04.

[8] Cockburn, A. 2000. *Writing effective use cases*. Addison-Wesley (October 15, 2000).

[9] Haak, B., Diaz, A., Pryor, J., Marcos, C. 2005 *Identificación Temprana de Aspectos*. Revista SCCC, 2005. Vol. 6 (Workshop in SE). In Spanish.

[10] Kruchten, P. 2004 *The Rational Unified Process: An Introduction*, 3rd Edition. Boston, MA: Addison-Wesley.

[11] Lesk, M. 1986 *Automatic Sense Disambiguation Using Machine Readable Dictionaries*. In Proc. of the 5th Annual Int. Conference on Systems Documentation (Toronto, Ontario 1986).

[12] Li, K., et al. 2005 *Object-Oriented Analysis Using Natural Language*. In Proc. of the ICYCS '05. (Beijing, China 2005).

[13] Pedersen, T., Banerjee, S., Patwardhan, S. 2005 *Maximizing Semantic Relatedness to Perform Word Sense Disambiguation*. University of Minnesota Super-computing Institute.

[14] Rashid, A., Moreira, A., Araújo, J. 2003 *Modularization and Composition of Aspectual Requirements*. In Proceedings of the AOSD'03. (Boston, Massachusetts 2003).

[15] Sampaio, A., et al. 2005 EA-Miner: a Tool for Automating Aspect-Oriented Requirements Identification. in Proc. of the 20th IEEE/ACM ICASE '05. (California, USA 2005).

[16] Rosenhainer, L. 2004 *Identifying Crosscutting Concerns in Requirements Specifications*. In Workshop on EA, in conjunction with OOPSLA 2004. (Vancouver, Canada 2004).

[17] Shepherd, D., Pollock, L., Tourwé. T. 2005 *Using Language Clues to Discover Crosscutting Concerns*. In Proc. of the Int. Workshop on MACS (ICSE'05). (St. Louis, Missouri 2005).

[18] Shepherd, D., Pollock, L., Vijay-Shanker, K. 2006 *Towards Supporting On-Demand Virtual Remodularization Using Program Graph*s. In Proc. of AOSD'06. (Bonn, Germany 2006).