SEARCHING FOR INFORMATION IN SOFTWARE DEVELOPMENT PROJECTS USING THE ALERT SYSTEM

Luka Stopar, Gregor Leban
Artificial Intelligence Laboratory
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel: +386 1 477-53-61
e-mail: luka.stopar@ijs.si, gregor.leban@ijs.si

ABSTRACT

Large open source projects use several communication channels to exchange information. Examples of these channels are bug tracking systems (BTS), source code management systems, mailing lists, forums and wikis. Since these information sources use different platforms there is no system which would allow users of these communication channels to find information appearing on any of these sources. Developing a system that will support such search functionality is the main goal of a European project called ALERT. In this paper we will introduce the project and describe the search features that can be used by the users of the ALERT system and highlight the advantages that they bring.

1 INTRODUCTION

Large open source projects commonly have their developers and users located all over the world. In order to exchange information such as questions, comments, requests and bugs reports, several information sources are employed. Each of these sources serves a particular purpose. Issue tracking systems such as Bugzilla[1], Mantis[2] and LaunchPad[3] are commonly used for reporting problems and feature requests. Mailing lists and forums allow users of the software to participate in open discussions about various topics related to the developed software. Wikis are often used as platforms for providing user support in the form of software documentation, user guides and tutorials. Source code management systems are used by software developers to commit modifications to source code and to describe the introduced changes.

Due to various information sources, a common problem affecting these communities is finding information. In order to find an answer to a question, the user has to open a separate web page for each of these sources and use different search interfaces to find information of interest. Because there is no integrated way of finding information, duplicates of the same questions can be found on different information sources. Display of search results is also quite limited. Results are only displayed as a list, frequently with a text snippet that matches the query. There is not additional summary of results that would allow discovery of useful and unexpected patterns.

Empowering the users by providing the missing functionalities of the current systems is the goal of the European project ALERT[4]. More specifically, ALERT system, which will be developed as the result of the project, aims to:

- Provide functionality to integrate information from issue tracking systems, source code management systems, forums, mailing lists and wikis.
- Extract information from structured (meta-data, source code commits) and unstructured sources (all text generated by users) and store it in a knowledge base
- Provide advanced search capabilities across all communication sources.
- Provide automatic methods for otherwise time consuming tasks such as finding bug duplicates and suggesting developers, who can fix an issue.
- Provide subscription mechanisms that allow users to specify their interests in order to be notified when something relevant is posted in some information source.

In this paper we will focus only on the ALERT's functionality related to search over all information sources. In the next section we will start by introduction the structure of the ALERT system. We will describe all main components of the system and the data flow. Then we will describe the search service and the visualization service – two main components responsible for enabling the user to find relevant information. Additionally we will also briefly describe more advanced search options such as finding potential bug duplicates and recommending developers who can fix an issue.

2 ALERT SYSTEM ARCHITECTURE

The ALERT system consists of several components, where each component is responsible for a particular task. The overview of the architecture is shown in Figure 1.

At the top of the figure we have a set of sensors that are responsible for detecting when new information is published in the information sources used by the community. Whenever new information is posted in these sources, the

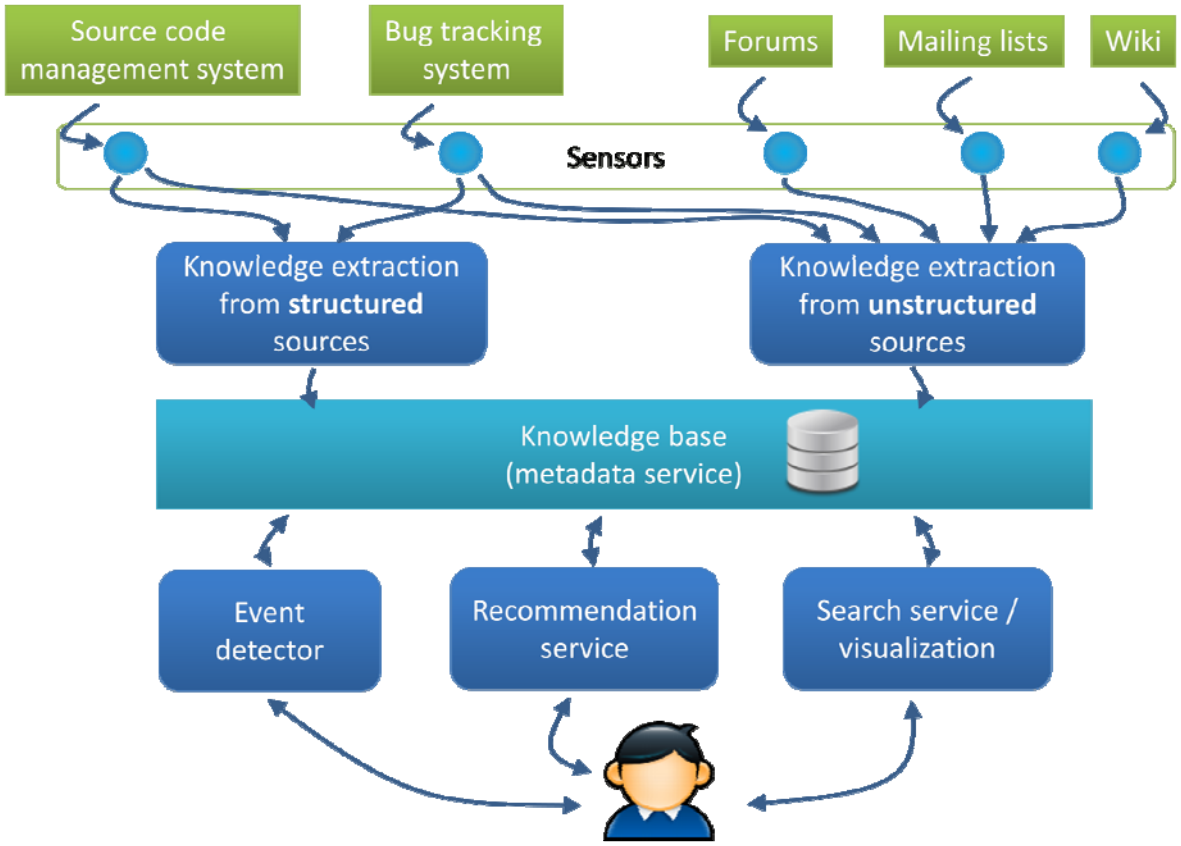

Figure 1. Architecture of the ALERT system.

sensors pass it forward to the components that are responsible for processing it.

Component "Knowledge extraction from structured sources" is able to process information on source code changes, such as what classes and methods were added or modified in a particular source code commit. The extracted information is then pushed to the Knowledge base where it can be accessed by other components.

The information generated in communication channels is mostly unstructured – it is plain text generated by the users. This type of data is processed by the component "Knowledge extraction from unstructured sources". One of the main tasks of the component is to annotate the text using the Annotation ontology [5]. The ontology contains relevant terminology for computer science and allows us to semantically enrich the text. The information extracted by the component is again stored in the Knowledge base.

After the data is stored in the Knowledge base it is ready to be consumed by other components. There are three main components that use the stored information to provide certain functionality for the user of the ALERT system. The Event detector enables the user to have control over what information he receives from the communication channels. The user can, for example, specify a set of topics he is interested in and as a result he will only be notified when posts on those topics are published in the communication channels.

The Recommendation service is responsible for recommending developers to fix an issue. Based on the activity of developers the service computes for each developer a set of scores that represents his/her expertise in different software development areas. When a new issue is created it can analyze what the issue is about (using the annotations from Annotation ontology) and identify the most appropriate developers to fix the issue.

The Search/Visualization service is responsible for providing to the user an integrated environment where he can search for information stored in the ALERT system. The details of the search service are described in the next section.

3 SEARCH SERVICE

The search interface provides the ability to search for information across all information sources. A screenshot of the interface is shown in Figure 2.

The top part of the interface contains a tab control offering four conceptually different search options. The first option is the general search. In the first text box it allows the user to enter plain search keywords. The second text box allows restricting the results based on the structured information available for the posts. When entering the text, the user is presented with a list of possibilities for the given text prefix. The possibilities are of the following types:

- People names. By specifying a person, the search results will be limited to posts where the person is the sender or recipient of the post.

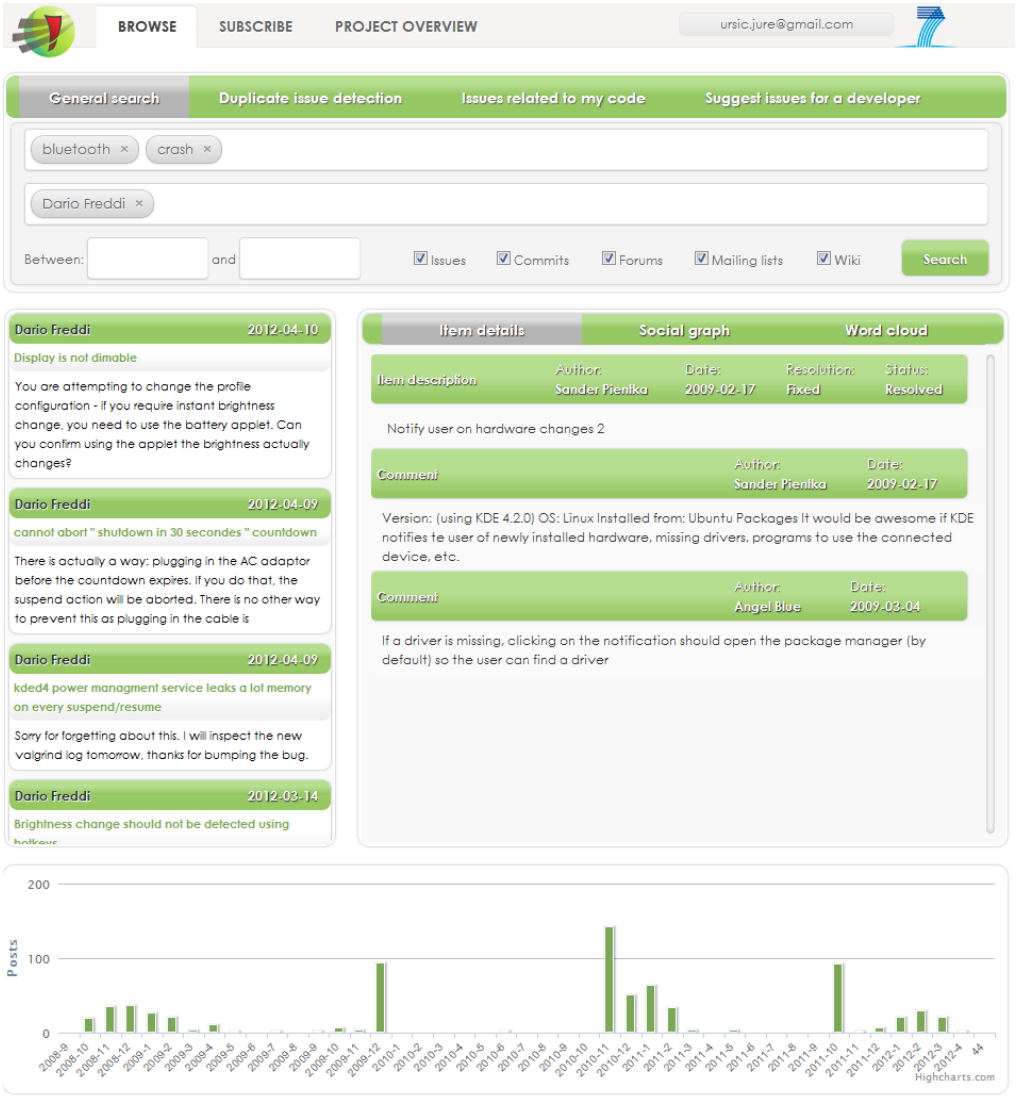

Figure 2. The search interface provided by the ALERT system

- Source code. By monitoring the source code management system we are aware of all the files, classes and methods developed in the project. By specifying the name of the file, class or method we can therefore limit the results to those posts that contain a reference to them.
- Project/component name. All issues on a bug tracking system are assigned to a particular product and component. By specifying a project or component we can limit results to a particular subset of issues.

The user can also specify a time constraint on the results by specifying a starting and ending date. Posts outside of the specified time window won't be considered in the query results. Additional constraint can also be the type of the posts. If the user would wish to ignore posts from a particular information source he can simply uncheck the appropriate checkbox.

Along with the general search, the user can also search for possible issue duplicates. On issue tracking systems the users often create duplicated reports of the same issue. Before starting to work on fixing the issue, a bug triager has to first determine if the issue is duplicate or not. Using our interface the bug triager can enter the id of the issue and the system will provide him with a list of most similar existing issues. For each existing issue the system even provides a score of the similarity. The similarity between issues is determined using an algorithm that takes into account the available meta information about the issue and the cosine similarity measure computed on the available text of the issue [6].

The third search option is to find issues related to my code. There are two crucial sources that ALERT has to monitor in order to support this functionality; the issue tracking system and the source code management system (SCMS). By monitoring SCMS we know for each developer which are the methods that he modified. By annotating the issues we also detect references to methods in the text. These references most often occur in stack traces that are provided

with the issue. By combining information from both sources we can suggest to the developer the issues that are possibly caused by his code.

The last search option is suggesting issues that a developer could solve. Although it seems similar to the previous one it is functionally different. In this case the user specifies the name of a developer. For each developer the ALERT system maintains an expertise profile that is created based on the topics that are mentioned in the posts created by the user. If the developer is, for example, in his posts frequently writing about Bluetooth, then he would be in a sense considered as an expert on this topic. The ALERT system can use the person's profile and match it to the existing open issues. Issues that are the best match are displayed to the developer and he can choose which ones he would like to fix.

4 VISUALIZATION SERVICE

After the user chooses the appropriate search option and specifies the search conditions, the search results are presented and summarized in different ways based on research done in [7].

The most ordinary display of the results is in a list. In Figure 2, this list is displayed in the middle left part. For each post in the list we show the author of the post, date, subject and a short content snippet. Clicking an item in the list shows its full content in the right part (Item details). If the clicked post is a part of a threaded message (such as an issue or email/forum discussion) the whole thread is displayed. The searched keywords are automatically highlighted in the text in order to more easily identify the information of interest. Depending on the post type, the item details tab also offers additional functionality. For issues, the user can also see a list of all posts that mention the issue and a list of most suitable developers to fix the issue. For source code commits, the shown information also contains a tree of files, classes and methods that were modified by the selected commit.

Together with the list of results, ALERT also provides three visualizations containing a summary of the search results. The first visualization is a timeline view and is displayed below the search results. It shows the distribution of search results over time. Using this visualization a user can spot times with high or low activity. Such patterns can be very helpful for gaining important insights. From our query in Figure 2 we can see that Dario Freddi was very active in December 2009 and then didn't participate again until December next year. When searching issues on a particular topic a spike on the timeline could identify when a bug was introduced in the code.

The second visualization is the social graph. By displaying a graph of people who are involved in the query results it shows a summary from the social perspective. There is a connection between two people in the graph if one is responding to the others post (for example, one person sends an email to the other). The size of the person's label depends on the number of times the person appears in the results. This visualization makes it easy to identify who are the most active/knowledgeable people on a particular topic. This information can be helpful if one would like find an expert on a particular topic in order to contact him directly.

The last visualization offered by the interface is the tag cloud. It offers a summary in a form of most relevant terms extracted from the results. The visualization can be helpful in different ways, depending on the entered search conditions. If a person is specified as a condition then the tag cloud can indicate what is the person's expertise or area of interest. In case the search is done using some keywords then the cloud can identify related topics that can be used to refine the query.

5 CONCLUSION

In this paper we presented an overview of the search functionality that is offered by the ALERT system. The system integrates information coming from several information sources – bug tracking systems, source code management systems, forums, mailing lists and wikis. We briefly described the architecture of the system and the way in which the information travels through individual components of the system. We described the search interface and the different search functionalities supported by the system. We also presented details of the visualization service which is responsible for displaying the results of the search queries. The service also provides different summaries of the results which enable the users to gain additional insights that would be otherwise hard to obtain.

6 ACKNOWLEDGMENTS

This work was supported by the Slovenian Research Agency, European Social Fund and ALERT (ICT-249119-STREP).

References

[1] "Bugzilla," 2011. Available: http://www.bugzilla.org/.
[2] "Mantis," 2011. Available: http://www.mantisbt.org/.
[3] "LaunchPad," 2011. Available: https://launchpad.net/.
[4] "ALERT," 2011. Available: http://www.alert-project.eu/.
[5] G. Leban, L. Dali, I. Novalija, "Enabling semantic search in open source communities," European Semantic Web Conference, Crete, 2012.
[6] G. Leban, "Analysis and prediction of bug duplicates in KDE bug tracking system," Information Society 2011, Slovenia, pp. 133-136..
[7] G. Leban and M. Grobelnik, "Displaying email-related contextual information using Contextify," International Semantic Web Conference, Shanghai, China, 2010, pp. 181-184.