# Emergent Architectural Component Characterization using Semantic Web Technologies

Pablo Inostroza and Hernán Astudillo

Departamento de Informática,
Universidad Técnica Federico Santa María
Avenida España 1680, Valparaíso - Chile
{pinostro,hernan}@inf.utfsm.cl

**Abstract.** Selecting components for systems development is a hard task that depends chiefly on quality of component characterizations, yet existing Web-based catalogs do not offer rigorous component descriptions, fitness to purpose, or composition mechanisms. This paper introduces an approach to abstraction-based components characterization, allowing evaluation against higher level architectural notions such as NFRs. A characterization evaluation mechanism enables community-based ratings of descriptions, with catalog semantics emerging from distributed, unrelated descriptions. The Semantic Web-based approach implementation combines a central, authority-based conceptual framework with distributed, confidence-based catalog definition and valuation. Some issues regarding trust models and large scale integration are explored and a prototype is introduced.

**Key words:** CBD; software architecture; Semantic Web; ontologies; knowledge bases; social networks.

## 1 Introduction

Component Based Software Development (CBD) aims to optimize software development by reusing existing components to build new systems, attending to benefits like shorter development times, lower costs and higher product quality. Thus, it is clear that components selection and description are fundamental tasks. There are many distributed catalogs on the web, many of which give a not rigorous component descriptions. On the other hand, majority of these does not consider non functional information about components, such as reliability, security, among others, commonly known as NFRs (Non Functional Requierements).

There are two possibilities to accomplish component characterization from an architectural viewpoint. It is possible a rigorous modeling of NFRs and their relationships to components, which could be guided by an expert architectural community. In this way, many economical resources should be destined to ontological engineering. On the other hand, it is possible to have many component characterizations from a more distributed community. We believe these two approaches have their advantages and disadvantages. Ontological Engineering approaches, yet rigorous, is a knowledge intensive process that becomes unattainable when the goal is to model all components, specially considering these are constantly updated to new versions and new components appear on the market. Distributed components characterization have problems related with trustworthiness of assertions, vocabulary consensus, among others. In this paper we take an intermediate approach, conceiving an ontology that allows to link high level architectural requirements with components through architectural abstractions. Knowledge intensive high level assertions are provided by a reduced expert community, while components characterization emerge from a wider distributed community.

To address the complex scenario we described, we adopted an strategy which consist in augment component descriptions with semantic metadata about NFRs following Azimut approach, not just from one central authority but also from different distributed communities. Modeling this situation not only implies consider architectural and design level abstractions, but also reflects an authority model that allows to infer trustworthiness, completeness among other considerations. Semantic Web technologies seems to be an appropriated way to implement our approach, since they provide an explicit way to formalize a domain taking advantage of another vocabularies descriptions distributed across the Web.

The paper is structured as follows. Section 2 presents some related work in NFRs characterization and Semantic Web based component description. Section 3 introduces Azimut framework and ontology that formalize it, while section 4 describes how architectural component characterizations emerges from distributed assertions. In section 5 there are some discussions about contributions of this article and future work. Finally, section 6 concludes this paper.

## 2 Related Work

There are several approaches to describe functional aspects of software components using Semantic Web technology. Some of these works like [1], [2], [3] and [4] have explored Semantic Web based descriptions to capture functionality descriptions of components, mainly based in UML or interfaces description. Although some of these works refers to the notion of NFRs, how to assert a link between them and component descriptions are not established.

In context of software component reuse, Yao et. al [5] presents a semantic-Based approach for software reusable component classification and retrieval, that mainly relies on mapping components to web services and then list them in a UDDI repository. Unfortunately, this work does not consider NFRs either.

On the other hand, there are approaches to describe component from a NFR perspective. Iribarne et. al [6] propose a XML Schema to capture as much functional nature as non-functional nature of a Software Component. Unfortunately, XML Schema does not provide a strong mechanism to reference resources, and its tree-based structure make difficult to assert graph style assertions, typical in knowledge representation structures. Tying functional and non functional descriptions within an unique schema is not a good idea, since these two descriptions could evolve from different communities.

## 3 Azimut Ontology

### 3.1 Main Concepts

Azimut Framework [7] [8] (see figure 1 for a general vision) aims to support architects in generating component assemblies for a given set of requirements, by using a derivation chain, in which defined stepping stones have a fundamental role. To characterize such *stones*, Azimut use a vocabulary taken from the distributed systems community [9], duly adapted to the software architecture context.

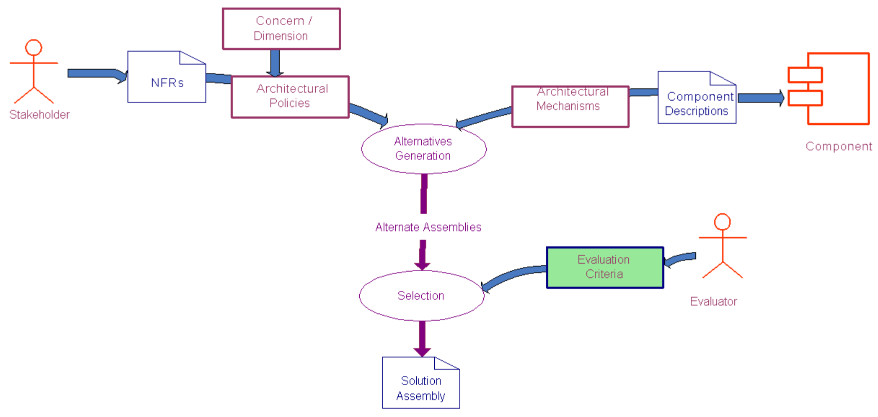


**Fig. 1.** Azimut Framework

– *Architectural Policy*: The first reification from NFRs to architectural concepts. Architectural policies can be characterized through specific concern dimensions that allow describing NFRs with more details. For example, in communication concern it is possible distinguish a synchrony dimension where it is possible to find asynchronous and synchronous policies. It is relevant that there are policies sets for a specific *Architectural Concern*, where is possible to distinguish *Architectural Dimensions*. Concerns and Dimensions allows to refine policies characterization using widely known Software Architecture constructs. It is even possible to consider a higher abstraction level, considering that a concern is related to an *Architectural Quality Attribute*, for instance, access control concern is related to Security quality attribute. Policy catalogs should gathers platform-independent architectural policies and stores dimensions for each concern and policies that have different values for each dimension. It incorporates knowledge for each architectural concern, and the dimensions themselves are collected from authoritative sources of the relevant discipline (e.g. Tanenbaum [10] for replication, Britton [11] for middleware communication, and Firesmith [12] for security).

- *Architectural Mechanism*: The constructs that satisfy architectural policies. Different mechanisms can satisfy the same architectural policy, and the differences between mechanisms is the way in which they provide certain dimensions. As an example, SMTP (the standard e-mail protocol) mechanism supports asynchronous communication policy.
- *Component*: A software artifact that is executable, multiple-use, non-context specific, composable with other components, encapsulated (i.e., non-investigable through their interfaces), and a unit of independent deployment and versioning [13]. Considering last examples, SendMail component provide SMTP mechanism, which as well supports synchronous policy.

### 3.2 Key Classes

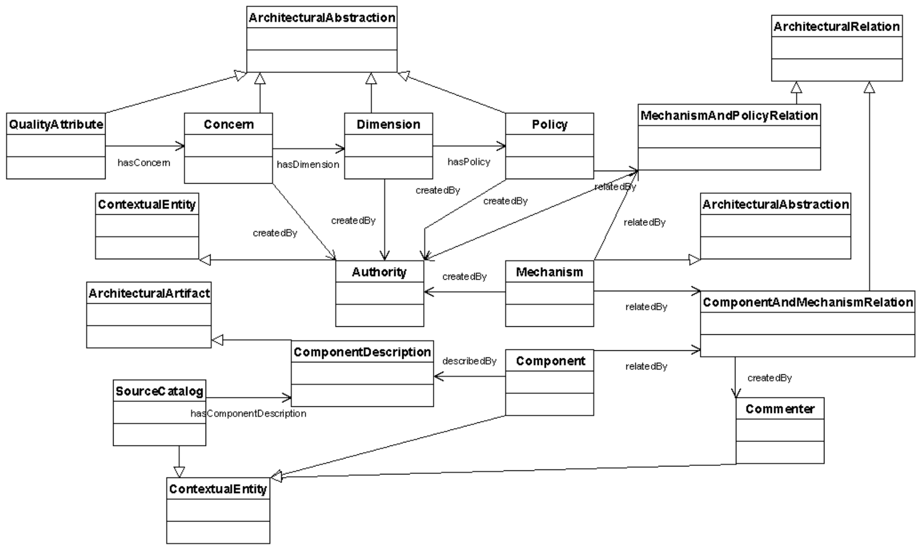


**Fig. 2.** Azimut Software Architecture Ontology

We extended and refine Azimut concepts presented in subsection 3.1 to produce a Software Architecture ontology that could support explicitly the implementation of policies, mechanisms and component catalogs considering authority model we briefly introduce next(see figure 2 for a high level vision of ontology).

Analyzing Azimut view-of-world, we classify their entities in four classes:

- *Architectural Abstraction:* This element includes key abstractions we formulated to understand the high level requirements from an architectural perspective. These concepts, introduced in [7], are the core of the ontology, since they introduce a layered approach to fill the gap between NFRs and lower level elements. Quality attributes, concerns, dimensions, policies and mechanisms are architectural abstractions.
- *Contextual Entity:* A contextual entity represents any concrete and real world element related with an architectural characterization. Until this point we have identified Components, which are contextual entities.
- *Architectural Artifact:* An architectural artifact is any formalized description of contextual entities. It is important to distinguish between an architectural artifact and a contextual entity: while the former represents a concrete person, organization or artifact from the real world, the latter describes it through formalized content. We have identified *ComponentDescription* as a part of this class. A Component Description is any information, possibly in natural language, that gives a description for a component.
- *Architectural Relation:* This entity kind does not arise from the domain, but is an artifact of ontology modeling. It merely denotes a “link” between two concepts, such as two architectural abstractions, or between an architectural abstraction and a contextual entity. At this moment, there are two subclasses of this entity: *ComponentAndMechanismRelation* and *PolicyAndMechanismRelation*. This entity is useful to capture some additional information about generated links. For example, an instance of ComponentAndMechanismRelation should capture some information such as who is the relation creator or some fuzzy values, such as the degree of support provided by the component (we discuss this in following sections).

We have added to Azimut ontology some constructs to model authority relations. Azimut ontology-based catalogs are augmented anytime an architectural abstraction, contextual entity or relation classes are instantiated. However, while some updates come from an authority entity (such as abstractions updates) another should be managed in a distributed way by a community. Thus, we distinguish two more subclasses of a contextual entity: *Authorities*, who represents the agents (organizations or persons) in charge of high level constructs updates, and *Commenters*, which are essentially distributed agents that could characterize components linking them with established mechanisms. We will show later how this notions help us to model reliability and completeness of component descriptions.

It is interesting to note that Components Descriptions could be added within Azimut framework or imported from external descriptions (see section 4.3 for details). For this reason, we conceive another contextual entity, the *SourceCatalogs* which records information about distributed catalogs that feed Azimut knowledge base.

### 3.3 Policies Characterizarion

There are two key concepts within Azimut view of world: mechanisms and policies and they must be related to reduce semantic gap between architectural requirements and low level artifacts. Policies must be characterized linking them with concerns, dimensions, quality attributes and mechanisms. An architectural community of experts have to capture those relationships, supported by authoritative sources of different disciplines related to their characterizations. This paper does not talk about how to characterize policies along with another high level architectural abstractions. In present work, those characterizations are assumed as given.

### 3.4 Expressiveness Considerations

Before developing Azimut Ontology it was necessary to choose one ontology language. RDF Schema was considered in a first stage because it represents a lightweight representation approach. However, there were requirements that requires a more expressive language. For instance, as we will see in subsection 4.3, it was necessary to establish identity reasoning. We decided to use OWL DL, the more expressive yet computable subset of Web Ontology Language (OWL) [14]. This decision will allows to extend current ontology to support more inference power.

## 4 Emergent Component Characterization

Once high level concepts and relationships are modeled and populated by an expert architectural community into Azimut ontology, it becomes necessary to establish links with components that could satisfy formalized requirements to make possible scenarios like presented in section 3.

In figure 3 it is possible to observe classes involved in this process. The *ComponentAndMechanismRelation* class relates three kinds each of which belongs to a different conceptual space. Next we present these three spaces and criteria that could emerge from their relation (see figure 4).

### 4.1 Mechanisms Space

Mechanisms are identified by the expert central authority, relating them with Architectural Policies to reduce conceptual gap between NFRs and components. For component characterization it is assumed that some mechanisms are identified and their relationships with policies are given. To improve mechanism and components semi automated linking we consider that just a label per mechanism is not enough. In Azimut ontology there are a set of related terms to each mechanism. This terms are also provided by the expert authority. SKOS lightweight ontology [15] are used to this purpose using *skos:prefLabel* and *skos:altLabel* properties to link more than one label to a mechanism. Multilingual labeling

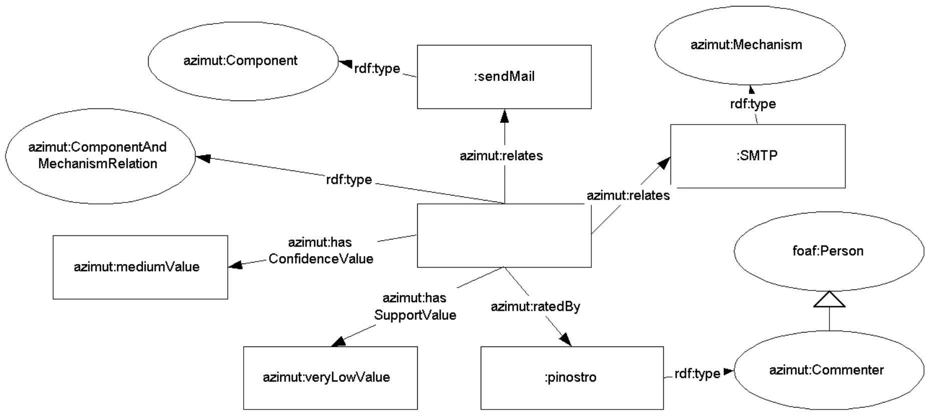


**Fig. 3.** Ontology concepts involved in a Mechanism-Component support assertion

support provided by SKOS allows even to relate non english labels to a mechanism. In figure 5 it is possible to observe that SMTP mechanism is also a SKOS concept, which have three labels: SMTP, Simple Mail Transfer Protocol and RFC2821, the document that define it. Thus, when any of these patterns are found in components descriptions, a link with a mechanism should be established, as we will show more ahead.

## 4.2 Commenters Space

Commenters community conforms a social network [16]. There are some works to deal with social networks in Semantic Web using FOAF ontology [17][18]. We consider any commenter is a subclass of FOAF Agent class, making possible to do some Social Network Analysis (SNA) [16] on community of commenters. To enrich component characterization and give to any Azimut user a personal perspective of components we have considered Trust notion within presented framework, allowing to users rank proposed components depending on people who characterize them. Golbeck proposed an extension to FOAF called Trust module [19] that extend this vocabulary providing a set of properties that allows to assert trust level between two agents.

## 4.3 Components Space

This space is conformed by all identified components. It provide consolidated set of URIs for components so it could be used for other projects that make another assertions about components, for example, about their functionality. Because a component must be unique, if it is desired to add different component descriptions to a component, *ComponentDescription* class should be used to capture them. Some relevant properties of this class are *hasText* that contains the natural language prescription and the *hasLanguage* property that relates description

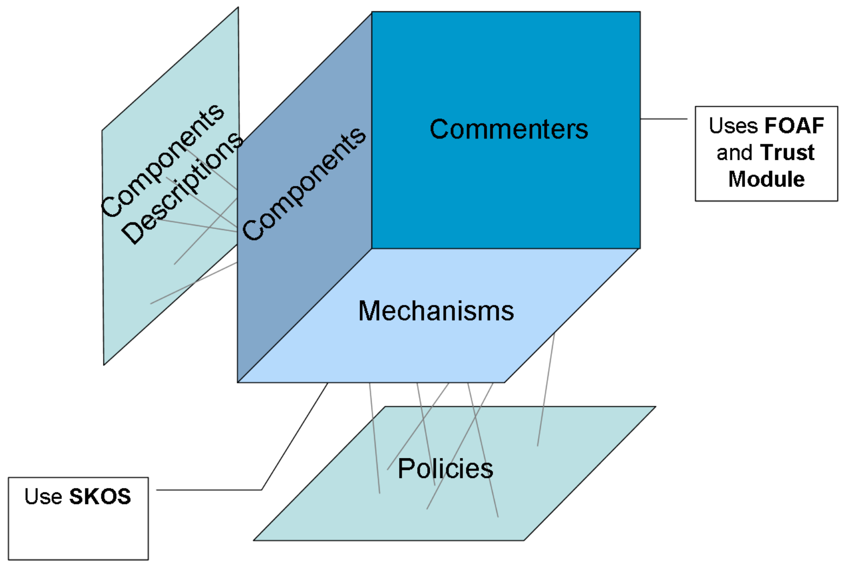


**Fig. 4.** Three dimensions involved in Component Characterization

to a particular language. Thus, Components Space is supported by Component Descriptions Space, allowing to integrate heterogeneous and distributed descriptions in a coherent framework. There are three strategies to populate components and components descriptions:

- *Wrappers*: It is possible to build some wrapper on some existing catalog as eCots [1], exporting components described in its schema to RDF Azimut compliant component description (possibly using XSLT). Disadvantages of this approach are related to querying facilities, since the RDF data are generated on demand. Another emerging approach to complement wrappers is GRDDL (Gleaning Resource Descriptions from Dialects of Languages) [20], that allows to explicitly annotate web pages and XML documents with semantic content, avoiding inefficient tasks as screen scraping. Wrappers must be implemented by catalog providers, so their use depends on which the cataloguers adopt Azimut concepts.
- *Crawling*: It is possible to crawl through existing catalogs XML documents, or worse, through existing catalogs web pages using screen scraping. Once initial data is collected, it could be transformed to RDF Azimut compliant component descriptions and then stored in an RDF native repository. The RDF base must be updated through scheduled crawling tasks.
- *Directly populating*: Of course, there is possibly to directly populate Components Descriptions on RDF native repository that supports Azimut system. Perhaps authorities or commenters could add their components descriptions, yet this possibility is in evaluation.

[1] eCots - Software Components Open Directory Project: http://www.ecots.org

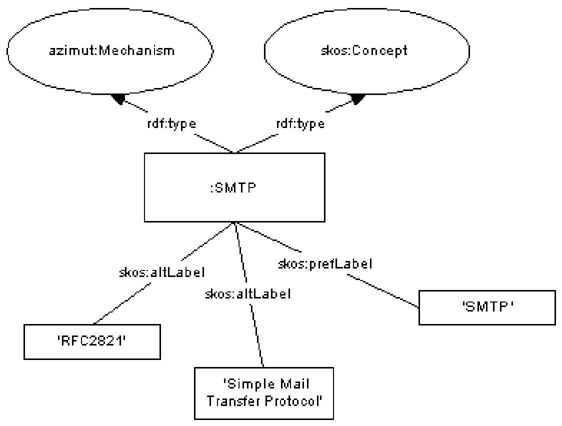


**Fig. 5.** Describin a mechanism using Azimut and SKOS ontologies

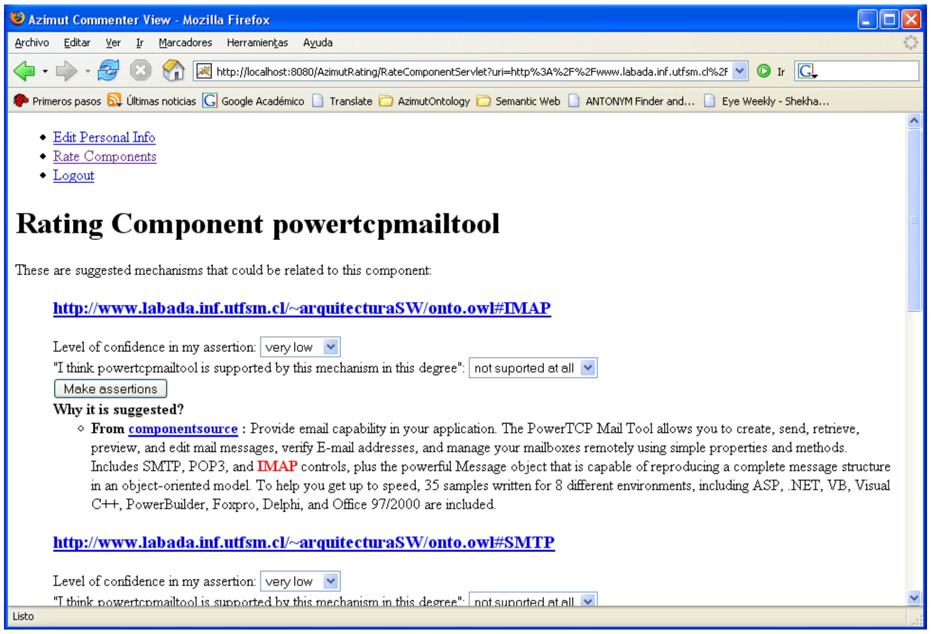
When new descriptions are loaded into Azimut knowledge base, it is possible to try to describe an existing component. To semantically establish that a new component description refers to an existing component, the Azimut system administrator could note this situation and resolve the ambiguity. Identity reasoning is required to determine the identity of individuals across multiple information sources. Then, owl:sameAs property could capture identity relation between two described components. To help administrator to establsh those assertions, it is possible to consider a NLP comparison between new and existing descriptions of components to suggest him the possible disambiguation that is needed.

### 4.4 Characterizing components

The process of characterize a component is described next:

1. User selects from a set of components which he wants to rate (until now there is only a syntactic components search).
2. System presents a list of possible related mechanism to help user. To make this, NLP techniques are used to match different integrated component descriptions with mechanisms controlled vocabulary supported by SKOS (see figure 6).
3. User selects a mechanism and link it with selected component, establishing two values: self confidence and support value.

Once there are high level descriptions provided by expert authority and components descriptions from distributed community, it is possible to query Azimut system to satisfy some architectural non functional requirements to obtain set of components that could satisfies them. The degree in which a component satisfy an architectural policy is produced by a chain of two links. Policy-mechanism link is established by an authority using Azimut ontology, while mechanism-component link is characterized by many assertions from different commenters. What is more, each commenter have different self confidence values for his different assertions and provide different support values for distinct mechanism-component relationships. It is possible to consider aggregated criteria values that combine all of this independent information to conform a global opinion about how a component supports a mechanism. An interesting extension to this approach is to complement those metric with trust notion introduced in subsection 4.2. Thus, each user could filter the information based on trust values inferred from his social network.


Azimut Commenter View - Mozilla Firefox
Archivo Editar Ver Ir Marcadores Herramientas Ayuda
http://localhost:8080/AzimutRating/RateComponentServlet?uri=http%3A%2F%2Fwww.labada.inf.utfsm.cl%2F
Ir
Primeros pasos Últimas noticias Google Académico Translate AzimutOntology Semantic Web ANTONYM Finder and... Eye Weekly - Shekha...
Listo

- Edit Personal Info
- Rate Components
- Logout

Rating Component powertcpmailtool

These are suggested mechanisms that could be related to this component:

http://www.labada.inf.utfsm.cl/~arquitecturaSW/onto.owl#IMAP

Level of confidence in my assertion: very low

"I think powertcpmailtool is supported by this mechanism in this degree": not suported at all

Make assertions

**Why it is suggested?**

- **From componentsource** : Provide email capability in your application. The PowerTCP Mail Tool allows you to create, send, retrieve, preview, and edit mail messages, verify E-mail addresses, and manage your mailboxes remotely using simple properties and methods. Includes SMTP, POP3, and IMAP controls, plus the powerful Message object that is capable of reproducing a complete message structure in an object-oriented model. To help you get up to speed, 35 samples written for 8 different environments, including ASP, .NET, VB, Visual C++, PowerBuilder, Foxpro, Delphi, and Office 97/2000 are included.

http://www.labada.inf.utfsm.cl/~arquitecturaSW/onto.owl#SMTP

Level of confidence in my assertion: very low

"I think powertcpmailtool is supported by this mechanism in this degree": not suported at all

**Fig. 6.** Making component-mechanism assertion in prototype

## 5 Discussions

We think Semantic Web based approach to components description allows to enrich their descriptions, aiming to reach very interesting analysis, such as SNA, trustworthiness,among others. Having all this viewpoints integrated in a common conceptual framework allows to improve quality of descriptions, one of our main objective. Global analysis of annotated components in any of these senses has been considered as part of our future work, yet this paper has suggest that this could be achieved resorting to existing works as [19] or [17]. In fact, the main contribution of this work consists in using different level of expressiveness according to what subset of the domain is being modeled, integrating different specific vocabularies under the Semantic Web umbrella. In this case, a thesaurus vocabulary like SKOS, social networks concepts and NLP lightweight techniques converge in the goal of characterize components from an architectural perspective. This approach focuses on extensibility, allowing new vocabularies could be added to complement existing descriptions in a natural way.

It is interesting to note that presented Azimut constructs, like *PolicyAndMechanismRelation* are fundamental to provide derivation from Architectural requirements to components, yet present work does not address this particular business. Selecting components involves different tasks such as map end-users queries to Architectural Policies or assert sophisticated mechanism-policy relationships. Those are part of ongoing work.

There are some important considerations that are very important if we want Azimut ideas will be used. Commenters interface must be attractive and easy to use to motive users to feed the knowledge base with components-mechanism and social ratings. We are working in an AJAX UI that aims to suggest some elements to the users rate them. As the rating information is requested in a more friendly and attractive way, the number and quality of assertions recorded in the Azimut system will be increasing.

## 6 Conclussions

This paper has presented a global vision of Semantic Web based approach to describe components within Azimut conceptualization. While knowledge intensive ontology engineering process has been proposed to assert the more complex or specific relations between high level architectural notions, components characterization emerges from a distributed and flexible annotation process, improved by the reuse of existing semantic vocabularies. This choice allows to explore interesting and different kinds of analysis on semantically consolidated data.

## References

1. S, J.G., Leicher, A.: Augmenting domain specific UML models with RDF. In: 3rd Workshop in Software Model Engineering (WISME), Lisbon, Portugal (2004)
2. Melnick, S.: Representing UML in RDF. Technical report (2000)
3. Cranefield, S.: UML and the Semantic Web. In: Proceedings of the International Semantic Web Working Symposium (SWWS). (2001) http://www.semanticweb.org/SWWS/program/full/paper1.pdf.
4. Korthaus, A., Schwind, M., Seedorf, S.: Semantic integration of business component specifications with RDF schema. In: Workshop on Sematic Web Enabled Software Engineering (SWESE), 4th International Semantic Web Conference (ISWC 2005), Galway, Ireland (2005)
5. Yao, H., Etzkorn, L.: Towards a semantic-based approach for software reusable component classification and retrieval. In: ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference, New York, NY, USA, ACM Press (2004) 110–115

6. Iribarne, L., Vallecillo, A., Alves, C., Castro, J.: A non-functional approach for COTS components trading. In: WER. (2001) 124–138
7. López, C., Astudillo, H.: Explicit architectural policies to satisfy nfrs using cots. In Bruel, J.M., ed.: Lecture Notes in Computer Science, Satellite Events at the MoDELS 2005 Conference: MoDELS 2005. Volume 3844. (2006) 227 – 236
8. López, C., Astudillo, H.: Multidimensional catalogs for systematic exploration of component-based design spaces. In: IWASE 2006: First International Workshop on Advanced Software Engineering. Proceedings of IFIP World Congress 2006. (2006)
9. : Policy and mechanism definitions. (`http://wiki.cs.uiuc.edu/MFA/Policy+and+Mechanism`)
10. Tanenbaum, A.S., van Steen, M.: Distributed Systems: Principles and Paradigms. First edn. Prentice Hall (2002)
11. Britton, C., Bye, P.: IT Architectures and Middleware: Strategies for Building Large, Integrated Systems. Second edn. Addison-Wesley Professional (2004)
12. Firesmith, D.: Specifying reusable security requirements. Journal of Object Technology **3**(1) (2004) 61–75
13. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2002)
14. McGuinness, D.L., van Harmelen, F.: OWL web ontology language: Overview. W3c recommendation, World Wide Web Consortium (2003)
15. Miles, A., Brickley, D.: SKOS core guide. W3c working draft, World Wide Web Consortium (2005)
16. Staab, S., Domingos, P., Mika, P., Golbeck, J., Ding, L., Finin, T., Joshi, A., Nowak, A., Vallacher, R.R.: Social networks applied. IEEE Intelligent Systems **20**(1) (2005) 80–93
17. Mika, P.: Flink: Semantic web technology for the extraction and analysis of social networks. Web Semantics: Science, Services and Agents on the World Wide Web **3**(2-3) (2005) 211–223
18. Brickley, D., Miller, L.: FOAF vocabulary specification. Technical report (2005)
19. Golbeck, J.: Computing and Applying Trust in Web-based Social Networks. PhD thesis, University of Maryland (2005)
20. Hazal-Massieux, D., Conolly, D.: Gleaning resource descriptions from dialects of languages (GRDDL). W3c team submission, World Wide Web Consortium (2005)