CROSSMINER

Developer-Centric Knowledge Mining from Large Open-Source Software Repositories

Project Number 732223

D3.4 Natural Language Components

Version 1.0
27 December 2017
Final

Public Distribution

Edge Hill University

Project Partners: **Athens University of Economics & Business**, **Bitergia**, **Castalia Solutions**, **Centrum Wiskunde & Informatica**, **Eclipse Foundation Europe**, **Edge Hill University**, **FrontEndART**, **OW2**, **SOFTEAM**, **The Open Group**, **University of L'Aquila**, **University of York**, **Unparallel Innovation**

Project Partner Contact Information

Athens University of Economics & Business Diomidis Spinellis Patision 76 104-34 Athens Greece Tel: +30 210 820 3370 E-mail: dds@aueb.gr	**Bitergia** José Manrique Lopez de la Fuente Calle Navarra 5, 4D 28921 Alcorcón Madrid Spain Tel: +34 6 999 279 58 E-mail: jsmanrique@bitergia.com
Castalia Solutions Boris Baldassari 10 Rue de Penthièvre 75008 Paris France Tel: +33 6 48 03 82 89 E-mail: boris.baldassari@castalia.solutions	**Centrum Wiskunde & Informatica** Jurgen J. Vinju Science Park 123 1098 XG Amsterdam Netherlands Tel: +31 20 592 4102 E-mail: jurgen.vinju@cwi.nl
Eclipse Foundation Europe Philippe Krief Annastrasse 46 64673 Zwingenberg Germany Tel: +33 62 101 0681 E-mail: philippe.krief@eclipse.org	**Edge Hill University** Yannis Korkontzelos St Helens Road Ormskirk L39 4QP United Kingdom Tel: +44 1695 654393 E-mail: yannis.korkontzelos@edgehill.ac.uk
FrontEndART Rudolf Ferenc Zászló u. 3 I./5 H-6722 Szeged Hungary Tel: +36 62 319 372 E-mail: ferenc@frontendart.com	**OW2 Consortium** Cedric Thomas 114 Boulevard Haussmann 75008 Paris France Tel: +33 6 45 81 62 02 E-mail: cedric.thomas@ow2.org
SOFTEAM Alessandra Bagnato 21 Avenue Victor Hugo 75016 Paris France Tel: +33 1 30 12 16 60 E-mail: alessandra.bagnato@softeam.fr	**The Open Group** Scott Hansen Rond Point Schuman 6, 5th Floor 1040 Brussels Belgium Tel: +32 2 675 1136 E-mail: s.hansen@opengroup.org
University of L'Aquila Davide Di Ruscio Piazza Vincenzo Rivera 1 67100 L'Aquila Italy Tel: +39 0862 433735 E-mail: davide.diruscio@univaq.it	**University of York** Dimitris Kolovos Deramore Lane York YO10 5GH United Kingdom Tel: +44 1904 325167 E-mail: dimitris.kolovos@york.ac.uk
Unparallel Innovation Bruno Almeida Rua das Lendas Algarvias, Lote 123 8500-794 Portimão Portugal Tel: +351 282 485052 E-mail: bruno.almeida@unparallel.pt	

Document Control

Version	Status	Date
0.1	Document outline	18 September 2018
0.2	Document outline Restructure	20 September 2018
0.3	Background Section Completed	23 October 2018
0.4	Readers Section Completed	25 October 2018
0.5	Metric Providers and Tools sections completed	14 November 2018
0.6	Introduction, Risks and Limitations, Conclusion sections completed	20 November 2018
0.7	WP3 Progress section completed	26 November 2018
0.8	Internal first draft completed	6 December 2018
0.9	Internal Partner Review Version	10 December 2018
1.0	Final QA version of the deliverable	27 December 2018

Table of Contents

Executive Summary

Work package 3 (WP3) aims to contribute to CROSSMINER a collection of text mining components that allows users and developers of Open source Software to analyse a diverse range of natural language sources: bug tracking systems, communication channels, social media and question and answer websites. The components developed in this work package will contribute to the enrichment of knowledge contained within the GitHub enabling open source users and developers to gain an insight into the tools they use and improve the performance and quality of the code they produce.

WP3 is constructed of 4 tasks that produce 5 deliverables. In Task 3.1 we have analysed text-representation methods and developed new state-of-the-art machine learners for processing natural language. In Task 3.2 we explored methods for reading, searching and storing natural language sources associated with open source projects.

This deliverable is the product built upon the foundations of work presented in D3.1, D3.2 and D3.3. This document focuses on the work associated with Task 3.3 - the development of natural language components. It presents details regarding all of the natural language (NL) components, readers, metric providers and tools that are integrated (newly developed, updated or migrated) into CROSSMINER, discussing in detail their role, importance and inter-relationship with other NL components. The document also provides information concerning risks and limitations of the components developed. The report concludes by presenting the overall progress of WP3, work completed, requirements that have been fulfilled and outstanding work.

List of Abbreviations

Term	Definition
API	Application Programmable Interface
BTS	Bug Tracking System
CC	Communication Channel
PDE	Plug-in Development Environment
JSON	JavaScript Object Notation
NL	Natural Language
NLP	Natural Language Processing
NNTP	Network News Transfer Protocol
OSS	Open Source Software
REST	Representational State Transfer

1 Introduction

Content sources related to open source software, such as GitHub, NNTP news groups and StackOverflow, provide developers with a diverse range of useful information relating to specific open source projects. These sources are rich with natural language. However, key knowledge relating to a project is often overlooked as it is hidden away in vast amounts of text. The key to unlocking this hidden knowledge contained within text is natural language processing (NLP).

With respect to CROSSMINER, this deliverable follows on from work completed in D3.2 and D3.3. It focuses specifically on the development and integration of components related to the natural language processing capabilities of the CROSSMINER platform. Developed components, such as readers, historic and transient metrics and text processing tools are discussed in detail. Consequently, work completed as part of this deliverable lays the foundations for Task 3.4, which is currently in progress and aims at developing a recommender that is capable of suggesting code snippets and documentation relevant to the code being developed.

1.1 Overview

The remaining of this deliverable is organised into six sections. Section 2 provides an overview of CROSSMINER components, focusing on natural language. Sections 3, 4 and 5 discuss the readers, metric providers and tools developed and integrated into CROSSMINER, respectively. Section 6 identifies the risks and limitations of the work presented in this deliverable. Finally, Section 7 presents the conclusion of this deliverable as well as reports on the current progress of WP3 and future & outstanding work.

1.2 Intentions

The aim of Task 3.3 is to develop and integrate natural language related components into CROSSMINER. Fulfilment of the aim requires addressing the following use case and technological requirements presented in Tables 1 and 2, respectively. Each requirement has an ID, a description and priority[1].

1.3 Outcome

The outcomes of this deliverable are the components related to natural language that have been implemented and integrated into CROSSMINER. The developed natural language components, such as readers, text processing tools and metrics, are designed to be compatible with the CROSSMINER platform, the knowledge base and can also be used to run bespoke workflows for knowledge extraction.

[1] Shall - must be fulfilled, Should - should be fulfilled, May - may be fulfilled

Ref	Description	Priority
U31	Able to search mailing-lists	Shall
U32	Able to search forums	Shall
U33	Able to search issues	Shall
U37	Able to search IRC	Shall
U38	Able to search simultaneously across all the data sources	Shall
U39	Able to present the results by data source or globally for the project	Shall
U40	Able to order the results by relevance	May
U41	Able to search using regular expression patterns	Shall
U42	Able to detect in the data sources text referring to one or several bugs	Shall
U43	Able to detect in the data sources text referring to one or several commits	Shall
U44	Able to extract sentiment analysis from the communication data sources	Shall
U46	Able to detect similarities in messages so as to suggest answers	Should
U47	Able to propose recommendations to improve community management	Should
U48	Able to identify recurring problems identified by users	Should
U52	Able to list unattended bugs	Should
U53	Able to list unattended messages	Should
U54	Able to create a cluster map for messages using Lingo or similar clustering technology	Should
U55	Able to create a cluster map for bugs contents using Lingo or similar clustering technology	Should
U60	Able to analyse level of user activity in issue trackers	Shall
U61	Able to identify if a thread reached a conclusion and if it was positive	Should

Table 1: WP3 Use Case Requirements related to Task 3.3

Ref	Description	Priority
D32	The NLP analysis component should record the sources used by the developer and use them as a form of automatic feedback to improve its suggestions.	Should
D72	The other mining tools developed in WPs 2-4 shall expose a REST API that the cross-project relationship miners can consume	Shall
D73	The other mining tools developed in WPs 2-4 shall expose project level metrics that cross-project relationship miners can consume	Shall

Table 2: WP3 Technology Requirements related to Task 3.3

2 CROSSMINER Component Overview

The CROSSMINER platform, as shown in Figure 1, consists of numerous components that have been or will be developed by CROSSMINER research and use case partners. When combined, the components provide software developers with the capabilities to monitor, perform in-depth analysis and evidence-based selection of open source software, and facilitate knowledge extraction from large open source software repositories. The platform is capable of extracting knowledge associated with:

- Source Code
- System Configurations
- Cross-Project Relationships
- Communication in Natural Language

Combined, these sources provide CROSSMINER with a diverse range of knowledge that is related to various aspects of the software development life-cycle, allowing developers to make informed decisions based upon the results they are presented. The remaining of this document focuses specifically on components related to natural language, which are marked with an orange border in Figure 1.

2.1 Natural Language Components

Natural language contains vital and potentially hidden information that can be exploited to assist developers in making vital decisions surrounding open source software development. Consider the following scenario. A developer is developing a new parser for XML and would like to compare two libraries, SAX and JAXB. The developer is relatively inexperienced with this type of work and would benefit greatly by using a library that has an active and helpful community. Both of these libraries are hosted on GitHub as open source projects and are associated numerous entries on StackOverflow. To provide an insight into these two projects and present the developer with relevant heuristics requires a series of interconnected components designed specifically to perform tasks associated with reading textual information, processing it, computing suitable metrics and enriching the original text with extra knowledge.

The objectives of natural language components developed for CROSSMINER are three-fold. Firstly, they should be compatible with the CROSSMINER platform, so that they can be used to analyse the history of given Open Source software projects to compute metrics that summarise the quality of support offered to users over time. Secondly, to contribute to the CROSSMINER knowledge base, by enriching documents with extra information that facilitates efficient faceted search. Thirdly, to be used as plug-ad-play parts of bespoke knowledge extraction workflows, in the context of Work Package 5. CROSSMINER NLP components are of three categories: readers, metric providers, and tools. A detailed description of each component type and its role in CROSSMINER is presented in the remaining of this section.

2.1.1 Readers

A reader is the first component in a typical natural language processing workflow. The aim of a natural language reader in CROSSMINER is to retrieve textual information related to an open source software project, along with metadata associated with it. CROSSMINER supports four types of sources of textual information: Bug Tracking Systems, Communication Channels, Social Media and Question and Answer websites.

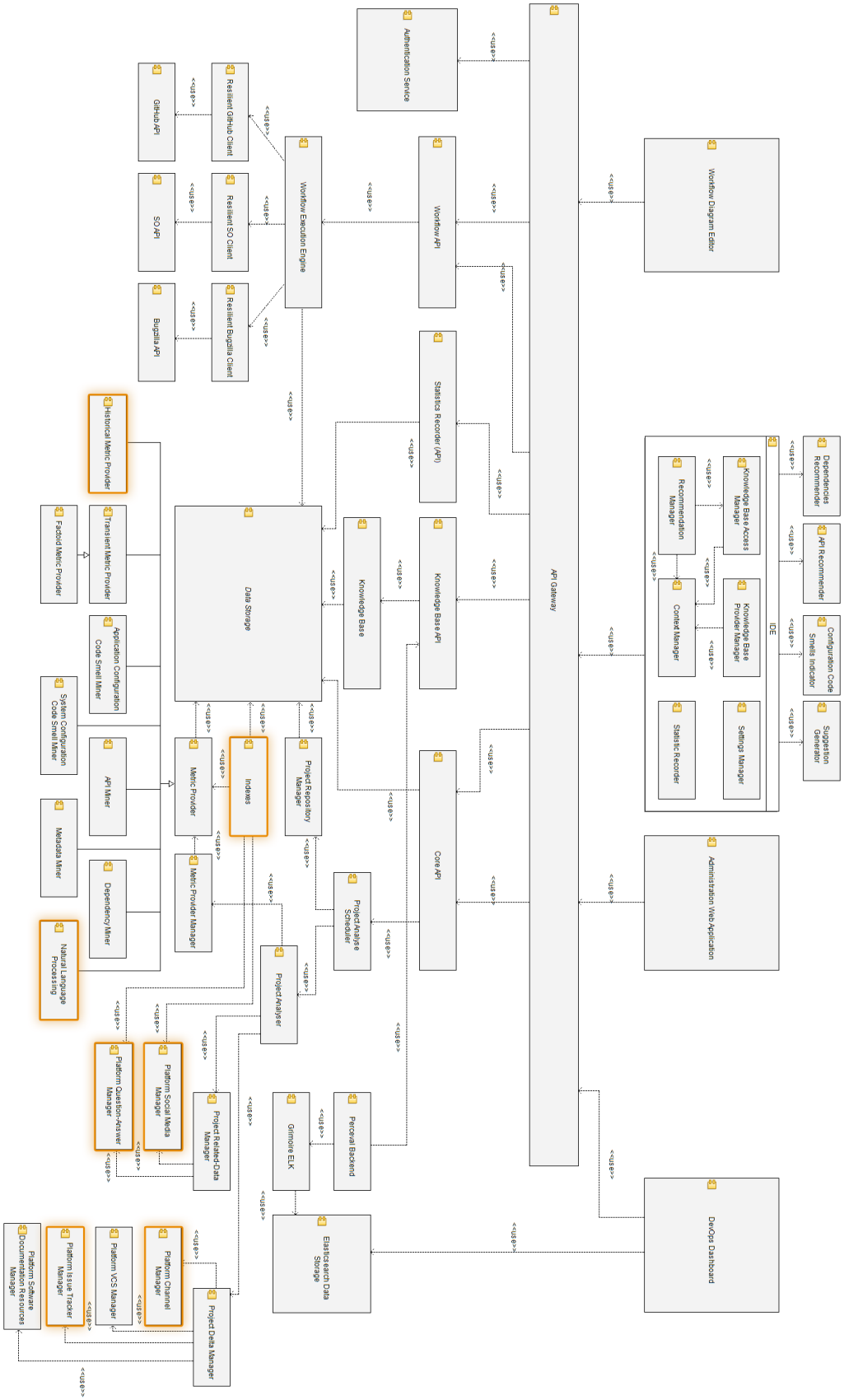

Figure 1: CROSSMINER Component Diagram

Readers have been developed for a variety of different sources, meaning that data retrieved is heterogeneous. A vital function within each reader is to restructure the data in such a way that it can be consumed by various other components within the natural language workflow in CROSSMINER. Depending on the source type, information retrieved by the readers can be passed either to metric providers or to natural language tools. Both paths result in enrichment of the knowledge base. A detailed description of the readers in this deliverable is found in Section 3.

2.1.2 Metric providers

Natural Language metric providers compute a diverse range of heuristics based upon the origin of the textual information[2]. In CROSSMINER, metrics are computed for every day that a project is being evaluated. For every day of the project, readers produce a delta, i.e. the set of changes that occurred in a particular communication means on that day. Changes may among others refer to new messages, updated bug reports, comments and thread status changes. During computation, a metric may rely upon information contained within the delta, results from other metrics and historical information from previous dates or days.

There are two types of metric providers in CROSSMINER. For a given project, transient metrics, as the name suggests, compute metrics for a particular day, overwriting the outcome that was computed previously, for the previous day of the project. Historic metrics are persistent, keeping a record of the changes for each day the project is evaluated. From a functional perspective, metric providers in CROSSMINER are organised into two tier stack as shown in Figure 2. In general, historic metrics enrich the knowledge base with knowledge relating to a project's history, and have dependencies on transient metrics in order to do this. This dependency relationship between the two types of metric providers is vital during the execution of the platform. A transient metric is only triggered during execution of the analysis components in CROSSMINER, if it is associated as a dependency with a historic metric. Without this dependency transient metrics will not be triggered during runtime. As a result of this architectural requirement, there are cases where historical metrics have been integrated into CROSSMINER with the sole purpose to trigger specific transient metrics. A detailed description of the metric providers associated with the processing of natural language and enrichment of the knowledge base in CROSSMINER is available in Section 4.

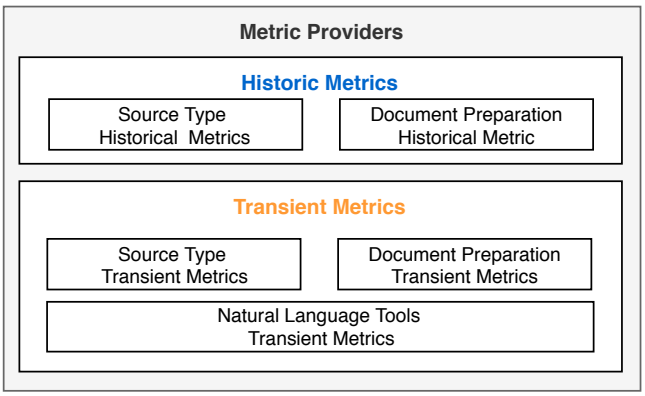

Figure 2: Metric Provider Stack

[2]There are some exclusions to this definition, however this is the general rule.

2.1.3 Tools

Natural language tools provide various capabilities that relate to either processing or indexing of knowledge in the CROSSMINER Knowledge Base. Once again depending on the origin of source data, tools can be triggered either by metric providers or by readers directly. An in-depth presentation of the tools developed and integrated in CROSSMINER is given in Section 5.

2.1.4 Knowledge Base

The knowledge base represents the final stage of the natural language processing workflow in CROSSMINER. The Knowledge Base consists of components that store knowledge that has been previously computed by metric providers and specific tools. For simplicity, the knowledge base consists of an instance of a MongoDB database and an ElasticSearch cluster. Knowledge contained within the knowledge base is presented to the users of the CROSSMINER platform, assisting them in the development of open source software. Although, the knowledge base and its development is not part of Work Package 3, the indexing capabilities for CROSSMINER are related to this work, as stated among the work package requirements, in Section 1.2.

2.2 Summary

This section provides an overview of the CROSSMINER platform. Since Work Package 3 is responsible for natural language processing in CROSSMINER, the latter part of this section discussed the role of various types of natural language components developed, the relationship of these components to the rest of the CROSSMINER platform and the importance of natural language processing for CROSSMINER. The following sections provide further details about each natural language component.

3 Readers and Reader Related Components

As discussed in D3.2, OSSMETER utilised a delta-based architecture for its readers and retrieved information from two distinct types of sources: *bug tracking systems* and *communication channels*. As part of Task 3.3, we introduced an additional, *non-delta based* reader architecture. This enables the CROSSMINER platform to support a further two types of sources, i.e. *social media* and *questions & answer websites*, for mining natural language, as shown in Figure 3. Furthermore, eight new readers have been developed, three were updated and a further six were migrated from OSSMETER. Our efforts now allow CROSSMINER to ingest natural language information from a total of 17 sources[3].

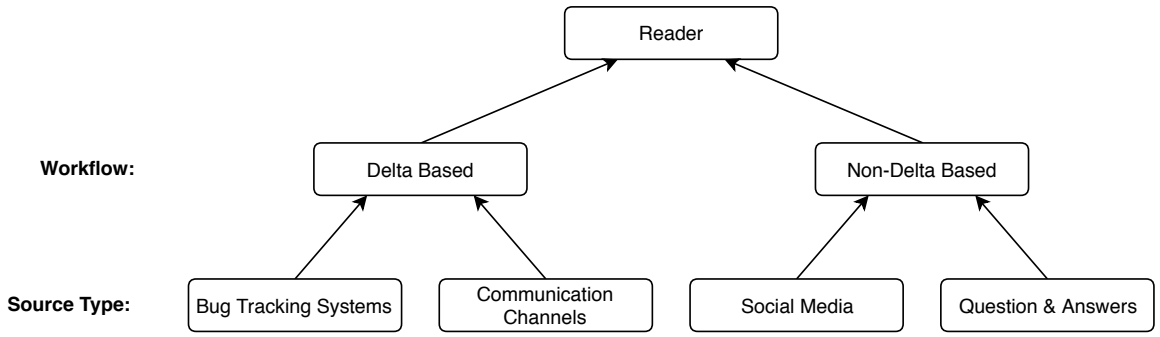

Figure 3: Readers in CROSSMINER

3.1 Delta Based Readers

All delta-based readers in CROSSMINER follow the same workflow illustrated in Figure 4. Information about an open source software project is retrieved from bug tracking systems or communication channels. The distinct functionality of delta-based readers is the ability to compute metrics based upon deltas. A delta is the sum of all activity that occurred on a particular day on a communication means, forming a time-line of historic activity as shown in Figure 5. Deltas are used to compute various metrics that enrich the knowledge contained in the knowledge base.

Regardless of the source type, each reader produces deltas in the same way. It begins by calculating the first date of activity for a particular project on a particular communication means. The calculation varies slightly depending on the source type. For example, bug tracking systems use the date of the first issue, whereas communication channels may use the first post in a forum or message posted in a newsgroup. The first date provides the CROSSMINER platform with the point in time, which can be incremented sequentially to produce deltas until the current day of execution.

Deltas contain lists of objects that represent events. For example, a delta can contain a new issue or update to an existing issue or comment on a bug tracking system. The information that is contained within the delta is then utilised by numerous metric providers to compute various heuristics associated to the source type, which in turn enriches the knowledge base. The remaining of this subsection provides details about particular delta-based readers in CROSSMINER and identifies those that are new, those that were updated and those that were migrated from OSSMETER.

[3]details of each of the 17 sources supported by CROSSMINER can be found throughout the remainder of this section

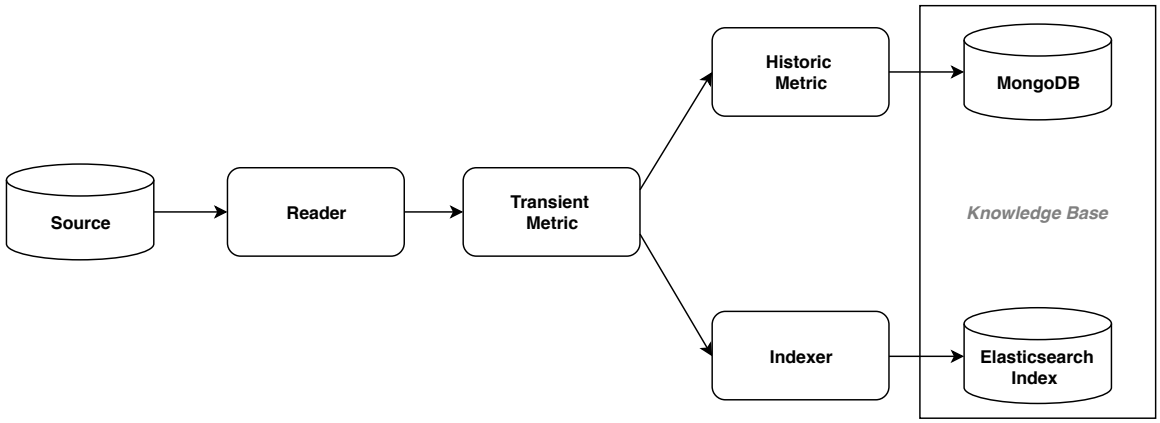

Figure 4: Delta Based Reader Workflow

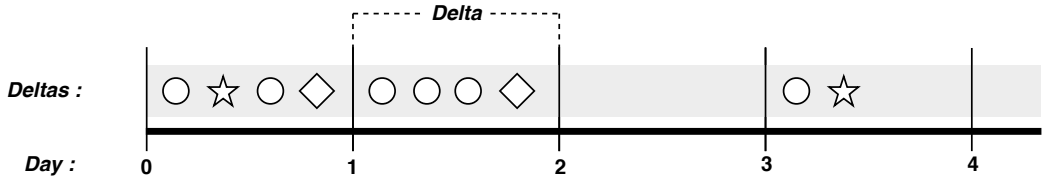

Figure 5: Delta

3.1.1 Bug Tracking Systems

Bug tracking systems (BTS), also known as issue trackers, assist developers in managing "bugs" throughout the lifetime of a software project. They are also used as channels that allow users of a software product to express ideas for feature enhancements or ask for technical support. This type of information is processed and utilised by metrics to measure characteristics such as how active development is, the number of bugs and the response time. The metrics enrich the knowledge base to provide suggestions such as solutions to issues faced by OSS developers.

In CROSSMINER, a BTS reader mines data related to issues and comments posted by users, as a minimum. Some BTS readers may also include additional elements unique to that particular system. These will be discussed in further detail where relevant.

As part of Task 3.3, we introduced support for two new bug tracking systems, updated three existing ones and migrated three from OSSMETER. The CROSSMINER platform now supports nine bug tracking systems. A summary of each bug tracking system supported by CROSSMINER is presented in Table 3.

Source	Status	Package name
Bitbucket	Updated	org.eclipse.scava.platform.bugtrackingsystem.bitbucket
Bugzilla	Updated	org.eclipse.scava.platform.bugtrackingsystem.bugzilla
GitHub	Updated	org.eclipse.scava.platform.bugtrackingsystem.github
GitLab	New	org.eclipse.scava.platform.bugtrackingsystem.gitlab
JIRA	Migrated	org.eclipse.scava.platform.bugtrackingsystem.jira
Mantis BT	New	org.eclipse.scava.platform.bugtrackingsystem.mantis
Redmine	Migrated	org.eclipse.scava.platform.bugtrackingsystem.redmine
Sourceforge	Migrated	org.eclipse.scava.platform.bugtrackingsystem.sourceforge

Table 3: Bug Tracking System Readers

3.1.1.1 New Bug Tracking System Readers

As mentioned earlier, CROSSMINER introduced support for two popular bug tracking systems: *GitLab* and *Mantis Bug Tracker*. With regards to the use case partners, both these new readers are to be utilised during the evaluation of several use cases.

GitLab is a web based application for all aspects of DevOps lifecycle and includes a bug monitoring (tracking) system. We have designed a delta-based reader in collaboration with YORK, who have developed *RESTMule*, an automatic client generator for sources with Open API specifications. Currently *RESTMule* is not ready to support *GitLab*, thus the final implementation of this reader will be completed shortly. The reader will mine information related to issues and comments. The GitLab reader will also support two forms of authentication: OAuth2 tokens[4] and Personal Access Tokens[5]. It is expected that the GitLab reader will require access to the project parameters shown in Table 4. It should be noted that they may change in future versions of the GitLab API.

Parameter	Type	Description	Example
OAuth2 Token	String	OAuth2 Token used for authentication	6Ji3tcrZfNQ3gna7CiE8JSLSiO6FiOIi
Personal Access Token	String	Personal Access Token used for authentication	6Ji3tcrZfNQ3gna7CiE8JSLSiO6FiOIi
Project	String	Name of the GitLab Project	Crossminer

Table 4: GitLab: Project Mining Parameters

Mantis Bug Tracker is a web-based, open source bug tracking system developed in PHP. It has been made available in several Linux distributions, Mac OSX and Windows. The reader integrated into CROSSMINER was developed using OKHTTP and Jackson[6]. As with many other bug tracking system readers, this reader can request and process information in issues and comments posted by users that relate to a given open source project. The reader accesses this information via authenticated requests made to the Mantis Bug Tracker REST API. Authentication requires generating an API token, which users can do via "Create API Token" located under "My account section" on the website of Mantis. The level of access a user has to information is subjective to the privileges they were assigned by the system administrator. Please note that in order for closed issues to also be included and processed by CROSSMINER a configuration setting located in

[4]GitLab OAuth2 Tokens - `docs.gitlab.com/ee/api/#oauth2-tokens`
[5]GitLab Personal Access Tokens - `docs.gitlab.com/ee/api/#personal-access-tokens`
[6]More information regarding these libraries can be found in D3.2

<path_to_install_directory>\config\config_inc.php had to be modified. The following line had to be added to the file or replace any previously assigned value: $g_hide_status_default = META_FILTER_NONE;

The project information shown in table 5 has to be provided to the CROSSMINER platform to mine information related to a specific project contained within a Mantis Bug Tracker instance.

Parameter	Type	Description	Example
project_id	String	the numerical identifier for the project	1
host	String	URL that points to the Mantis Bug Tracker instance	http://localhost:8888/mantis/
token	String	A generated string consisting of 32 characters	6Ji3tcrZfNQ3gna7CiE8JSLSiO6FiOIi

Table 5: Mantis Bug Tracker: Project Mining Parameters

3.1.1.2 Updated Bug Tracker Readers

As stated in D3.2, some OSSMETER readers required updating to address issues related to outdated or depreciated APIs. Three readers, *Bitbucket*, *Bugzilla* and *GitHub*, received significant modifications to their source code from their OSSMETER counterpart to maintain their mining functionality and compatibility with CROSSMINER.

Bitbucket: In deliverable 3.2, we stated that the Bitbucket reader would be migrated since our initial testing showed that it was operational. However, since then, Bitbucket has been going through a transitional phase with regards to its REST API. In OSSMETER, this reader utilised both version 1 and version 2 of the Bitbucket REST API. It appears that Bitbucket is discontinuing support for version 1 of API, meaning that eventually the end points of version 1 will be no longer available. Since the reader's functionality depended on certain version 1 endpoints that were no longer available, the reader would crash as data it was expecting was no longer being returned. To address these problems, we updated all the methods that made calls to version 1 endpoints to their version 2 equivalent and made modifications to the serialisation and deserialisation functionality to support the new information that was introduced.

For the reader to make requests to the Bitbucket REST API, a user must be authenticated. The Bitbucket reader in CROSSMINER supports basic authentication (username and password). The following project information shown in Table 6 has to be provided to CROSSMINER to mine information about a specific project contained in a Bitbucket repository.

Parameter	Type	Description	Example
User	String	The name of the user	Bob
Repository	String	The name of the repository	CROSSMINER
Login	String	The user name used to log into Bitbucket	Cross
Password	String	The password used to log into Bitbucket	M1ner

Table 6: Bitbucket: Project Parameters

Bugzilla: in D3.2, we mentioned that the reader developed for Bugzilla in OSSMETER used a depreciated version of the Bugzilla API, meaning that some functionality of the reader would no longer work and therefore required updating. Since Bugzilla 5.0, the Mozilla Foundation has switched to using a REST API. The Bugzilla REST API also adheres to the Open API specification[7]. This enables the use of *RESTMule* developed by our partners from the University of York, to provide the foundations for the updated Bugzilla reader in CROSSMINER. However, currently RESTMule is not ready to support Bugzilla, thus the final implementation of this reader will be completed shortly. In the meantime the original Bugzilla reader developed in OSSMETER has been made available in CROSSMINER.

The Bugzilla reader in CROSSMINER is capable of mining information related to issues, comments and attachments. This will remain the same in the RESTMule variant to maintain the consistency across both implementations. The reader currently supports two authentication methods: basic authentication (username or password) or token authentication.

Presented in Table 7 are the project parameters related to Bugzilla. Note that at least one authentication method has to be provided in order to make authenticated requests to the repository.

Parameter	Type	Description	Example
Username	String	The user name used to log into Bugzilla	Cross
Password	String	The password used to log into Bugzilla	M1ner
Product	String	Refers to a product contained within Bugzilla	Crossminer
Component	String	Refers to a specific component within a project	Bugzilla Reader

Table 7: Bugzilla: Project Parameters

GitHub: Similar to the Bugzilla reader, the API used in the OSSMETER version of the GitHub reader used eGit API version 4.1, which has undergone several improvements since its use in OSSMETER. Furthermore, GitHub has also revised its REST API (currently version 3) with several improvements and also made it adhere to the Open API specification. Hence why the resilient client generator was used to provide the foundations of the GitHub reader in CROSSMINER. The GitHub reader in CROSSMINER mines information related to issues, pull requests and comments posted by users; this was to maintain consistency with the data mined by the reader implemented in OSSMETER. The GitHub reader supports two types of authentication: basic authentication (username or password) or token authentication.

Although GitHub enforces a restriction on the number of calls that can be made per hour, the capabilities of the resilient client generator ensures that this is managed correctly and will not negatively impact CROSSMINER. The GitHub reader is associated with the following project parameters presented in Table 8.

[7]More information regarding Open API specification and the benefits it provides can be found in D3.2

Parameter	Type	Description	Example
User	String	The name of the user	Bob
Login	String	Username used to access GitHub (*basic authentication*)	Cross
Password	String	Password used to access GitHub *(basic authentication)*	Miner
Token	String	A unique token used to make authenticated requests *token authentication*	6ea3a349d921f6d33dc745df44d2487db3c
Repository	String	The name of the repository	crossminer
Owner	String	the username of the user that owns the repository	Bob2020

Table 8: GitHub: Project Parameters

3.1.1.3 Refactored Bug Tracker Readers

The readers listed below received minor refactoring and were tested to ensure compatibility with CROSSMINER during their migration from OSSMETER. The modifications did not concern their original functionality. Each of the readers mines information about issues and comments. The project parameters required by CROSSMINER for each source are presented in Table 9.

Source	Parameter	Type	Description	Example
Jira	Login	String	User-name used to access Jira	Cross
	Password	String	Password used to access Jira	M1ner
	Project	String	The name of the project	crossminer
Redmine	Login	String	User-name used to access Jira	Cross
	Password	String	Password used to access Jira	M1ner
	Name	String	Name of the user	Bob
	Project	String	The name of the project	crossminer
Sourceforge	URL	String	A url that represents the bugs of a project	http://sourceforge.net/rest/p/ soapui/bugs

Table 9: Migrated Bug Tracking Systems Readers: Project Parameters

3.1.2 Communication Channels

Within the context of CROSSMINER, a communication channel refers to an online community where various discussions related to OSS take place. Communication Channels are used to provide information for calculating metrics that measure various factors relating to a community of users, such as how active and reactive a community is, or estimate the general sentiment about specific discussions, or provide historic information concerning bugs and their respective solutions.

In OSSMETER, communication channels existed in two forms. Network News Transfer Protocol (NNTP) articles and SourceForge discussions. Due to the requirements from our use case partners, we extended the capabilities of CROSSMINER to also support forums as a new type of communication channel. From the platform's perspective, forums are handled differently due to their "threaded" nature and the information they

contain. This required additional functionality to be added to the Communication Channel Delta[8] and the addition of new metrics, which will be discussed in further details in section 4.

As shown in Table 10, CROSSMINER is capable of ingesting information from six Communication Channels; three readers are newly developed, two were refactored and one was migrated from OSSMETER.

Source	Status	Package name
Eclipse Forums	New	org.eclipse.scava.platform.communicationchannel.eclipseforums
IRC	New	org.eclipse.scava.platform.communicationchannel.irc
NNTP	Refactored	org.eclipse.scava.platform.communicationchannel.nntp
Sourceforge	Refactored	org.eclipse.scava.platform.communicationchannel.sourceforge
Sympa Emails	New	org.eclipse.scava.platform.communicationchannel.sympa
Zendesk	Migrated	org.eclipse.scava.platform.communicationchannel.zendesk

Table 10: Communication Channel Readers

3.1.2.1 New Communication Channel Readers

The following readers for communication channels were developed for CROSSMINER.

Eclipse Forums: Eclipse Forums are a centralised repository for people who develop and use Eclipse-based tools to discuss technical matters. They are organised in the following structure:

- ***Category*** - is a collection of related forums.
- ***Forum*** - is a collection of related topics
- ***Topic*** - is a collection of related posts.
- ***Post*** - is a message associated with a specific topic.

The reader integrated into CROSSMINER was developed using OKHTTP and JACKSON libraries. Given a specific forum, the Eclipse Forums reader is capable of extracting all of the posts associated with topics that are related to the forum. Accessing this information requires requests to be made to the Eclipse Foundation REST API. Eclipse enforces a rate limit of 1,000 requests per hour for both authenticated[9] and anonymous requests.

The reader, from an authentication perspective, supports both anonymous and authenticated requests. For authenticated requests, Eclipse requires a OAuth2 token. This can be generated following the instructions provided by the Eclipse Foundation[10]. The reader in its current form does not request information from protected resources, therefore its functionality is not impacted regardless of the availability of a token. However, authenticated users may request to increase the number of calls they are able to submit per hour, subject to Eclipse approval, allowing more data to be consumed by CROSSMINER per hour.

The project parameters associated with the Eclipse Forums reader are presented in Table 11.

[8] A component responsible for managing Communication Channel Deltas in CROSSMINER

[9] Note the rate limit can be extended, by contacting the appropriate eclipse representative. See `https://api.eclipse.org/docs` for more information.

[10] Eclipse Forums OAuth2 token generation - `https://api.eclipse.org/docs/auth`

Parameter	Type	Description	Example
Client Id	String	Client Id (optional)	G12DrqtW6qPuXC88Oorqc4rt4Xw5
Client Secret	String	Client secret (optional)	wpl9NBtLn2KxA4234mPxWoDbb5I
Forum Id	String	The unique id of the eclipse forum	305

Table 11: Eclipse Forums: Project Parameters

Internet Relay Chat (IRC) is an application for user communication in textual format. The chat is based on a server and client model. Each user uses an IRC client, either locally installed or web-based. All client applications are connected to a server. A user message is sent through the client application to the server and in turn to other user clients. IRC is mainly designed for group communication organised in channels, but also supports user to user private messaging. Communication messages can be logged by the intervening server. In the log, each message is timed and marked with the sender's username.

IRC used to be very popular in the 1990s and early 2000s. Nowadays, most users have moved to modern social media. However, IRC is still used for communication in software communities and organisations. Analysing IRC communication messages about OSS can inform about the quality of support offered to users. As analysing IRC logs is of interest to our use case partners, we developed a delta based reader of archived IRC logs. The reader assumes that an archive of IRC logs stored locally and does not use specialised libraries.

Parameter	Type	Description	Example
Log Archive	String	The Log archive to be processed	Crossminer_irc.log
Channel	String	The specific IRC channel which will be the focus	Readers

Table 12: IRC: Project Parameters

Sympa: is a mailing list management software. Similarly to most email servers, Sympa stores all emails sent by users in archives. The archive contains a folder for each year of activity, which contains a folder for each month, which in turn contains a folder for each day of that month. These day folders contain a text file for every email sent on that day. Emails are stored following the Multipurpose Internet Mail Extensions (MIME) format.

Parsing and analysing Sympa email archives is of interest to our use case partners. Thus, we developed a reader for Sympa archives, that can read all emails in an archive, or just the emails of a particular day. In succession, emails can be processed using any of the natural language processing components of the CROSSMINER platform to compute quality metrics or searched by the users via the Eclipse IDE plugin, as part of the knowledge base. The reader uses *javax-mail*, *apache commons mail* and *apache commons net*.

Parameter	Type	Description	Example
Log Archive	String	The Log archive to be processed	Crossminer_emails.log

Table 13: Sympa: Project Parameters

3.1.2.2 Refactored Communication Channel Readers

As mentioned earlier, three readers were migrated from OSSMETER. The first reader is capable of accessing newsgroups that utilise the Network News Transfer protocol (NNTP), the second processes information from Sourceforge discussions and the third one from ZenDesk. Each of these readers was subjected to testing after refactoring and was not modified to it's source code for use within CROSSMINER. Presented in the table below are the project parameters for each of the readers.

Source	Parameter	Type	Description	Example
NNTP	Authentication Required	String	Represents a flag (this is dependent on the NNTP server)	true
	Username	String	User-name used to access the news group	Cross
	Password	String	Password used to access the news group	M1ner
	Port	Int	Port number used to access the news group	1234
	Description	String	Description of the news-group	This is a news group for the CROSSMINER project
	Name	String	Name of the user	Bob
	Interval	Int	Polling interval in seconds [11]	5
	URL	String	A url which points to the NNTP server	nntp://host.domain/newsgroup
	News group name	String	Name of the news group	CROSSMINER
Sourceforge	URL	String	A Url that points to discussions hosted on Sourceforge	http://sourceforge.net/rest/p/gimponosx/discussion/
Zendesk	Authentication Required	String	Represents a flag (this is dependent on the Zendesk configuration)	true
	Username	String	User-name used to access Zendesk	Cross
	Password	String	Password used to access Zendesk	M1ner
	Port	Int	Port number used to access Zendesk	1234
	Description	String	Description of the project hosted in Zendesk	This is the CROSSMINER project
	Name	String	Name of the user	Bob
	Interval	Int	Polling interval in seconds [12]	5
	URL	String	A URL which points to the Zendesk server	http://zendesk.com/Crossminer
	News group name	String	Name of the news group in Zendesk	CROSSMINER

Table 14: Migrated Communication Channel Systems Readers: Project Parameters

[12]Polling interval is the period of time between the end of the time-out period of the last retry or completion of a network request, and the next request for data on the network.

3.2 Non-Delta Based Readers

Non-delta based readers aim to enrich the knowledge base with further contextual information regarding a specific open source project. Due to the large variety of data available on "Social Media" and "Question & Answer websites" it is difficult to process information in daily deltas . Non-delta based readers utilise queries to focus on information relevant to the requirements of the end user. The information is then processed by the workflow shown in Figure 6. Indexed information is searchable by the developers at the time of development through a specialised Eclipse plugin.

A potential issue that non-delta based readers face is that the size of data returned by the query can be large and may take a considerable amount of time to process. To alleviate this delay, CROSSMINER users can use computational resources available on other nodes of their network. For this purpose we utilised Crossflow, a tool that enables parallel and distributed execution of workflows, where nodes within the workflow can be tailored to meet specific requirements. Crossflow was developed by the University of York. We have worked closely with our partners to design a suitable workflow capable of enriching the ElasticSearch cluster contained within the knowledge base and to ensure that our natural language components are compatible with the requirements of Crossflow.

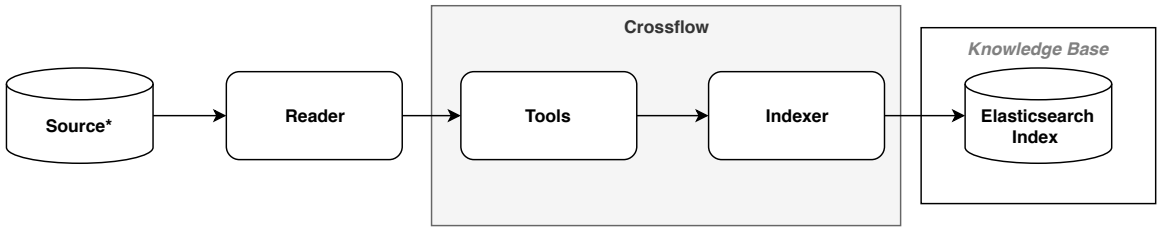

Figure 6: Non-Delta Based Workflow

3.2.1 Social Media

From a software perspective, social media can provide an insight into how a community perceives a project. This perception is an important factor that developers may wish to take into consideration when selecting which projects to use. As part of this deliverable we integrated support for Hacker News into CROSSMINER.

Hacker News[13] is a social media website mainly focusing on sharing news related to technology and related discussions. In Hacker News, users can also post questions, job offers, polls, among others; these can be discussed too through comments. As any social media website, it allows users to upvote and downvote publications. Access to posts and comments is free to all users, either subscribed or not.

Hacker News has two REST APIs: one developed by the owners of Hacker News and one provided by the search engine Algolia. The former[14] offers access to the data posted in Hacker News almost on real time by using Firebase[15], Google's mobile platform. This API allows to access all information available in Hacker News by publication ID, where publications are either posts or comments. Without knowing the ID of a publication, only a limited number of publications is accessible, such as the most recent posts or the most

[13] news.ycombinator.com
[14] github.com/HackerNews/API
[15] firebase.google.com

upvoted ones. The API does not require authentication and, to the best of our knowledge, it does not have any query quota.

The second API[16], developed by the search engine Algolia, is similar to the one designed by the owners of Hacker News, with extra functionality that allows to search publications. As mentioned previously, to access a publication in Hacker News using the first API, it is necessary to know its ID. This is not necessary on Algolia's API. As it indexes Hacker News on real time, natural language queries are possible. This API to Hacker News is back-ended on the original API, therefore it allows using all its functionality in addition to natural language queries. Algolia's API does not require authentication. However, there is a quota of $10,000$ queries per hour per IP.

We have developed a reader for Hacker News using OKHTTP and Jackson libraries. The REST API used is the one provided by Algolia, as it is more advanced that the original one and offers extra features. The reader has three modes. The first mode allows users to access the Hacker News API simply and easily, through a *Builder Pattern*. Users can submit queries, retrieve specific elements and filter them. The second mode allows to retrieve all elements on Hacker News in sequential order. The third mode allows to download the publications that match a specific query and their comments. The last two models are useful in populating CROSSMINER's knowledge database.

3.2.2 Question & Answer Websites

Unlike social media sources, Question & Answer (Q&A) websites consist of questions and answers derived from a community of users. Users of these websites seek and/or disseminate knowledge to a community to help others. For CROSSMINER this information can be exploited to help developers start new projects faster, by seeking answers to questions about the use of particular libraries. As part of Task 3.3, we integrated support for StackOverflow into CROSSMINER.

StackOverflow[17] is a crowd-sourced community, governed by rules similar to those of social media, where users can post questions regarding software development and also reply to the questions of other users. Users can comment, upvote and downvote questions and answers. StackOverflow covers a wide range of programming languages and libraries used by programmers.

StackOverflow offers a REST API to access the information posted on its website. As we expect CROSSMINER users to submit many queries to StackOverflow, we have decided to create a reader that will use dumps provided by StackOverflow's parent company, StackExchange[18]. In this way, CROSSMINER can after index StackOverflow locally and offer real time access to questions and answers without any restrictions, such as query quotas.

The reader has been developed using a *Builder Pattern*, meaning that CROSSMINER users can define which StackOverflow elements to index from the dump. For example, users can decide to index questions, answers, accepted answers, posts posted between certain dates, or posts regarding a specific tags[19], only. As StackOverflow dumps are very large XML files, we have decided to back-end the reader with Java's SAX parser. Specially designed for large files, SAX efficiently accesses a file line by line, instead of loading the entire file in memory.

[16] hn.algolia.com/api
[17] stackoverflow.com
[18] Dumps can be found in: archive.org/details/stackexchange. A new dump is released every three months, approximately.
[19] Tags are textual elements that denote a topic on StackOverflow, e.g. *Java*, *C++* and *parsing*.

It should be indicated, that the new reader can be used on other dumps provided by StackExchange, as it is using a standardised format. Using this tool, CROSSMINER users can read dumps from any StackExchange website, such as *SuperUser*[20] and *ServerFault*[21].

3.3 Summary

This section presented a comprehensive discussion about all readers that were developed and migrated as part of Task 3.3 to access natural language information for use within CROSSMINER. We have developed 8 new readers in total, updated 3 readers and migrated 6 from OSSMETER. Additionally, we extended the functionality of the Communication Channel delta to support forums. With the introduction of the non-delta based reader architecture, the CROSSMINER platform now has the ability to enrich the knowledge base with further contextual information relating to open source projects, in addition to the previously existing delta-based enrichment. We also introduced support for two new sources of information in the form of social media and question & answer websites. Overall CROSSMINER is capable of mining textual information from a total of 17 different sources.

[20] superuser.com
[21] serverfault.com

4 Metric Providers

As discussed in Section 2, metrics compute heuristics that enrich the knowledge base with useful information, that OSS developers can use to make informed decisions throughout the stages of software development. The purpose of this section is to present all metrics integrated into CROSSMINER as part of Task 3.3. The remaining of this section is organised into 2 subsections each discussing transient metrics and historic metrics, respectively. Each metric is marked with a status label from the following:

- **New:** A new development for CROSSMINER
- **Updated:** Metric received modifications to include preprocessing of text.
- **Upgraded:** Metric received upgrades as a consequence of developments since D3.3
- **Migrated:** Migrated from OSSMETER

4.1 Transient Metrics

As their name suggests, transient metrics are used to calculate heuristics that are associated with a particular period in time, i.e. a day in the case of CROSSMINER. They are stored temporarily within the knowledge base and their output is passed as parameters in the calculation of other transient and historic metrics. Transient metrics, with regards to this deliverable, are related to five areas: Bug Tracking Systems, Newsgroups, Forums, Natural Language Processing Tools and Document Preparation.

4.1.1 Natural Language Processing Transient Metrics

The transient metrics presented in Table 15 are associated with various natural language tools integrated into CROSSMINER. These metrics process the input text and return either a classification value or a conversion of text, which is in turn useful for the transient metrics that compute heuristics from Bug Tracking Systems, Newsgroups and Forums.

Metric	Status	Description
Code detector	New	This metric determines which parts of a bug comment or a newsgroup article are code and natural language
Plain Text Processing	New	This metric pre-processes each bug comment, article or forum post into a split plain text format
Request Reply Classification	Upgraded	This metric computes if each bug comment, newsgroup article or forum post is a request of a reply. It is based on the corresponding classification tool described in D3.3
Sentiment Classification	Upgraded	This metric computes the sentiment of each bug comment, newsgroup article or forum post. Currently it uses an upgraded version presented in D3.3
Emotion Classification	Upgraded	This metric indicates the emotions expressed in each bug comment, newsgroup article or forum post. We have replaced the tool that was used in OSSMETER with the one presented in D3.3
Severity Classification	Updated	This metric computes the severity level of each bug or thread.
Topic Classification	Updated	This metric computes a summary of topics dicussed in a collection of documents and the number of documents related to each topic

Table 15: CROSSMINER Transient Metrics related to Natural Language Processing

4.1.2 Bug Tracking Systems

Presented in Table 16 are the transient metrics available in CROSSMINER.

Metric	Status	Description
Active Users	Migrated	This metric keeps track of the users that submitted bug comments in the last 15 days
Bug Metadata	Migrated	This metric holds various metadata of a bug header, i.e. priority, status, operation system and resolution
Number of Comments	Migrated	The number of bug comments over time
Content Classes	Migrated	Identifies content classes in bug comments
Daily Requests & Replies	Migrated	This metric stores the number of comments, requests and replies for each day of the week.
Emotions	Migrated	Identifies emotions expressed in bug comments
Hourly Requests & Replies	Migrated	This metric stores the number of articles, requests and replies for each hour of the day.
Number of New Bugs	Migrated	Tracks the number of new bugs over time
Number Request & Replies	Migrated	Computes for each bug whether it is answered. If yes, it also computes the response time

Table 16: CROSSMINER Transient Metrics related to Bug Tracking Systems

4.1.3 Newsgroups

The transient metrics shown in Table 17 are the heuristics supported by CROSSMINER for Newsgroups.

Metric	Status	Description
Active users	Migrated	This metric keeps track of the users that submitted news in the last 15 days.
Articles	Migrated	Tracks the number of articles
Content Classes	Migrated	Identifies content classes in Newsgroup Articles
Daily Requests & Replies	Migrated	This metric stores the number of articles, requests and replies for each day of the week
Emotions	Migrated	Identifies the emotions expressed in Newsgroup articles
Hourly Requests & Replies	Migrated	This metric stores the number of articles, requests and replies for each hour of the day
Sentiment	Migrated	Computes sentiment at the beginning of each thread, at its end, and on average
Threads	Migrated	This metric holds information for assigning newsgroup articles to threads. The threading algorithm is executed from scratch every time.
Thread Request & Replies	Migrated	The metric identifies for each thread whether it is answered. If yes, it also computes the response time

Table 17: CROSSMINER Transient Metrics related to Newsgroups

4.1.4 Forums

Due to the unique nature of Communication Channel sources we decided to handle and compute transient metrics associated with forums differently than Newsgroups. In particular, forums posts are already threaded and there is no guarantee that they directly relate to a bug. The integration of these components has been delayed due to problems discovered with the data returned by Eclipse Forums not matching what was displayed on-line. Therefore, we have chosen not to integrate the metrics shown in Table 18 before testing them properly to ensure accuracy. The metrics are subject to change depending on the outcome of testing.

Metric	Status	Description
Active Users	New	This metric keeps track of the users that submitted news comments in the last 15 days
Number of posts	New	The number of posts over time
Content Classes	New	Identifies content classes in posts
Daily Requests & Replies	New	This metric stores the number of comments, requests and replies for each day of the week
Emotions	New	Identifies the emotions expressed in forum posts
Hourly Requests & Replies	New	This metric stores the number of forum posts, requests and replies for each hour of the day
Number of New Posts	New	Tracks the number of new forum posts over time
Number of New Forum Topics	New	Tracks the number of new forum topics
Number Request & Replies	New	Indicates for each forum post whether it is answered. If yes, it also computes the response time

Table 18: CROSSMINER Transient Metrics related to Forums

4.1.5 Document Preparation

The transient metrics presented in Table 15 are associated with the new Indexing capabilities of CROSSMINER. These metrics prepare documents for indexing from bug tracking systems and communication channels. Each document contains information from delta-based readers and other metrics and is represented in JSON format that is compatible with indexing.

Metric	Status	Description
Bug Tracker Indexer	New	This metric prepares documents from bug tracking systems for indexing
Communication Channel Indexer	New	This metric prepares documents from communication channels for indexing

Table 19: CROSSMINER Transient Metrics related to Document Preparation

4.2 Historic Metrics

As discussed in Section 2, historic metrics keep track of various heuristics associated with a specific open source project over its lifetime. In CROSSMINER, historic metrics are organised into four areas: Bug Tracking System, Newsgroup, Forum and Document Preparation.

4.2.1 Bug Tracking Systems

Table 20 presents the historic metrics related to bug tracking systems in CROSSMINER:

Metric	Status	Description
Number of Bugs	Migrated	It indicates how many bugs there are across all the bug tracking systems related to a project over time.
Number of Comments	Migrated	It indicates the evolution in the quantity of comments for all the bugs related to a project over time
Emotions	Migrated	Summarises the emotions expressed in the bug tracking systems of a given project
Number of New Bugs	Migrated	Indicates the number of new bugs created over time
Number of new users	Migrated	Indicates the number of new users over time
Open Time	Migrated	Measures the average duration an issue is open
Number of patches	Migrated	Computes the number of bug patches over time
Number of Requests & Replies	Migrated	Indicates the number of requests and replies out of all comments in the bug tracking systems for a given project
Response Time	Migrated	Measures the average time it takes for a request to be replied, for a given project.
Sentiment	Migrated	Reveals which sentiments are expressed in a bug tracking system for a project
Severity	Migrated	Estimates the severity levels of bugs found in the bug tracking system of a project
Status	Migrated	Indicates the status of a bug, e.g. open, closed
Topics	Migrated	Indicates the topics that were discussed throughout the project via clustering
Unanswered	Migrated	Indicates how many issues remain unanswered over time
Users	Migrated	Holds the users that have posted at the bug tracking systems of a project over time

Table 20: CROSSMINER Historic Metrics related to Bug Tracking Systems

4.2.2 Newsgroups

Presented in Table 21 are the Historic metrics associated with Newsgroups in CROSSMINER.

Metric	Status	Description
Number of Articles	Migrated	Computes the number of articles per day for each newsgroup separately
Emotions	Migrated	Computes the number of emotional dimensions in comments submitted every day
New Threads	Migrated	Computes the number of new threads per day for each newsgroup separately
New Users	Migrated	Computes the number of new users per day for each newsgroup separately
Requests & Replies	Migrated	Computes the number of request and reply newsgroup articles per day for each newsgroup separately
Average Requests & Replies	Migrated	Computes the average number of request and reply newsgroup articles per day
Response Time	Migrated	Computes the average time in which the community responds to open threads per day for each newsgroup separately. (Format: dd:HH:mm:ss:SS, where dd=days, HH:hours, mm=minutes, ss:seconds, SS=milliseconds)
Sentiment	Migrated	Computes the overall sentiment per repository up to the processing date. The overall sentiment score ranges from -1 (negative sentiment) to +1 (positive sentiment). In the computation, the sentiment score of each thread contributes equally, independently of its size.
Severity	Migrated	Computes the number of newsgroup severity levels in threads submitted every day
Severity Response Time	Migrated	Computes the average sentiment, the sentiment at the beginning of threads and the sentiment at the end of threads per severity level, in newsgroup threads submitted every day
Severity Sentiment Time	Migrated	Computes the average sentiment, the sentiment at the beginning of threads and the sentiment at the end of threads per severity level, in newsgroup threads submitted every day
Threads	Migrated	Computes the number of threads per day for each newsgroup separately
Topics	Migrated	Computes the labels of topics (thematic clusters) in articles submitted every day
Unanswered Threads	Migrated	Computes the number of unanswered threads per day for each newsgroup separately
Users	Migrated	Computes the number of active and inactive users per day for each newsgroup separately

Table 21: CROSSMINER Historic Metrics related to Newsgroups

4.2.3 Forums

The historic metrics shown in Table 22 were all newly developed for CROSSMINER to account for the new type of communication channel that was implemented. For reasons discussed earlier in Section 4.1.4, we have chosen not to integrate these at this time until we can ensure their accuracy.

Metric	Status	Description
Number of Posts	New	Indicates how many forum posts there are within a specific forum over time
Number of Topics	New	Indicates how many forum topics there are within a specific forum over time
Emotions	New	Summarises the emotions expressed in the bug tracking systems of a given project
Number of New Posts	New	Indicates the number of new posts created over time
Number of new users	New	Indicates the number of new users over time
Number of Requests & Replies	New	Indicates the number of requests and replies out of all forum posts within a specific forum
Response Time	New	For a given project, it measures on average the time it takes for a reply to a forum post.
Sentiment	New	Reveals which sentiments are expressed in posts related to a specific forum
Status	New	Indicates the status of a topic, e.g. open, closed
Unanswered	New	Indicates how many topics remain unanswered over time
Users	New	Holds the users that have posted at the bug tracking systems of a project over time

Table 22: CROSSMINER Historic Metrics related to Forums

4.2.4 Document Preparation

Unlike other historic metrics, the metric shown in Table 23 does not contribute any knowledge from delta based sources to the knowledge base[22]. This historic metric was included to enable the transient metrics responsible for document preparation to be triggered during execution, as discussed in Section 2.1.2.

Metric	Status	Description
Index	New	This metric is an architectural requirement associated with the preparation of documents for indexing. This does not 'compute' knowledge for the knowledge base.

Table 23: A CROSSMINER Historic Metric related to Indexing

4.3 Summary

In summary, this section has described all transient and historic metrics associated with natural language processing in CROSSMINER. We have introduced a total of 25 new metric providers, updated a further two, upgraded three and migrated a further 49 from OSSMETER into CROSSMINER.

[22]It should be noted that this historic metric should be hidden from users of the platform, so that they cannot select to exclude it

5 Tools

A number of tools related to natural language have been integrated into CROSSMINER. Most tools were designed and developed from scratch, adding new capabilities to the platform.

Components presented in this deliverable depend on various tool-kits[23] Traditionally these dependencies would be added using a dependency management mechanism such as Maven or Gradle. However, due to the limitations[24] of Eclipse headerless PDE (Plug-in Development Environment), libraries required by each tool-kit had to be added manually as a plug-in package that exposes the libraries at runtime. This ensures that the libraries are accessible by the tools that require them. Additionally, each tool kit was carefully scrutinised to ensure that its licence is compatible with CROSSMINER.

CROSSMINER natural language tools can be divided into three types depending on the functionality they provide to the platform, as shown in Figure 7. In this section, we describe the types of tools and the tools themselves, detailing their role in CROSSMINER, as well as their dependencies.

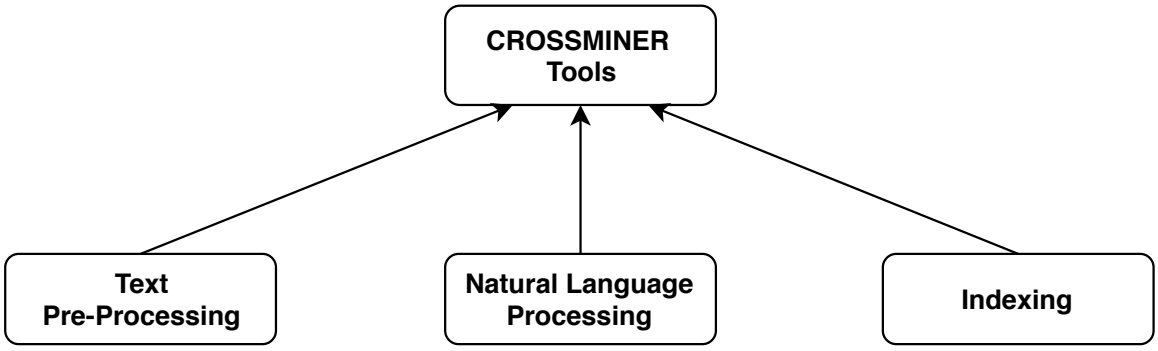

Figure 7: Types of Tools in CROSSMINER

5.1 Text Pre-Processing Tools

Due to the diverse nature of natural language sources used in CROSSMINER, data needs to be pre-processed before natural language processing. In other words, data is transformed into a standard format in a pre-processing stage. For example, GitHub and StackOverflow text comes formatted in markdown syntax. The former always returns text in the original markdown syntax, whereas StackOverflow returns text in HTML format.

To ensure that the input of natural language processing tools is formatted as required, we have implemented and integrated into CROSSMINER a series of pre-processing tools, accessible to CROSSMINER tools and users. The pre-processing tools are:

[23] In the context of this document, a tool-kit represents a collection of code libraries (jars).

[24] More information regarding this limitation can be found at `http://maven.apache.org/plugins/maven-eclipse-plugin/pde.html`

- An HTML Parser
- A Markdown Parser
- A Code Detector

5.1.1 HTML Parser

An HTML Parser extracts clean text from HTML formatted input. In other words, it removes tags, such as <*b*>, <*font*> and <*a*>, and returns only the text, that would be shown if the input was opened in a web browser.

The HTML parser uses an interface to JSOUP HTML parser and has been extended with extra methods, such as splitting text into paragraphs[25], parsing and removing certain HTML tags, or filtering of text with respect to specific labels, only.

5.1.2 Markdown Parser

Markdown provides syntax for formatting plain text in websites. Markdown is widely used in websites, such as GitHub and StackOverflow, as it can be easily parsed and converted into HTML. We integrated a markdown parser able to convert the enriched text, either into plain text or HTML formatted text. The tool is back-ended by the Atlassian Commonmark parser.

5.1.3 Code Detector

In software communities, messages usually contain code listings often referred to as snippets. From a human perspective, this is helpful for developers for reasons such as solving a problem or declaring a bug. Nonetheless, from a natural language processing perspective, code-related elements can be a source of noise.

Although some data sources, such as StackOverflow or GitHub, support special tags to mark code portions, others, such as newsgroups, do not provide such features. Moreover, marking code portions, in most cases, is left to the user decision, therefore the tag is sometimes used inaccurately.

For these reasons, we have developed a code detector and presented it in D3.1. It is identifies if a body of text contains code or natural language in English. The code detector is currently used by the sentiment analyser, the emotion classifier and the severity classifier.

5.2 Natural Language Processing Tools

The following tools are utilised for processing natural language in CROSSMINER. For each tool, we present a brief description of its role within CROSSMINER and its input and output. All tools have been designed and developed so that they can be used as standalone tools and also are compatible with both delta-based and non delta-based workflows. More details about the research aspects associated with these tools are presented in D3.3.

[25] Paragraphs are originally defined by the HTML tag <*p*>.

5.2.1 Core Natural Language Processing Tools

In preparation for complex natural language processing tools, such as a sentiment analyser or a severity classifier, it is necessary to apply a series of core natural language processing tools, i.e. tools that perform fundamental text processing tasks. The tools listed below have been developed and integrated in CROSSMINER.

- A text tokeniser
- A text lemmatiser
- A text normaliser
- A Part-of-Speech tagger

The text normaliser was developed in house from scratch, whereas all other tools are back-ended by NLP4J[26]. Depending on the needs of the text processing task that remain to be investigated, more core natural language processing tools may be added in the future.

5.2.2 Severity Classifier

A software bug can impact development in many ways. In bug tracking systems, such as Bugzilla, bugs are often assigned a severity label based on the impact that the bug may have for the project. However, filling in this label is not mandatory for posting a bug and it is often omitted. Within the context of CROSSMINER, the severity of a bug is a vital piece of information that can be used to influence decisions made by open source software developers. The severity classifier is a tool that can predict the severity of a bug found on a communication means. We have designed a severity classifier that considers seven levels of severity: *normal*, *enhancement*, *major*, *minor*, *critical* or *blocker*.The tools uses six *one-vs-rest* Support Vector Machines (SVM) organised in a tree structure.

5.2.3 Emotion Classifier

An emotion classifier is a tool that determines the emotions that are expressed in an input text. Similar to the sentiment analyser, it is a tool that helps to quantify the opinion of users for a software project. Unlike OSSMETER, which used a lexicon-based emotion classifier, in CROSSMINER we have trained a neural network that allows estimating the emotions expressed in a text, e.g. *love*, *joy*, *anger*, *sadness*, *fear* and/or *surprise*.

5.2.4 Request Reply Classifier

Unlike Q&A websites, where, by definition, only one question is allowed per post[27], in other sources of data explored in CROSSMINER, it is possible to have multiple questions/requests combined with answers/replies. Detection of these requests and replies can improve the management of the communication means of a project. Developers can know whether posts in sources, such as forums or issue trackers, contain multiple request and whether these have been replied. In addition, external developers, potential adopters of the project, can determine how well the community is managed, and whether their requests would be replied. We have developed and integrated into CROSSMINER a request/reply classifier based on a linear Support Vector Machine (SVM).

[26] emorynlp.github.io/nlp4j/

[27] In some cases, users can ask questions to the author of another question or an answer, in the form of comments. However, these questions usually aim at clarifying the initial one.

5.2.5 Content Classifier

Threaded messages contain indicators that can help developers evaluate an OSS project. Messages can be of different content/functionality types. For example, a message can propose a solution to a previously reported problem, or it may reinforce a previous question that was not answered, or verify that a previously proposed solution actually worked and addressed a problem. As part of OSSMETER, a detailed hierarchy of content types was constructed. In CROSSMINER, we used this hierarchy to develop a content classifier, i.e. an automated tool that assigns one or more class labels to a message based on its content. Applying the content classifier to communication channel articles and bug tracker comments provides with useful statistics about the kind of interaction taking place. The content classifier that was developed and integrated in CROSSMINER utilises a multi-class SVM.

5.2.6 Sentiment Analyser

A sentiment analyser is a tool that determines whether a text expresses positive or negative sentiment, or is neutral, i.e. does not express any sentiment. In the domain of software engineering and software development, sentiment analysis is a useful tool that can provide a general estimation of the feelings of users about a software project. In other words, by applying a sentiment analyser on text from a source, such as Bugzilla or Eclipse Forums, we can determine whether users are satisfied or not through the polarity of sentiment expressed in messages. The sentiment analyser developed for and integrated into CROSSMINER is a multi-class classifier based on VastText, a neural network developed in house.

5.2.7 Thematic Clustering

Due to the large quantity of information available along all the sources, it is sometimes necessary to know, *grosso modo*, which are the topics/subjects discussed. By knowing the topics discussed by users, developers can improve the management of a project. In other words, thematic clustering allows identifying trends of errors, discussions or thoughts. Therefore, in CROSSMINER, we provide a thematic clustering tool. The tool uses *Carrot*^{2}, an open source clustering tool that groups documents into labelled clusters according to their topic. Cluster labels are determined automatically by the *Lingo* algorithm, which internally scores terms to find the candidate that bests represents each group of cluster.

5.3 Index Manager

The Index Manager is a new component developed for CROSSMINER. It provides the facilities to manage, interact and populate the ElasticSearch cluster contained within the knowledge base. It was developed following the singleton design pattern and using the tool-kit exposed by the *org.eclipse.scava.elasticsearch.dependencies* package. This approach restricts the instantiation of the *IndexManager* class to a single object that is responsible for coordinating all interactions between the CROSSMINER platform and the ElasticSearch cluster, as shown in Figure 8.

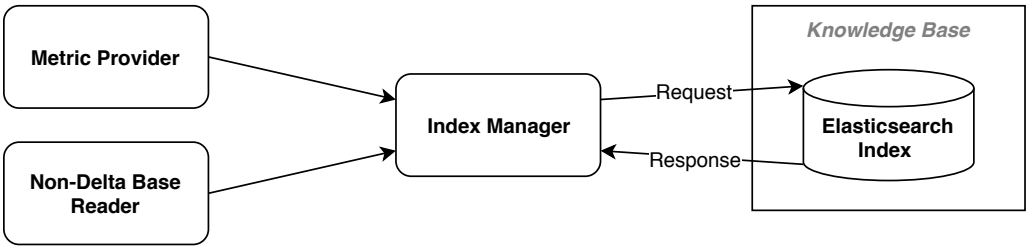

Figure 8: Index Manager

The Index Manager is designed to interact with an ElasticSearch cluster running version 6.3. The toolkit integrated into CROSSMINER contains all the necessary libraries associated with version 6.3 of the "High-level Rest Client API" and "Java API" (transport client).

Currently, the Index Manager publicly exposes two static methods called *performIndexing* for indexing a document. These method automatically handles the steps for indexing a document including the creation of the index, adding mappings, converting objects to JSON and creating or updating elements within the ElasticSearch cluster. The methods require arguments such as: *indexName*, *mapping (optional)*, *documentType*, *uid*, and *document*. Each are described in Table 24.

Parameter	Type	Description
Index Name	String	The name of the index you wish to add the document to
Mapping	String	(*optional*)A string in JSON format that contains information information regarding the data type of the fields contained within a document.
Document	object	A Java object that contains the information you wish to index
Document Type	String	The type of the document, e.g. 'github_issue'
Unique ID	String	A uniquely identifiable string for the document

Table 24: *performIndexing* Arguments

The Index Manager is also capable of supporting various configurations of ElasticSearch depending on the user's specific requirements. Users of the CROSSMINER platform are required to modify/populate the *elasticsearch.properties* file located in the *elasticsearch.properties* directory to provide the tool with the correct configuration settings for their ElasticSearch cluster contained within the knowledge base. This is a vital step to ensure that the Index Manager is capable of interacting with the user's ElasticSearch cluster.

In relation to the other natural language components developed as part Task 3.3., the Index Manager is employed by the delta-based workflow, as shown in Figure 4, and also by two transient metrics responsible for document preparation. It is also used as a component in the non-delta based workflow, as shown in Figure 6, for enriching the knowledge base with information from "social media" and "question & answer websites". The latter plays a critical role in our future D3.5.

5.4 Summary

This section has presented a collection of tools integrated into CROSSMINER that are utilised by numerous metric providers to facilitate access, pre-processing, processing and indexing of natural language sources.

6 Risks and Limitations

The purpose of this section is to discuss the risks and limitations related to this deliverable. Due to the nature of software development and the design choices that were made, the components discussed in this deliverable have limitations and are associated with risks, as shown below:

Risks:

- The APIs and methods used to access information may become outdated / deprecated.
- The number of API calls allowed for sources may be reduced by the API policy makers.
- Discontinuation of support for libraries used for natural language processing tools.

Limitations:

- For readers that utilise dumps/archives, manual intervention is required when new ones are made available.
- Large volumes of archived data from non-delta based readers may take a long time to process.
- The size of classification models of our tools may increase the overall system requirements for CROSSMINER.
- Information contained within dumps/archives is not always up-to-date.

7 Conclusions

To conclude, this deliverable has discussed in detail the natural language components integrated into CROSSMINER as a consequence of the requirements provided by our use case partners. Presented below is a summary of the outcomes of this deliverable with respect to each of the natural language components discussed throughout this report:

- **Readers:**
  - Two methods for enriching the knowledge base: via delta based and non-delta based readers
  - CROSSMINER now has the capabilities to mine information from Bug tracking systems, Communication channels, Social Media and Question & Answer Websites.
  - Integrated a total of 17 readers
- **Metric Providers:**
  - Developed various new metric providers to support forums, a new type of communication channel
  - Upgraded several natural language processing metric providers to utilise: text pre-processing tools and/or the core set of NLP tools
  - Migrated / integrated metric providers associated with: Bug tracking systems, Newsgroups and Forums
  - Included transient and historic metrics that prepare documents for indexing
- **Tools:**
  - Developed and integrated a core set of natural language processing tools.
  - Integrated several new and improved tools, developed in Task 3.1.
  - Developed and integrated an Index Manager, i.e. a component that adds support for populating the ElasticSearch cluster contained within the CROSSMINER knowledge base.

7.1 Work Package 3: Progress

The objective of this section is to provide an insight into the overall progress of Work Package 3 (WP3). WP3 has a total of four tasks and five deliverables. To date three tasks have been completed and four deliverables have been submitted. Each completed deliverable is summarised below.

- **D3.1** This deliverable reported the current progress of Task 3.1 and includes an investigation of state-of-the-art methods for text representation as well as presented experimentation with English vs Code classifier. This deliverable provided the part of the foundations for D3.3 and was delivered on time in M12.
- **D3.2** The focus of this deliverable was to investigate methods for reading and searching text sources associated with OSS projects. It identified various new sources of information, highlighted issues with existing readers and provided insight to which readers could be migrated from OSSMETER to CROSSMINER. It also explored how two new source types can be exploited in CROSSMINER. Finally, it presented an analysis of two open source search engines with regards to their suitability for CROSSMINER. Its purpose was to provide the foundations for deliverables D3.3 and D3.4 and was delivered on time in M18 of the project.
- **D3.3** Since task one spanned over 24 months, this deliverable represents the completion of Task 3.1 and presents the experimentation and results in relation to natural language tools developed for CROSSMINER. This deliverable was delivered on time in M24 of the project.

- **D3.4** This deliverable is associated with design, development and integration of the various natural language related components into the CROSSMINER platform. This deliverable was also delivered on time in month 24 of the project.

With regards to the current overall progress of WP3, there has been significant advancements made towards the completion of the requirements that its is linked to. Requirements associated to this work package are divided into two areas: those provided by CROSSMINER use case partners and those provided by technology partners . The progress for each area of requirements is presented in Table 25 and 26, respectively.

Ref	Description	Priority	Progress	Notes
3. Natural Language Searching				
U30	Able to search "part of speech" parts of code	Should	Planning	Task 3.4. Involves the analysis of comments
U31	Able to search mailing-lists	Shall	Completed	Task 3.3
U32	Able to search forums	Shall	Completed	Task 3.3
U33	Able to search issues	Shall	Completed	Task 3.3
U34	Able to search configuration files	Shall	In Progress	Task 3.4. Involves the analysis of comments
U34	Able to search documentation	Shall	Planning	Task 3.4
U36	Able to search a blog	Shall	Planning	Task 3.4. Involves the analysis of snippets
U37	Able to search IRC	Shall	Completed	Task 3.3
U38	Able to search simultaneously across all the data sources	Shall	Completed	Task 3.3
U39	Able to present the results by data source or globally for the project	Shall	Completed	Task 3.3
U40	Able to order the results by relevance	May	In Progress	To be investigated further in Task 3.4
U41	Able to search using regular expression patterns	Shall	Completed	Task 3.3
U42	Able to detect in the data sources text referring to one or several bugs	Shall	In Progress	Just the implementation stage to be completed
U43	Able to detect in the data sources text referring to one or several commits	Shall	In Progress	In collaboration with CWI. Just the implementation stage to be completed.
U44	Able to extract sentiment analysis from the communication data sources	Shall	Completed	Task 3.3
U45	Able to extract sentiment analysis from wikis	Should	In Progress	Task 3.4. The Sentiment Analysis component is completed. A reader for wikis to be implemented.
U46	Able to detect similarities in messages so as to suggest answers	Should	Completed	Task 3.3
U47	Able to propose recommendations to improve community management	Should	Completed	Task 3.3

Ref	Description	Priority	Progress	Notes
U48	Able to identify recurring problems identified by users	Should	Completed	Task 3.3
U49	Able to detect in the data sources one or several commits hashes	Shall	In Progress	Task 3.4. In collaboration with CWI. Implementation stage to be completed.
U50	Able to list commits with bugs	Shall	In Progress	
U51	Able to list bugs with commits	Shall	In Progress	
U52	Able to list unattended bugs	Should	Completed	Task 3.3
U53	Able to list unattended messages	Should	Completed	Task 3.3
U54	Able to create a cluster map for messages using Lingo or similar clustering technology	Should	Completed	Task 3.3
U55	Able to create a cluster map for bugs contents using Lingo or similar clustering technology	Should	Completed	Task 3.3
U56	Able to create a cluster map for commit messages contents using Lingo or similar clustering technology	Should	In Progress	Task 3.4. In collaboration with CWI
U57	Able to identify the list of changed third-party API methods from GitHub issues	Should	In Progress	
U58	Able to identify that the topic of a forum thread is the migration of a given third-party API	Should	In Progress	
U59	Able to identify code snippets that use old and new third-party API in forum threads concerning migration of the usage of the given third-party API	Shall	In Progress	
U60	Able to analyse level of user activity in issue trackers	Shall	Completed	Task 3.3
U61	Able to identify if a thread reached a conclusion and if it was positive	Should	Completed	Task 3.3
4. Analysis of Docs/Code Snippets				
U62	Able to extract text from HTML and markdown to feed natural language analysis and identify code snippets	Shall	Completed	Task 3.4
U63	Able to extract text from PDF to feed natural language analysis and identify code snippets	May	In Progress	Task 3.4
U64	Provides recommendations to add documentation commonly found in successful projects	Should	Planning	Task 3.4
U65	Provides recommendations to improve the structure of the documentation	Should	Planning	Task 3.4
U66	Able to analyse Java code snippets	Shall	Planning	Task 3.4
U67	Able to analyse JavaScript code snippets	Should	Planning	Task 3.4
U68	Able to analyse C code snippets	Shall	Planning	Task 3.4
U69	Able to analyse PHP code snippets	Shall	Planning	Task 3.4

Ref	Description	Priority	Progress	Notes
U72	Able to determine migration pattern from two (or more) code snippets when one of them uses the old third-party API and the other uses the new third-party API	Shall	Planning	Task 3.4
U73	Able to identify the part of the API that the developer is currently using to provide code snippets in relation with current development activity	Shall	Planning	Task 3.4
U74	Able to analyse the API documentation (when available) and determine if it matches the current API of the library	May	Planning	Task 3.4
8. Project Documentation Analysis				
U110	Able to analyse PDF documents	May	Planning	Task 3.4
U111	Able to identify the documentation contains a Getting Started	Should	Planning	Task 3.4
U112	Able to identify if the documentation contains a User Guide	Should	Planning	Task 3.4
U113	Able to identify if the documentation contains a Developer Guide	Should	Planning	Task 3.4
U114	Able to identify if the documentation contains code snippets	Should	Planning	Task 3.4
U115	Able to analyse if the documentation has a license	Shall	Planning	Task 3.4
U116	Able to analyse readability of documentation	Shall	Planning	Task 3.4
U117	Able to identify the list of changed third-party API methods from documentation	Should	Planning	Task 3.4
U118	Able to identify the list of deprecated third-party API methods from documentation	Should	Planning	Task 3.4
U119	Able to look up the documentation of the public API provided by a project	Shall	Planning	Task 3.4
U120	Able to look up the source code examples of a public API of project	Shall	Planning	Task 3.4
U121	Able to analyse documentation to collect information related to the usage of the API provided by the project	Should	Planning	Task 3.4

Table 25: WP3 Use Case Requirements - Current progress

Ref	Description	Priority	Progress	Notes
D32	The NLP analysis component should record the sources used by the developer and use them as a form of automatic feedback to improve its suggestions.	Should	In Progress	Task 3.3. In collaboration with FEA

Ref	Description	Priority	Progress	Notes
D72	The other mining tools developed in WPs 2-4 shall expose a REST API that the cross-project relationship miners can consume	Shall	Completed	Task 3.3
D73	The other mining tools developed in WPs 2-4 shall expose project level metrics that cross-project relationship miners can consume	Shall	Completed	Task 3.3

Table 26: WP3 Technology Requirements - Current progress

At the time of writing this deliverable, there is one ongoing task in WP 3, Task 3.4. This task focuses on the development of a recommender that suggests code snippets and discussion from social media relevant to the code that is being developed in the Integrated Development Environment (IDE). For the recommendation we shall consider a variety of resources including code, documentation and online discussions. The deliverable associated with this task is due during M30 of the project.

7.2 Future Work

With regards to WP3, our efforts will be divided between several areas. As discussed in Section 3, GitLab and Bugzilla readers will be developed and integrated into CROSSMINER once RESTmule has the capabilities to support both sources. Metric providers associated with forums will be tested and integrated once the issue with the forums has been fixed. We will also address any issues that arise due to modifications to other CROSSMINER components that have been developed outside the scope of this deliverable. We need to conduct further investigation and discussions surrounding requirement D32, shown in Table 26. However, the majority of our efforts will focus on Task 3.4, the development of a recommender that suggests code snippets and discussions from social media relevant to the code that is being developed in the Integrated Development Environment (IDE).