Highlighting Stakeholder Communities to Support Requirements Decision-Making*

Zeina Azmeh, Isabelle Mirbel, and Pierre Crescenzo

I3S Laboratory, CNRS UMR 7271
University of Nice Sophia Antipolis, France
{firstname.lastname}@unice.fr

Abstract. [Context & motivation] Stakeholders participation is recognized as a key issue in the development of useful and usable systems. The Web has given rise to a growing number of collaborative working tools that facilitated the participation of stakeholders (and especially end-users). These tools create new opportunities of practice regarding requirement elicitation. **[Question/problem]** Nevertheless, they result in an information overload lacking structure and semantics. Consequently, requirements analysis and selection becomes more challenging. **[Principal ideas/results]** In this paper, we propose an approach based on semantic web languages as well as concept lattices to identify relevant groups of stakeholders depending on their past participation. **[Contribution]** These groups can be used to enable facilitated decision-making and handling of requirements. We detail the different steps and the possible configurations, using an example inspired by a collaborative software development environment.

Keywords: Stakeholder communities, concept lattices, requirements elicitation.

1 Introduction

Requirements engineering is an essential process of software engineering, during which, the complete behavior of a software system can be defined. The success of this process plays a crucial role in the success of the whole software project. A part of this success is achieved by the good selection of pertinent stakeholders, and by the proper understanding of their particular needs, in a core activity called requirements elicitation. Stakeholders participation is thus recognized as a key issue in the development of useful and usable systems, which can be hard to attain efficiently. The Web has given rise to several platforms serving the purpose of collaborative software development [3]. These online platforms enable the covering of a larger number of stakeholders that are able to express their needs freely online. The problem lies in the large number of requirements that need to be handled. Deciding on these requirements can not be done in a straightforward manner, especially with the poor stakeholder profiles that are not helpful for evaluating neither the stakeholders nor their requirements. This in addition to the fact that there is an overload of data generated by these stakeholders that is quite hard to process or to share, since it lacks structure and semantics. There is a need for a mechanism able to facilitate the selection of requirements to be analyzed, by knowing the past

* This work was supported by the DreamIT Foundation - University of Nice Sophia Antipolis.

activity of stakeholders who are involved in them. Stakeholders who were previously involved in accepted requirements, must be judged to have a higher priority over other stakeholders. They are intuitively more important than stakeholders who proposed only refused requirements, or proposed nothing at all.

We propose an approach for discovering communities of stakeholders to support requirements management (classification for instance) and decision-making (prioritization and potentiality of being accepted for instance). The approach works on deriving profiles for representing evaluated requirements, according to some values like priority and status. Then, it clusters stakeholders into communities according to their participation in requirements belonging to these profiles. This results in having a better overview of stakeholders and knowing in what profile of requirements they participated previously. Consequently, this helps to better evaluate their new requirements. The approach is based on semantic Web languages and concept lattices. We propose an ontology to represent the different actors and activities that are involved in collaborative software development environments. The objective of using semantic web languages is to annotate the user-generated data to enable a better understanding and sharing of knowledge [13], as well as the ability to reason about the data. Concept lattices are data structures that reveal the hidden relationships between the different entities of the contained data. They can be constructed using a method called Formal Concept Analysis (FCA) [10], which clusters a set of given objects into concepts, according to the attributes they share. The set of derived concepts are ordered into a lattice afterwards.

We explain our approach using an example inspired by a collaborative software development environment. We show how to analyze annotated data using concept lattices to extract stakeholder communities and we interpret the obtained results.

The paper is organized as follows: in the next section, we give an overview about concept lattices using Formal and Relational Concept Analysis (FCA, RCA). In Section 3, we present our approach and detail its different steps. In Section 4, we present a conducted experiment. In Section 5, we discuss the related work. Finally, in Section 6, we conclude the paper and describe our future work.

2 Background

In this section, we give the basic definitions of Formal and Relational Concept Analysis (FCA, RCA). We explain their use for the generation of concept lattices along with simple examples.

2.1 Formal Concept Analysis (FCA)

We base our approach on FCA [10] which is a classification method that permits the identification of groups of objects having common attributes. It takes a data set represented as an n×m table (formal context) with objects as rows and attributes as columns. A cross "×" in this table means that the corresponding object has the corresponding attribute. An example of a formal context is shown in Table 1, for a set of objects O={1,2,3,4,5,6,7,8,9,10} and a set of attributes A={odd,even,prime, composite,square}.

From a formal context, FCA extracts the set of all formal concepts. A formal concept is a maximal set of objects (called extent) sharing a maximal set of attributes (called intent). For example, in Table 1, a=({4,6,8,10},{even, composite}) is a formal concept because the objects 4, 6, 8, and 10 share exactly the attributes *even* and composite (and vice-versa). On the other hand, ({6},{even, composite}) is not a formal concept because the extent {6} is not maximal: other objects share the same set of attributes.

Table 1. A formal context for objects O and attributes A.

	odd	even	prime	composite	square
1	×				×
2		×	×		
3	×		×		
4		×		×	×
5	×		×		
6		×		×	
7	×		×		
8		×		×	
9	×			×	×
10		×		×	

FCA reveals the inheritance relations (super-concept and sub-concept) between the extracted concepts and organizes them into a partially ordered structure known as Galois lattice or concept lattice. The resulting concept lattice is illustrated in Fig. 1(L).

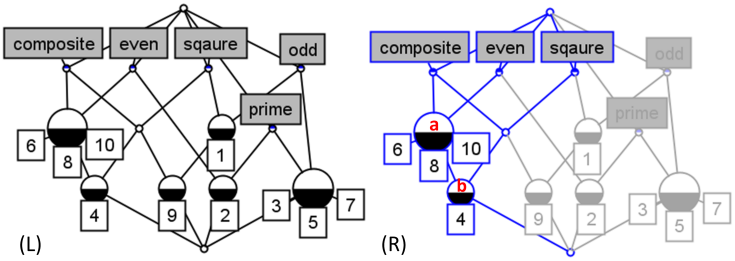

Fig. 1. Formal concept lattice[1] for the context in Table 1 (L); focus on the concept b (R).

This lattice reveals phenomena that may not be recognized intuitively. For example, in Fig. 1(R) appears the concept b=({4},{composite,even,square}) as a sub-

[1] Built using the Concept Explorer (ConExp) tool: http://conexp.sourceforge.net/users/index.html In a lattice, a full node indicates that the concept introduces objects and attributes that weren't introduced before; a half-full node introduces either objects when the bottom half is full, or attributes when the upper half is full; and an empty node represents an intermediary concept, which does not introduce any objects or attributes.

concept of the concept a. It inherits a's attributes composite and even, and extends it by the square.

2.2 Relational Concept Analysis (RCA)

RCA [14] is an extension of FCA that takes into consideration the relations between the objects. Thus, it takes as input two types of contexts: (non-relational) ones that are previously used with FCA to classify objects by attributes, and inter-context (relational) ones that represent the relations between the objects. RCA generates lattices similar to the ones generated by FCA, but enriched with the information about the relation between the objects. We take as an example two sets of numbers, {1,2,3,4,5} and {11,12,13,14,15,16,17,18,19,20}. We build two non-relational contexts similar to the one in Table 1. We consider a relation called Divides between the first and second sets of numbers, and we build the relational context in Table 2.

Table 2. The relational context Divides.

Divides	11	12	13	14	15	16	17	18	19	20
1	×	×	×	×	×	×	×	×	×	×
2		×		×		×		×		×
3		×			×			×		
4		×				×				×
5					×					×

RCA takes the two non-relational contexts (numbers×attributes), and the relational context Divides, then generates the two lattices in Fig. 2.

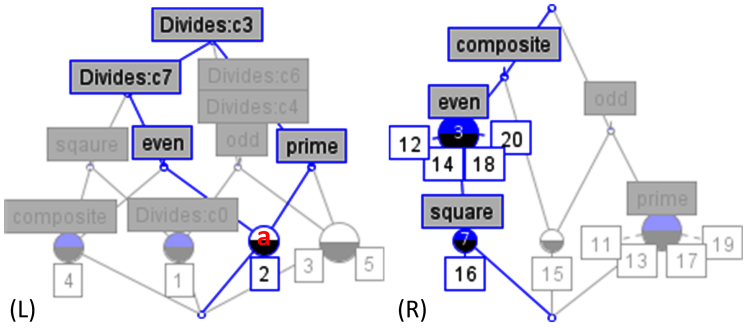

Fig. 2. The enriched lattices generated by RCA.

These lattices are similar to FCA lattices, but one of them is enriched with the relation *Divides*. For example, by regarding the concept a=({2},{prime,even, Divides:c7,Divides:c3}) in lattice (L), we notice that the numbers in its extent can divide the numbers in the extents of the concepts 7 and 3 in lattice (R).

3 Highlighting Stakeholder Communities

The objective of our approach is to discover stakeholder communities, according to the requirements in which they participate. The approach is composed of two main steps: structuring the data of a project, by semantically annotating the different actors and artifacts, together with the possible interactions between them; and analyzing this annotated data, using concept lattices to discover stakeholder communities.

We explain our approach along with an example that is inspired by a collaborative software development platform, called Launchpad [2]. In such platforms, stakeholders are themselves involved in various activities of the software development life-cycle, which may not be necessarily the case in platforms devoted to requirements elicitation. The different tasks performed by stakeholders result in rich information that we can exploit to discover groups of requirements and groups of stakeholders. On the other hand, requirements captured through this kind of platforms are low level requirements.

Indeed, Launchpad enables stakeholders of proposing blueprints (new functionalities that they require) and reporting bugs (existing functionalities that need to be enhanced or repaired). Every project in this platform has a set of artifacts like: blueprints, bugs, and code branches, as well as a set of stakeholders participating in these artifacts. This platform provides the ability to track blueprints and bugs, as well as code branches. A large collection of projects are being managed through this platform, we mention some featured projects like: MySQL, Ubuntu, Mozilla, etc.

Let us consider the simplified example in Fig. 3. It presents a sample of data that can be obtained from this platform, involving a set of stakeholders and their different activities performed on a set of blueprints.

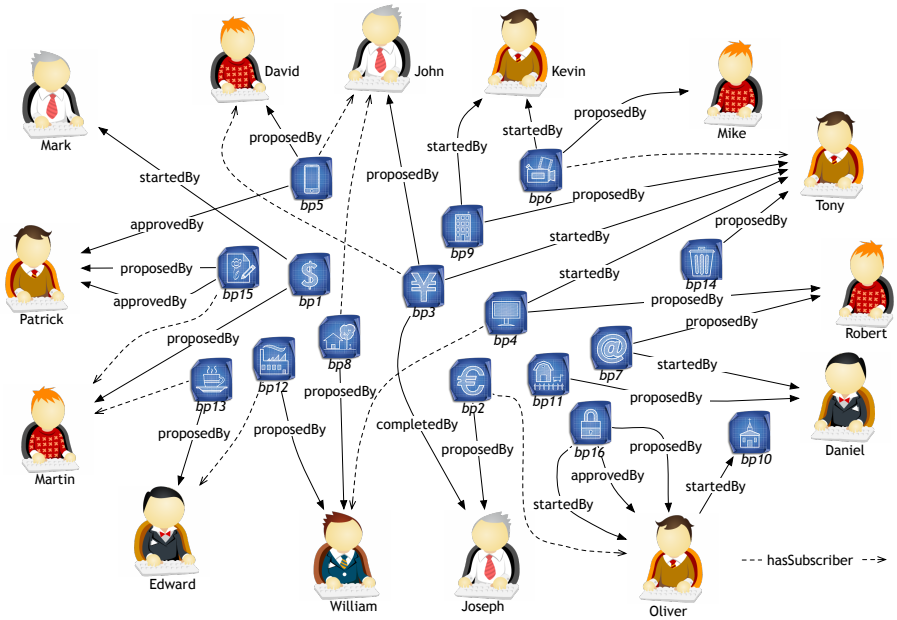

Fig. 3. An example of stakeholders performing different activities on a set of blueprints.

The stakeholders contributing to a project are not necessarily members of this project. Their profile pages give an overview about their personal information, which is usually very poor and insufficient for determining stakeholders importance. We can also have access to the list of artifacts, in which they participate. The artifacts also have profiles, in which we can find different attributes related to them like: status, importance, .., and other attributes indicating the involved stakeholders with their different activities.

The problem in this kind of platforms lies in the large number of blueprints and bugs that we can find for each project. Deciding on these requirements can not be done in a straightforward manner, especially with the poor stakeholder profiles that are not helpful for evaluating neither the stakeholders nor their requirements.

We propose to annotate semantically data from such a platform, using an ontology that we define and explain hereafter. Then we process the annotated data with concept lattices, to highlight stakeholder communities.

3.1 Ontology for Collaborative Software Development

We propose the ontology in Fig. 4 for collaborative software development (CSD) [13]. The advantage of annotating data from CSD environments with the help of such an ontology is embodied in the ability to share data across platforms. This is in addition to the ability to reason about the data, by exploiting classes and properties at different levels of granularity.

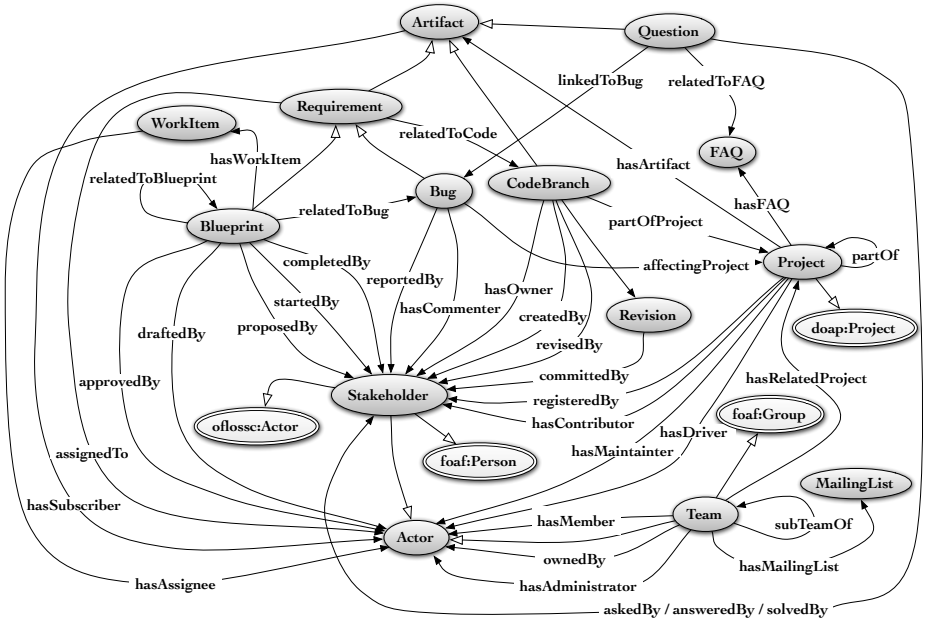

Fig. 4. An ontology for collaborative software development.

In this ontology, we represent the different actors (stakeholders and teams) and their interactions with the different artifacts of a software project.

This ontology is described using the RDF Schema vocabulary [5]. It makes use of several other ontologies like the FOAF vocabulary (Friend of a Friend) to describe stakeholders and groups [6], and the DOAP vocabulary (Description of a Project) to describe a software project itself with its various resources [1]. It is also connected to an ontology called OFLOSSC [17] that annotates community members and resources for open source development.

Every software project is annotated by the class *doap:Project*. It has a set of actors and a set of artifacts. Actors can either be individuals (stakeholders) or teams of individuals. A stakeholder is annotated by the class *foaf:Person*, while a team is annotated by the class *foaf:Group*. Project artifacts can be blueprints, bugs, code branches, or questions. We consider blueprints and bugs to be two different kinds of requirements. As we mentioned before, a blueprint is a proposal of a new functionality, while a bug is a proposal of enhancement of an existing functionality.

Using this ontology, we can reason about the annotated data. For example, a requirement is a coarse-grained artifact that can be replaced by either a blueprint or a bug to get further relationships.

As the matter of fact, it can be used to annotate data retrieved from almost any collaborative software development platform. For example, we developed a crawler that harvests data from Launchpad pages, and represents them in an RDF graph [16], conforming to the defined ontology.

We explain below the use of concept lattices to analyze the annotated data.

3.2 Using Concept Lattices

We mentioned above in Section 2 that concept lattices are data structures that reveal the hidden relationships between the different entities of the contained data.

The objective of using concept lattices is to extract hidden profiles for the set of requirements of a certain project, then to use these profiles for extracting stakeholder communities. Therefore, we make use of two kinds of information: the set of attributes defined for each requirement, as well as, the different interactions (participation) between stakeholders and the considered requirements.

Let us take again the example in Fig. 3. In this example, we have profile information for each blueprint (requirement), as well as stakeholder participation information. This different information is retrieved using the Launchpad crawler and is annotated in an RDF format. We analyze this annotated data according to two steps: blueprints explicit profile information is used to build a lattice of blueprints, which enable us to discover implicit blueprint profiles, following their common attributes; then, the identified blueprint profiles are used to classify the stakeholders according to their participation in these blueprints.

In Launchpad, an explicit (provided by stakeholders) blueprint profile consists of several attributes. These attributes can be numerical (like: the number of involved stakeholders) or nominal (non-numerical). Nominal attributes can further be divided into two types: ordinal attributes that have sortable enumerated values, and categorical attributes that have enumerated values with no ordering.

We suppose that a requirement analyst can specify his configurations of the attributes to consider. This includes specifying what attribute values are considered to be

equivalent, for example: a blueprint that has the status "new" or the status "under discussion" can be considered in the two cases to be "pending approval". An expert can also specify if the values of an attribute are ordinal or not, for example: the priority values of a blueprint can be specified as "low $<$ medium $<$ high".

In our running example, we consider values for priority and definition status only, for simplicity sake. These two attributes have sortable values: priority can take the values (undefined, low, medium, high), while definition status can take the values (unknown, approved, started, suspended, completed). In fact, we extract the information we need to analyze by querying the RDF data using the SPARQL query language [21].

Let us suppose having the formal context in Table 3, describing the set of blueprints by their values of priority and status. Since these attributes are ordinal, a blueprint in this formal context that has a high priority, covers also the other values of priority. This is also the case for the status values.

Table 3. Blueprints formal context.

	Priority				**Status**				
	high	*medium*	*low*	*undefined*	*completed*	*suspended*	*started*	*approved*	*unknown*
bp1	×	×	×	×	×	×	×	×	×
bp2	×	×	×	×		×	×	×	×
bp3	×	×	×	×			×	×	×
bp4		×	×	×				×	×
bp5		×	×	×				×	×
bp6		×	×	×			×	×	×
bp7		×	×	×	×	×	×	×	×
bp8			×	×					×
bp9			×	×			×	×	×
bp10			×	×		×	×	×	×
bp11			×	×					×
bp12			×	×					×
bp13				×	×	×	×	×	×
bp14				×					×
bp15				×				×	×
bp16				×			×	×	×

FCA classifies the set of considered blueprints into the concept lattice in Fig. 5. This lattice reveals the blueprints that are more important than the others. These are the blueprints that appear in the lower part of the lattice, because they have more attributes than the others. This is the case for the blueprint *bp3*, appearing at the bottom, since it has the best values for the considered attributes (priority:high and status: completed).

In this lattice, several groups (profiles) of blueprints can be extracted. We consider for example, the four following profiles (appearing in Fig. 5): blueprints that are approved regardless of priority (includes all the blueprints except for *bp14*, *bp8*, and *bp12*); the ones that are completed regardless of priority (*bp13*, *bp11*, *bp7*, and *bp3*); the ones having medium priority at least and are approved at least (*bp5*, *bp4*, *bp6*, *bp1*, *bp2*, *bp7*, and *bp3*); and finally the ones having a high priority and that are started at least (*bp1*, *bp2*, and *bp3*).

We use these four blueprint profiles to construct a new formal context of stakeholders. We make use of RCA (expressing the relation between objects and the concepts of another lattice), as we can see in Table 4. In this formal context, we can determine if a stakeholder has a profile or not by fixing a minimal number of blueprints belonging to

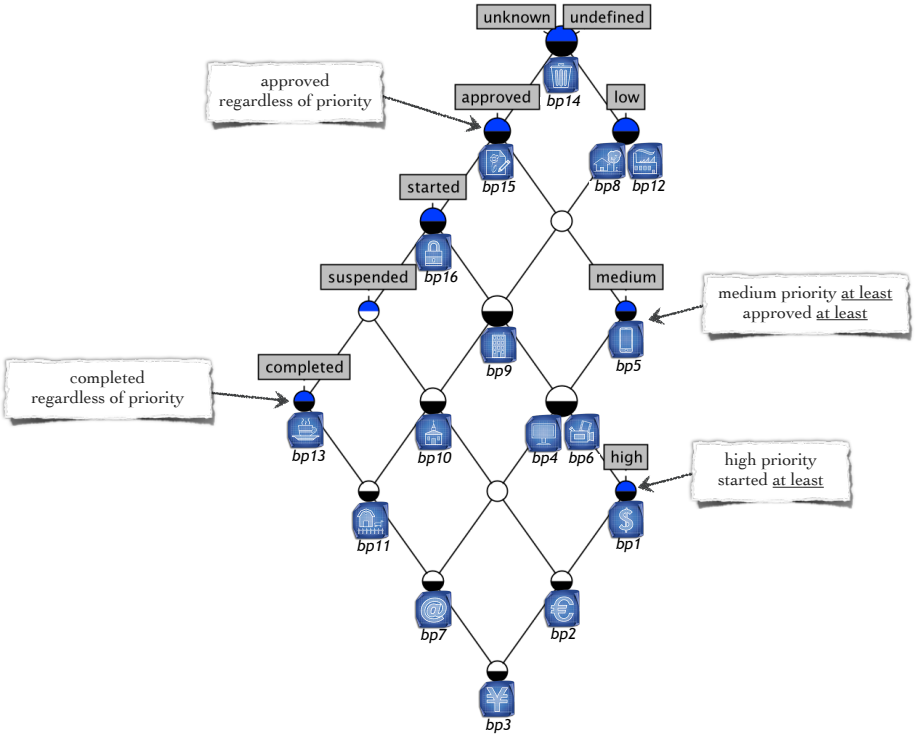

Fig. 5. Blueprints lattice revealing different profiles.

this profile, in which the stakeholder participated. In this example, we considered this number to be one blueprint at least. We also considered stakeholder participation to be a coarse-grained relation that includes proposing, approving, starting, and completing a blueprint. This aggregation of relations and getting the corresponding data is obtained through a direct SPARQL query on the RDF data.

Table 4. Stakeholders formal context.

	completed regardless of priority	**high priority started at least**	**medium priority at least approved at least**	**approved regardless of priority**
david		×	×	
john		×		
robert		×	×	
tony		×	×	×
oliver		×	×	×
mark		×		
joseph		×		
martin		×	×	
daniel	×			
kevin		×	×	×
mike		×	×	
william				
patrick		×	×	×
edward	×			

The stakeholder lattice that results from the context in Table 4, is shown in Fig. 6. In this lattice, we can notice the formation of four communities of stakeholders. Stakeholders inside each community share the fact that they participated in blueprints belonging to one of the four chosen profiles. We can notice that the stakeholder called *William* does not belong to any community. This is normal since we did not consider the blueprint profile (low priority and unknown status), in which he participates. We can notice also that these communities are overlapping. For example, the members of community $C4$ participate in blueprints of all profiles. While for example, the members of community $C1$ participate only in blueprints of a high priority and that are started at least.

Like this, stakeholder profiles can now be enriched with an additional information concerning their participation, obtained in a collective relative manner.

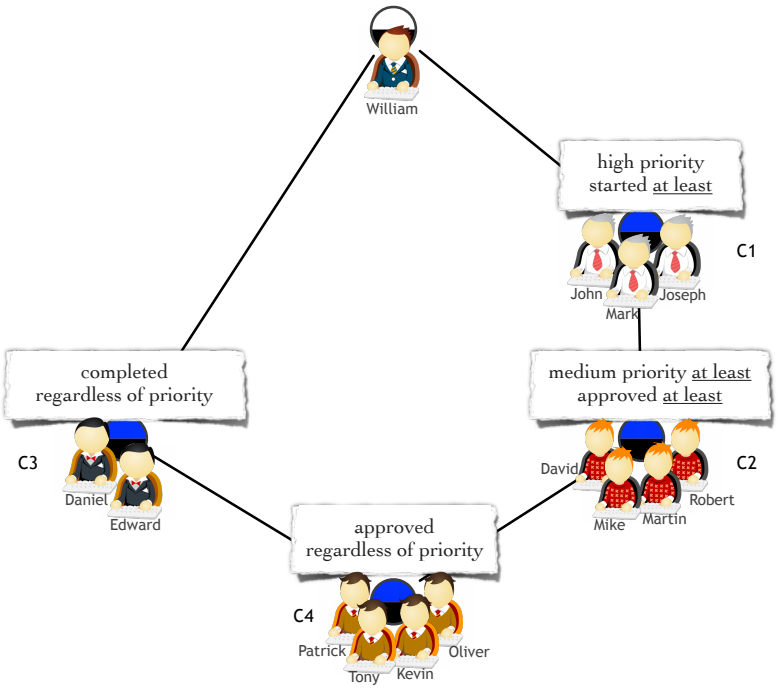

Fig. 6. Lattice of stakeholder communities.

Possible Configurations: FCA and RCA tend to generate fairly large lattices when dealing with datasets of large sizes. Possible solution to such an issue is to use the notion of a Galois Sub-Hierarchy [12], which is a compressed representation of the lattice. It encodes in a non-redundant way all the information that is necessary for the recovery of the complete lattice [14]. Another possibility is to impose constraints on the number of requirements inside each profile, and the number of requirements a stakeholder must participate in to have a certain requirement profile.

4 Proof of Concept

In this section, we present one of our conducted experiments, the Inkscape project[2], and we show the obtained results.

The Inkscape project contains 3840 contributing stakeholders, and 227 blueprints. We choose to show how the approach processes the blueprints only, because of the limited paper space. The analyst configurations[3] that we choose for classifying the blueprints are the following:

- **definitionState**, takes the values: **pendingApproval** = {new, review, drafting, discussion}, **approved**, **discarded** = {obsolete, superseded};
- **priority**, takes the values: **undefined**, **not**, **low** $<$ **medium** $<$ **high** $<$ **essential**;
- **relatedToBlueprint**, takes the value true if it is related to another blueprint;
- **relatedToBug**, takes the value true if it is related to a bug.

We also specify the relations that we want to take into consideration. We consider that stakeholders who **proposed** and **subscribed** to blueprints, have **participatedInBlueprint**. Running the approach on this data results in two lattices[4]: a lattice of blueprints, and a lattice that classifies stakeholders by the extracted blueprint profiles.

The blueprints lattice, shown in Fig. 7, gives us an overview of the blueprints, according to the considered attributes. It shows three main profiles: discarded, pendingApproval, and approved blueprints. The approved blueprints profile contains itself three other main sub profiles. We list these profiles in Table 5 together with the number of blueprints inside each one of them. Thereafter, we build the stakeholders lattice us-

Table 5. Extracted blueprint profiles.

Blueprint profile	Description	# blueprints
$blueprint\text{-}p1$	discarded	41
$blueprint\text{-}p2$	pendingApproval	161
$blueprint\text{-}p3$	approved	25
$blueprint\text{-}p3.1$	approved-low	4
$blueprint\text{-}p3.2$	approved-medium	12
$blueprint\text{-}p3.2.1$	approved-medium-relatedToBug	10
$blueprint\text{-}p3.3$	approved-high	6
$blueprint\text{-}p3.3.1$	approved-high-relatedToBug	4
$blueprint\text{-}p3.4$	approved-essential	2

ing the three main blueprint profiles. This lattice is shown in Fig. 8, it highlights six communities of stakeholders. The communities 1, 3, and 6 correspond to stakeholders participating in approved, pendingApproval, and discarded blueprints, respectively. $Community1$ for example, contains itself two other sub communities, with a total

[2] Data retrieved on September 18, 2012, from `https://launchpad.net/inkscape`.

[3] The attributes and values are provided by Launchpad. We specify the values that we consider to be equivalent and also specify the ones that should be treated according to some order.

[4] Here, we show a compact version of the lattices. The complete lattices can be visualized on: `www-sop.inria.fr/members/Zeina.Azmeh/REFSQ13/`

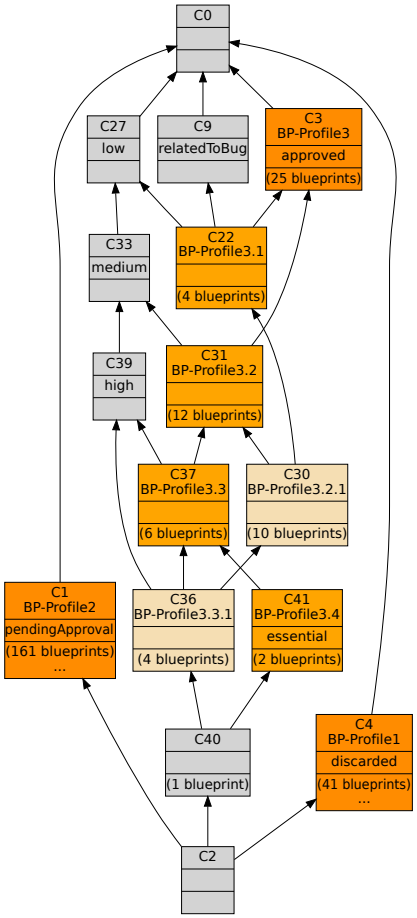

Fig. 7. The blueprints lattices.

number of 80 stakeholders (63 of them participated in accepted blueprints only). Usually, stakeholders appearing closer to the bottom of a lattice tend to have more profiles. For example, the three stakeholders of $Community4$ have participated in blueprints belonging to the three blueprint profiles.

Lattices utilization: The blueprints lattice gives us a better view on the blueprints according to their various attributes. It enables us to identify the different profiles, in addition to exploring a classification of these profiles. This is quite useful because if we consider for example the case of accepted blueprints, the next activity that may be applied to them might be selection for processing. Having such a blueprint classification (embodied in the lattice) enables us to identify the blueprints that have the best values for the chosen attributes. These blueprints are the ones appearing closer to the bottom of the lattice (because they cover more attribute values than the others).

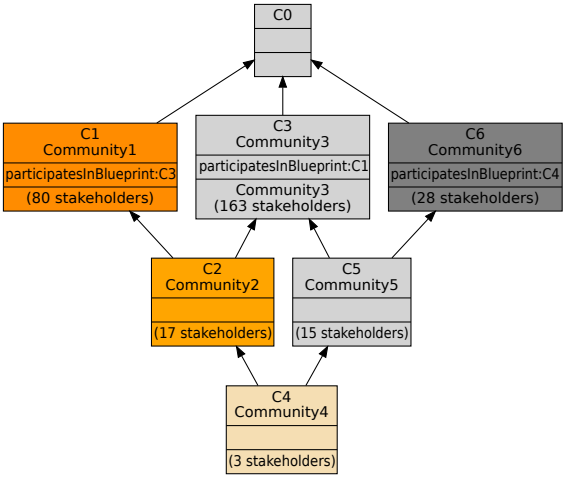

Fig. 8. Lattice of stakeholders of the Inkscape project.

Using the stakeholders lattice, we can discover stakeholders having specific profiles. In this experiment, we considered the main blueprint profiles only, but if we consider all the profiles in Table 5, we can then discover for example, the stakeholders who participated in approved blueprints with high to essential priority values (the profile $blueprint\text{-}p3.3$). Blueprints that may be proposed in the future by these stakeholders may have a higher probability of being accepted than the ones potentially proposed by stakeholders of $Community6$ (who participated in discarded blueprints only). We can also decide to prioritize stakeholders having the best blueprint profiles, and consequently prioritize their blueprints.

Threats to validity: According to the experiments that we conducted, we noticed some limitations in our proposed approach. These limitations appear in three situations: when the contributing stakeholders are newbies with no previous participation in any blueprint or bug, when there is no sufficient number of evaluated blueprints or bugs to extract stakeholder communities, and when the retrieved dataset is fairly huge (the case for instance of the Ubuntu project having more than 290,000 bugs).
Since our approach has a learning aspect (evaluating stakeholders according to their evaluated artifacts), in the first two situations, the approach will fail to produce useful lattices. In such a case, stakeholders might get grouped into only one community; the community of stakeholders participating in requirements that are pendingApproval.
Having large datasets will also cause some inefficiency to the approach, regarding the ability to analyze the resulting lattices due to the added complexity.

Discussion: In this experiment, we considered the stakeholders' participation in blueprints only. Considering their participation in other artifacts (bugs or code branches) gives us more information about them that would help in better evaluating them. The advantage of using RCA is that we can choose to consider any other artifact to enrich the stake-

holders lattice without affecting the approach. Especially with the use of the ontology that we are proposing, since we can choose the different levels of granularity that we wish to consider. Nevertheless, this may add more complexity to the resulting lattices, regarding their readability and understandability.
What should be noticed is that the approach can be totally configured regarding the chosen levels of granularity, even when deciding the blueprint profiles to consider. For example, the stakeholders lattice was generated considering a coarse granularity. Considering finer granularity would leads us to generate the stakeholders lattice using all of the extracted blueprint profiles.

5 Related Work

In this section, we list related work of two main categories: works dealing with requirements engineering using social network analysis (SNA), and works dealing with community detection. Social Network Analysis is the application of methods to understand the relationships among actors and on the patterns and implications of the relationships. A social network is a structure consisting of actors and the relations defined on them. It is often depicted as a graph. A community in a social network is a group of people that are gathered according to their common properties or approximating interests.

Social Network Analysis (SNA): In [15], Lim and Finkelstein propose a tool for the elicitation of pertinent highly wanted requirements in a software system. It is a semi-automatic approach that makes use of social network analysis for requirement engineering. Fitsilis et al. present in [8] the use of SNA for the management and prioritization of software requirements. In [20], Pagano lists the challenges, embodied in the fairly huge amount of unstructured data that may suffer of a low quality and possible conflicts. He also lists the current techniques aiming at facing each of these challenges, like SNA and collaborative filtering. In [18], Mulla and Girase proposed an approach that uses social networks and collaborative filtering for requirements prioritization.

Community Detection: Community detection and graph partitioning share the goal of separating a network into groups of nodes having few connections between them [19]. The difference is that in community detection, the objective is to find the naturally occurring groups regardless of their number or size. Another difference is that in graph theory, ideal partitioning results in disjointed groups, while in community detection, groups may be overlapping.

In [24], Veerappa and Letier propose an approach for stakeholders clustering based on their approximating ratings on requirements. The discovered stakeholders groups can be used afterwards for requirements decision-making. In [7], Cuvelier and Aufaure explain the notion of a community in the light of graph theory. They present the principle graph definitions and the different graph related measures that can be employed for social network analysis [23]. They list and detail the existing methods for community detection, categorized according to the used techniques.

In [25], Wang et al. models the interactions of users and information in a bipartite graph. They propose to manipulate the resulting graph by one-mode projections

to capture the shared interests of users and the information similarity. In [9], Flake et al. propose an algorithm for detecting communities in graphs of Web pages connected by hyperlinks. In [4], Blondel et al. present a heuristic method for discovering communities based on modularity optimization. Modularity, which is detailed in [19], is a score for measuring the density of links inside and outside communities that helps in determining the belonging of a node to a certain community. Other works that we can find adopt divisive algorithms that work on splitting a network by deleting edges [22]. Additionally, others use agglomerative algorithms that work on adding nodes to groups until no individual node remains [11].

Discussion: To our knowledge, current techniques for SNA and community detection manipulate social graphs according to their topologies only. They do not consider the semantics conveyed by the network elements. Consequently, a lot of important information may get discarded (as we showed in Section 4). Moreover, in the presented works, communities are considered as disjointed groups of nodes that do not overlap. While using concept lattices, the communities overlap.

An advantage of using FCA and RCA lies in the fact that we can represent any social network with its complete set of data, without any loss of information. Then the derived concepts enable us to reveal groups of each type of nodes with inclusion relations between these groups.

6 Conclusion

In this paper, we presented an approach for discovering communities of stakeholders to support requirements handling and decision-making. The approach is based on semantic Web languages and concept lattices. It reveals stakeholder communities by analyzing their past participation in requirements. We considered as a study context a platform for collaborative software development, from which we retrieved datasets about projects and annotated them semantically. The use of concept lattices enabled us to analyze heterogeneous multi-relational social networks of stakeholders and artifacts.

There are diverse perspectives for this work. On top of these perspectives is to study the utilization of the approach for the purpose of prioritization as well as recommendation of requirements and/or stakeholders, in addition to introducing the notion of trust among stakeholder communities. Another point that we would like to work on is to connect data from several platforms to enrich user profiles. An important issue to be considered also, is the scalability of the approach, when considering fairly large datasets. This may imply the dynamic updating of the resulting concept lattices, using incremental lattice construction algorithms. We may also consider the use of Galois Sub-Hierarchies (GSH) [12] as compact alternatives for concept lattices.

References

1. DOAP vocabulary (description of a project), https://github.com/edumbill/doap/wiki
2. Launchpad: https://launchpad.net/

3. Begel, A., Herbsleb, J.D., Storey, M.A.: The future of collaborative software development. In: Proceedings of CSCW '12. pp. 17–18. ACM, New York, NY, USA (2012)
4. Blondel, V., Guillaume, J., Lambiotte, R., Mech, E.: Fast unfolding of communities in large networks. J. Stat. Mech p. P10008 (2008)
5. Brickley, D., Guha, R.V.: Rdf vocabulary description language 1.0: Rdf schema. Tech. rep. (2 2004), http://www.w3.org/TR/2004/REC-rdf-schema-20040210/
6. Brickley, D., Miller, L.: FOAF Vocabulary Specification 0.98. Namespace document (August 2010), http://xmlns.com/foaf/spec/
7. Cuvelier, E., Aufaure, M.A.: Graph mining and communities detection. In: Aufaure, M.A., Zimányi, E. (eds.) Business Intelligence, LNBIP, vol. 96, pp. 117–138. Springer (2012)
8. Fitsilis, P., Gerogiannis, V., Anthopoulos, L., Savvas, I.K.: Supporting the requirements prioritization process using social network analysis techniques. In: Proceedings of WETICE '10. pp. 110–115. IEEE CS, Washington, DC, USA (2010)
9. Flake, G.W., Lawrence, S., Giles, C.L., Coetzee, F.M.: Self-organization and identification of web communities. Computer 35(3), 66–71 (Mar 2002)
10. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin/Heidelberg (1999)
11. Girvan, M., Newman, M.: Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99(12), 7821–7826 (2002)
12. Godin, R., Mili, H.: Building and maintaining analysis-level class hierarchies using galois lattices. SIGPLAN Not. 28(10), 394–410 (Oct 1993)
13. Happel, H.J., Maalej, W., Seedorf, S.: Applications of ontologies in collaborative software development. In: Mistrík, I., Grundy, J., Hoek, A., Whitehead, J. (eds.) Collaborative Software Engineering, pp. 109–129. Springer Berlin Heidelberg (2010)
14. Huchard, M., Hacene, M.R., Roume, C., Valtchev, P.: Relational concept discovery in structured datasets. Ann. Math. Artif. Intell. 49(1-4), 39–76 (2007)
15. Lim, S.L., Finkelstein, A.: Stakerare: Using social networks and collaborative filtering for large-scale requirements elicitation. IEEE Trans. Softw. Eng. 38(3), 707–735 (May 2012)
16. Manola, F., Miller, E.: RDF primer. W3C Recommendation 10, 1–107 (2004), http://www.w3.org/TR/rdf-primer/
17. Mirbel, I.: OFLOSSC, an ontology for supporting open source development communities. In: Cordeiro, J., Filipe, J. (eds.) ICEIS (4). pp. 47–52 (2009)
18. Mulla, N., Girase, S.: A new approach to requirement elicitation based on stakeholder recommendation and collaborative filtering, 3 (3), 51-60. IJSEA (2012)
19. Newman, M.: Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103(23), 8577–8582 (2006)
20. Pagano, D.: Towards systematic analysis of continuous user input. In: Proceedings of the 4th international workshop SSE '11. pp. 6–10. ACM, New York, NY, USA (2011)
21. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation 4, 1–106 (2008), http://www.w3.org/TR/rdf-sparql-query/
22. Shen, Y., Pei, W., Wang, K., Li, T., Wang, S.: Recursive filtration method for detecting community structure in networks. Physica A: Statistical Mechanics and its Applications 387(26), 6663 – 6670 (2008)
23. Tang, L., Liu, H.: Graph mining applications to social network analysis. In: Aggarwal, C.C., Wang, H. (eds.) Managing and Mining Graph Data, Advances in Database Systems, vol. 40, pp. 487–513. Springer US (2010)
24. Veerappa, V., Letier, E.: Clustering stakeholders for requirements decision making. In: Berry, D., Franch, X. (eds.) Requirements Engineering: Foundation for Software Quality, Lecture Notes in Computer Science, vol. 6606, pp. 202–208. Springer Berlin Heidelberg (2011)
25. Wang, F., Xu, K., Wang, H.: Discovering shared interests in online social networks. In: ICDCS Workshops. pp. 163–168. IEEE Computer Society (2012)