CROSSMINER

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

Project Number 732223

D7.10 Eclipse-based CROSSMINER IDE - Final Version

Version 1.0
28 June 2019
Final

Public Distribution

FrontEndART

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 3621 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	

Table of Contents

Document Control

Version	Status	Date
0.1	Initial baseline of content from D7.2	14 June 2019
0.8	Version to be reviewed	24 June 2019
1.0	Final version	28 June 2019

Executive Summary

The Eclipse-based CROSSMINER IDE component of the CROSSMINER platform provides end user graphical user interface for the platform, and is implemented in task T7.5 (Eclipse-based CROSSMINER IDE) of work package WP7 (Advanced Integrated Development Environments). This component is one of the primary interaction interfaces between the user (such as the developer) and the CROSSMINER back-end services, such as the knowledge base. The component is basically a standard Eclipse Integrated Development Environment extended with a CROSSMINER plug-in ("*CROSSMINER Eclipse IDE Plug-in*" in the following). It will provide different features to access the platform functionalities, which will be usable in different scenarios.

This document presents deliverable D7.10 (Eclipse-based CROSSMINER IDE - Final Version), which is a feature complete implementation of the Eclipse-based CROSSMINER IDE. The deliverable covers the plug-in related technology and use case requirements defined in deliverable D1.1 (Project Requirements). The CROSSMINER Eclipse IDE Plug-in uses the proposed communication interfaces and is mostly integrated with other modules of the CROSSMINER platform, however, integration is not yet fully tested.

1 Introduction

The Eclipse-based CROSSMINER IDE is one of the front-ends of the CROSSMINER platform, implemented as an Eclipse plug-in. It provides an interface to the developers through the widely used Eclipse IDE, which helps them to access the various features of the platform. It offers features useful especially for the developers, e.g. searching, installing or upgrading libraries and projects, accessing various code recommendations, relevand documentations or posts. It also provides anonymous user activity monitoring features that can help researchers and managers to analyze the activity patterns developers follow during development.

Figure 1 shows how the CROSSMINER Eclipse IDE Plug-in is connected to the CROSSMINER platform. It is connected to the Logic Layer (the CROSSMINER server) through the common CROSSMINER API. The CROSSMINER Eclipse IDE Plug-in uses the latest version of the API (although late changes might not be considered, these issues, if raised, will be handled in the platform integration period).

In Section 2 we describe how this deliverable improves the state of the art/practice, in Section 3 the features of the Eclipse-based CROSSMINER IDE are described, and finally in Section 4, we show how the original project requirements are fulfilled.

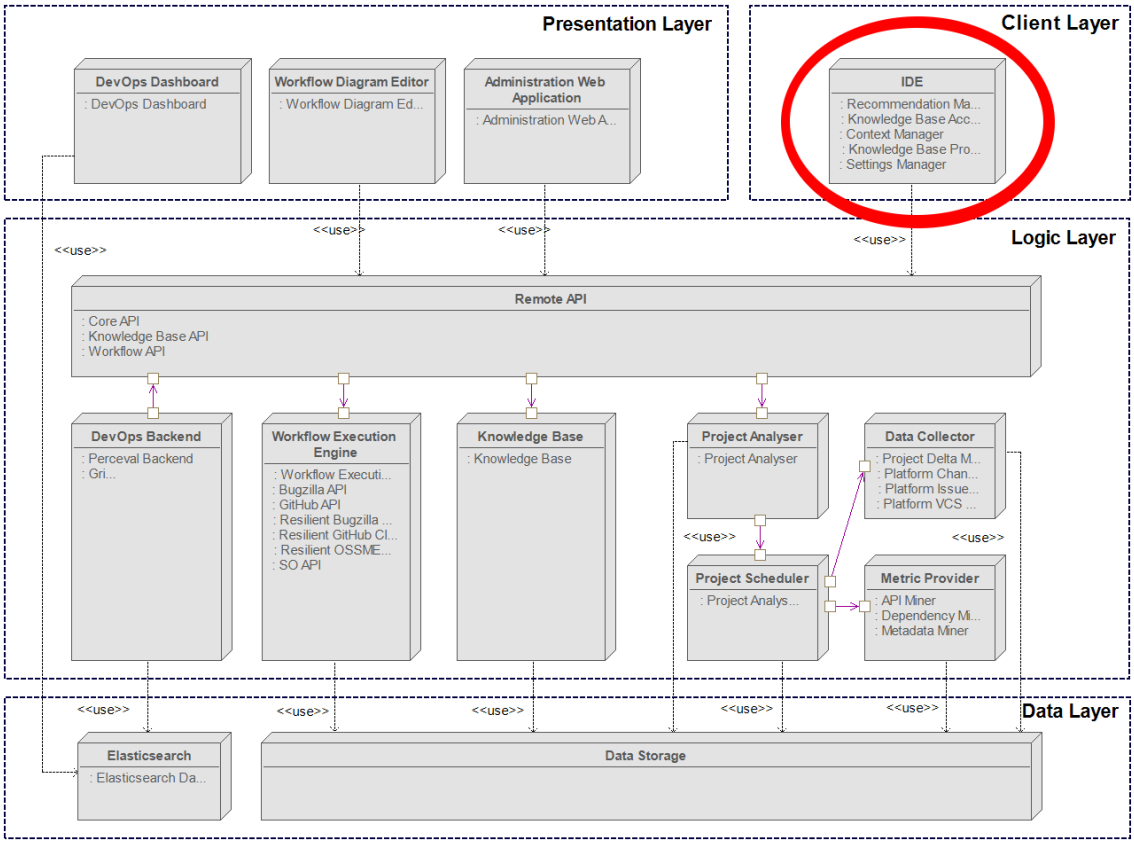

Figure 1: Location of the Integrated Development Environment in the CROSSMINER platform

2 Innovation

The whole CROSSMINER platform is innovative in the sense that it utilizes several techniques to automate some parts of the recent practices of open source development. Namely, open source development often relies on reusing existing solutions, which requires knowledge of the reuseable artefacts. Searching for available components, gaining knowledge on them in order to choose the right one is a time consuming task that involves searches in blogs, discussion forums, documentations, etc. The implementation of some (general) features using the selected components has probably done by others already, thus, searching for solutions is also a frequent activity of developers.

The CROSSMINER platform automates these tasks and combines the results of different sources, thus, it can speed up development. There are several solutions integrated in various IDEs that can offer some help on writing code (e.g. automatic word extensions), but these are usually "local" or "wired" solutions and based on a limited knowledge base. In contrast, the CROSSMINER platform can collect information from several sources (e.g. from code on GitHub, from discussions on StackOverflow). This information is stored in a Knowledge Base, and can be accessed through the CROSSMINER REST API. Eclipse-based CROSSMINER IDE utilizes this API to provide relevant information to the developers during development, that helps and speeds up their work.

3 Technical documentation

In this section, we present the features of the CROSSMINER Eclipse IDE Plug-in. The features will be presented in four main categories: installation (3.1), recommendation-related features (3.2), user activity moditoring related features (3.3), and plug-in settings (3.4).

3.1 Installation

CROSSMINER Eclipse IDE Plug-in will be distributed in the common Eclipse update site format. Update sites are used to organize and export features so they can be installed into Eclipse. A feature is used to package a group of plug-ins together into a single installable and updatable unit. Features have a manifest that provides basic information about the feature and its content. It may include plug-ins, fragments and any other files that are important for the feature. A feature can also include other features. The delivery format for a feature is a JAR, but each included plug-in will be provided as a separate JAR.

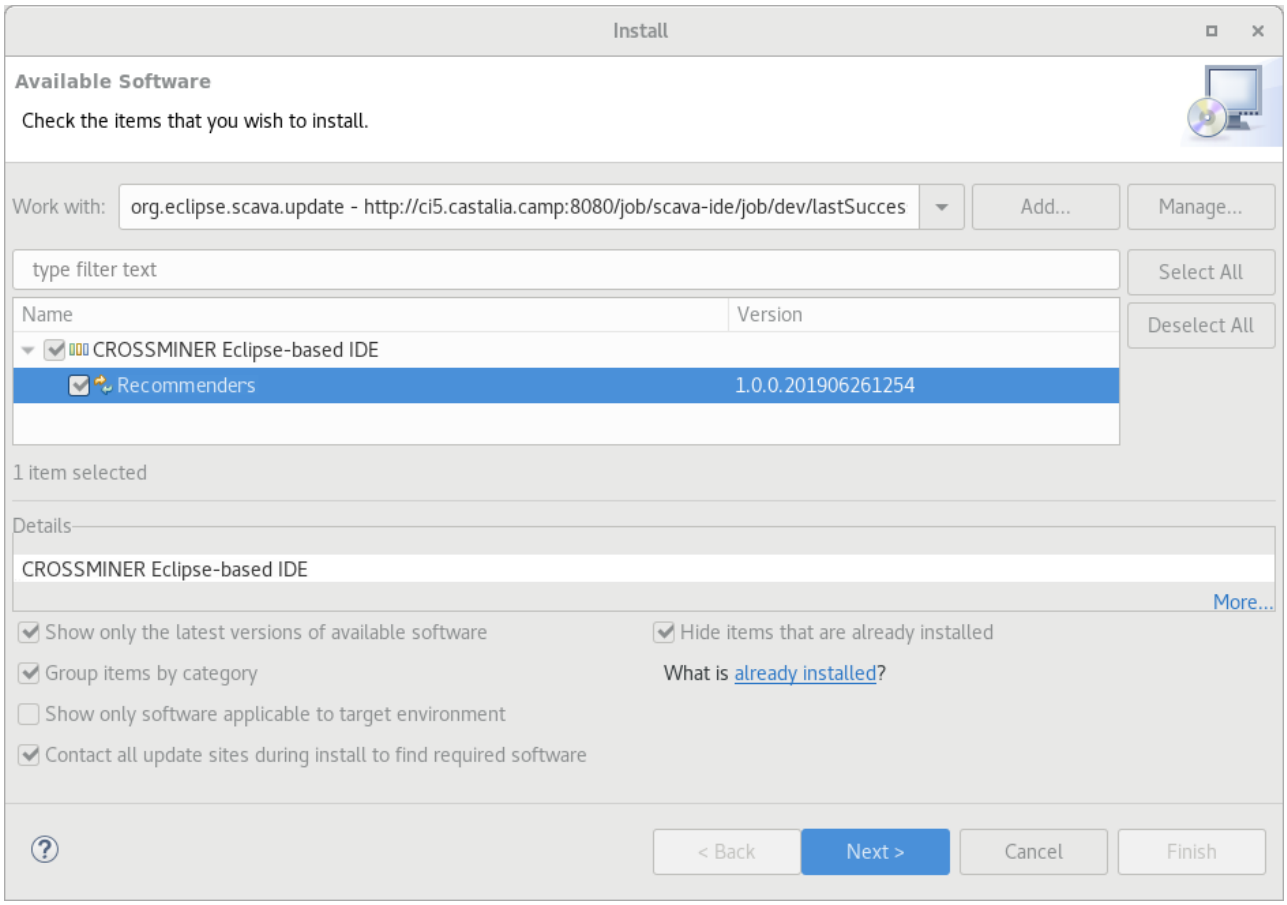

Figure 2: Installation of the CROSSMINER Eclipse IDE Plug-in

To install the plug-in via an update site select the *Install new software* under the Help menu in Eclipse. As you see on Figure 2, in the following window you can choose the location of the update site[1]. The location can be a remote or local one. After you select the update site's location it will be shown in the list. You can browse the plug-in's features and disable it if it is not necessary for you. When you are ready, click the *Next* button. On the following screen you have to accept the license agreement. On the last screen you can see the installation details, including a list which contains all of the selected features for your installation. To finish the installation process click on the *Finish* button.

[1]Currently the update site is available on the project's CI server at `http://ci5.castalia.camp:8080/job/scava-ide/job/dev/lastSuccessfulBuild/artifact/eclipse-based-ide/org.eclipse.scava.root/releng/org.eclipse.scava.update/target/repository/`. It can also be directly built from `https://github.com/crossminer/scava/tree/dev/eclipse-based-ide`

3.2 Recommendations

There are several kinds of suggestions and recommendations which could be helpful for the developer during their daily tasks. In this chapter we introduce those features of the CROSSMINER Eclipse IDE Plug-in that result some kind of recommendations. Based on the type returned by the recommendation, the CROSSMINER Eclipse IDE Plug-in provides interfaces for library or project recommendations, source code recommendation, and text-based recommendations. No mixed recommendations can be asked for in the CROSSMINER Eclipse IDE Plug-in. The following subsections will detail the features connected to these recommendations.

3.2.1 Library or Project Based Recommendations

Software developers tend to reuse already written components. These are usually encapsulated in one or more third party libraries. Recommendation whose subjects are these third-party libraries are called *library or project based recommendations*. This feature helps you to search related libraries or third party projects for your project.

3.2.1.1 Project search The goal of this feature is to search for projects that can help you to implement some features in your software. You can give keywords on what you need, and the knowledge base will return projects that might be helpful in your software. To search for projects which can be installed into your system, select the *Project Search* feature under the CROSSMINER menu. After the dialog (see Figure 3) is opened, you can type the desired search expression into the search field.

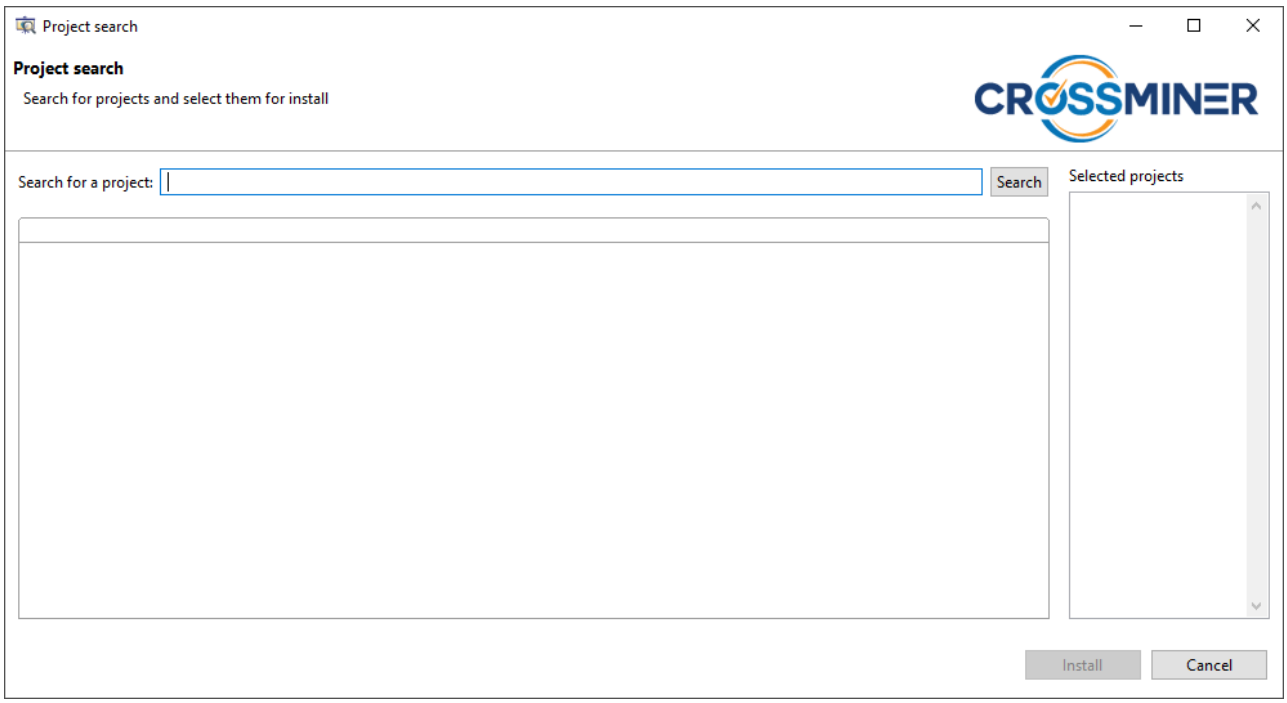

Figure 3: Project search dialog

After that, hit enter or click on the *Search* button. The results are going to show up in a new tab in the middle of the dialog, as shown in Figure 4. Every additional search is going to open a new tab and the results are shown inside that.

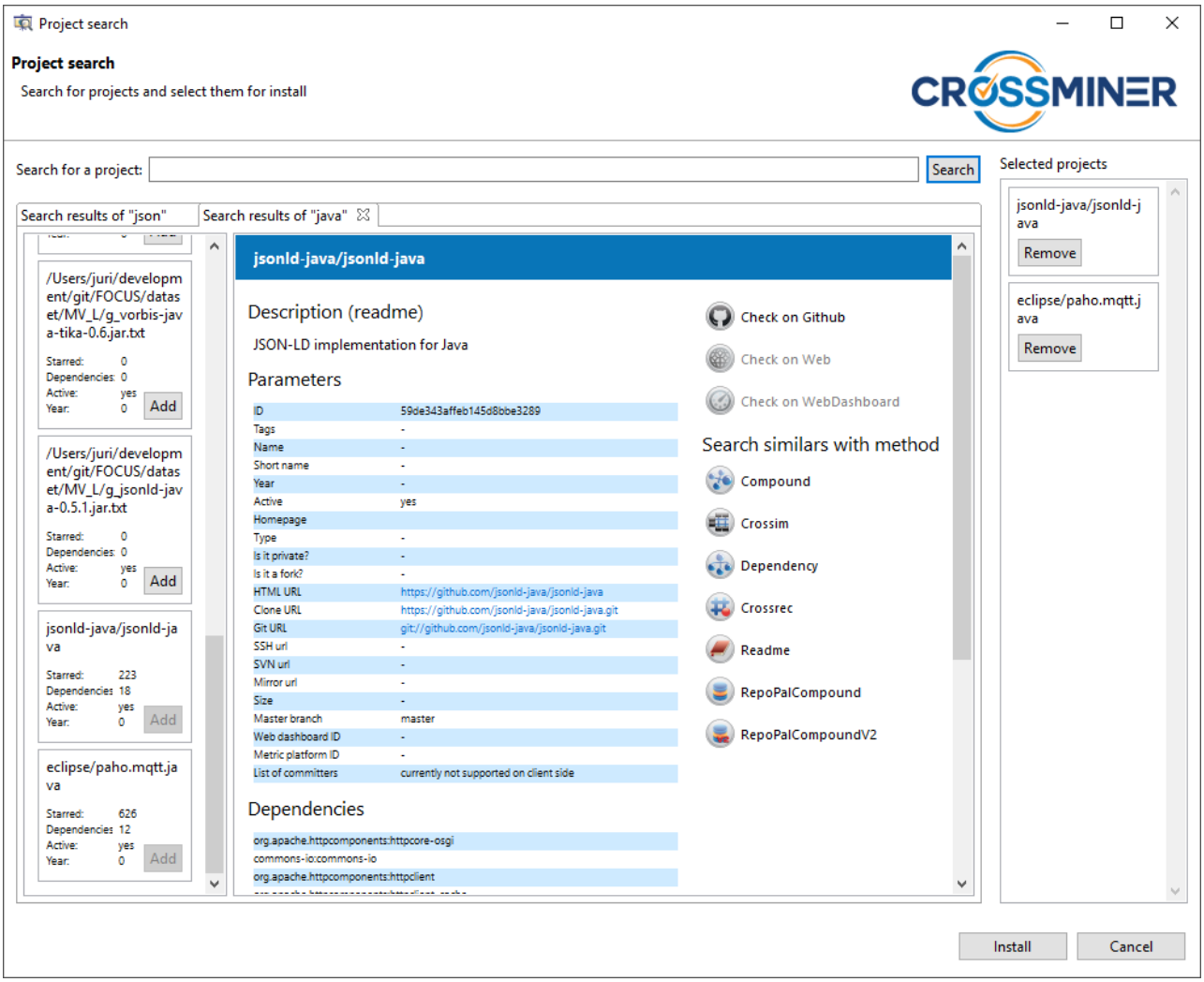

Figure 4: Project search dialog – search

On the left part of this tab you can see the results list. By clicking on these items a couple of information are shown in the right part of the search tab, like a description or the dependencies of the selected project. You can navigate to the GitHub repository, to the official website of the project or to the related CROSSMINER Web-based Dashboard site. You can initiate a new search for similar projects to the given project by clicking on the similarity method buttons. These are the *Compound*, *Crossim*, *Dependency*, *Crossrec*, *Readme*, *RepoPalCompound*, and *RepoPalCompoundV2*. Alternatively, you can select the given project result to be installed by clicking on the *Add* button in the results box. The selected projects are shown on the right side of the dialog in a list. By clicking on these items you can view the details of them just as in the previous case. You can deselect these items by clicking on the *Remove* button.

After you have chosen the projects to be installed click on the *Install* button at the bottom of the dialog, and the install dialog, shown in Figure 5, will appear.

In the install dialog you can set the base path to the location where you want to install the projects by default. After updating this path, all of the projects destination location is going to be updated. You can see this property of the projects on the bottom of the dialog, in the project list. All of the projects can be installed into different locations by giving them a custom destination path. After setting all of the desired installation paths, you can choose to install all the projects at once by clicking on the *Install all* button or you can install them one-by-one manually, by clicking on the *Install* buttons of the projects. The installation process can be canceled by clicking on the *Cancel* buttons. After installation, the projects can be examined or imported from the given locations.

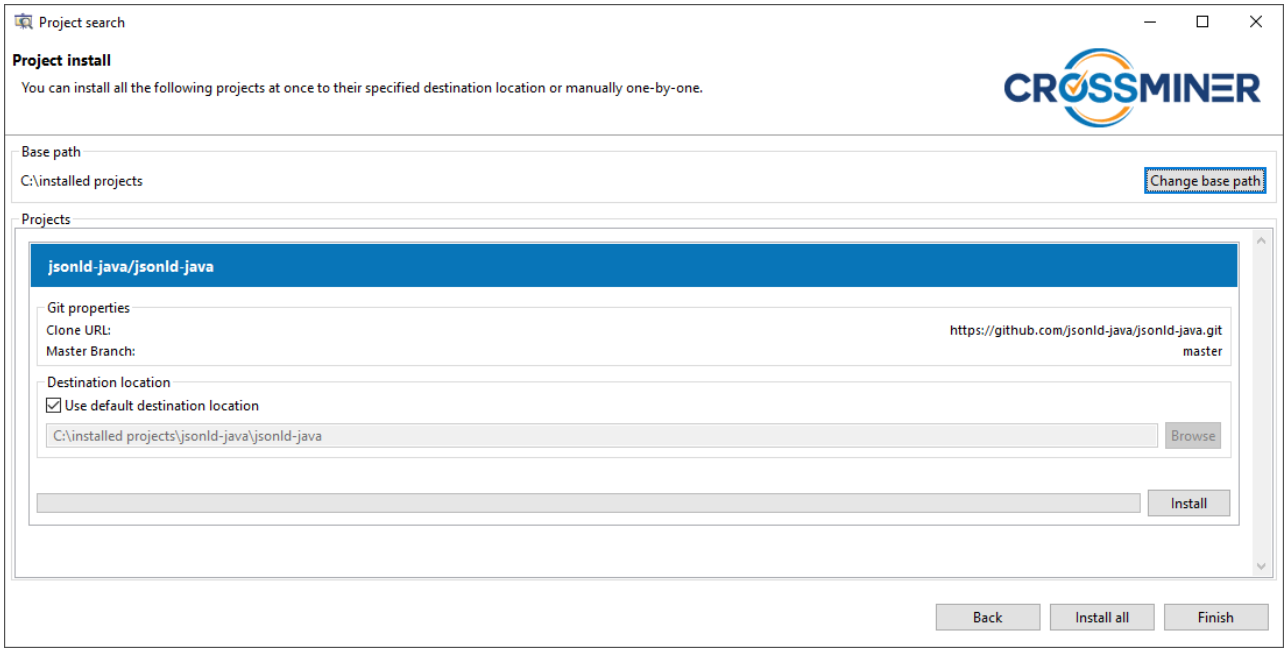

Figure 5: Project search dialog – install

3.2.1.2 Searching additional libraries To search for libraries that can be useful during the development of your project, open the *Library search* dialog. For this, first you have to select the project in the *Package explorer* view. This feature works only with Maven projects, that contains their `pom.xml` file in their root folder. Then navigate in the menu to the CROSSMINER menu and click on the *Search libraries* option. This will bring up the *Library search* dialog.

This dialog is split into two parts. On the left side you can see the currently used libraries in your project and on the right side there are the suggested libraries. Both the left and the right side contains their libraries in two separate groups. On the left side, in the upper group there are the libraries that has been selected to be used as the base of the search. By default all of the used libraries are selected to be used as a base for the search. The dialog at this state can be seen on the Figure 6.

The recommended libraries may vary depending on the set of already used libraries selected as a base for the query. The not yet selected libraries are on the bottom of the dialog. The libraries that

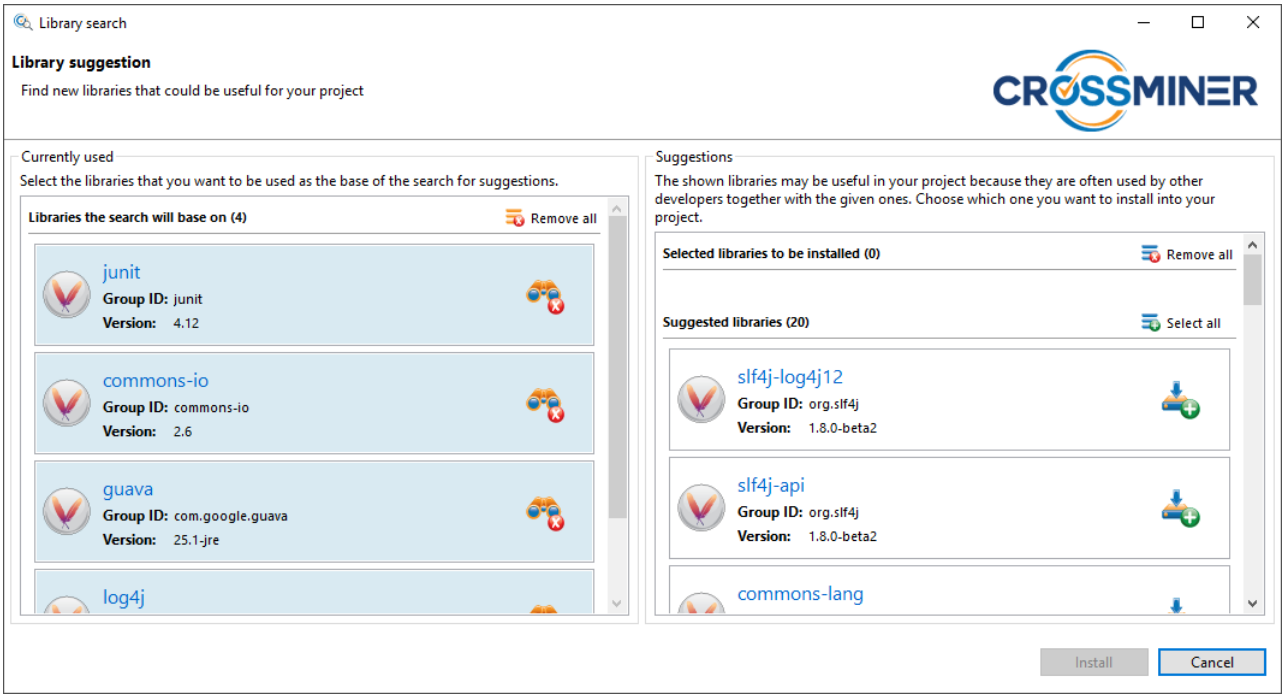

Figure 6: Library search dialog

CROSSMINER recommended to be used in your project are on the right hand side of the dialog (as can be seen on Figure 7). The recommended libraries can be selected for installation, in which case they are moved to the top of the right hand side of the dialog (see Figure 8).

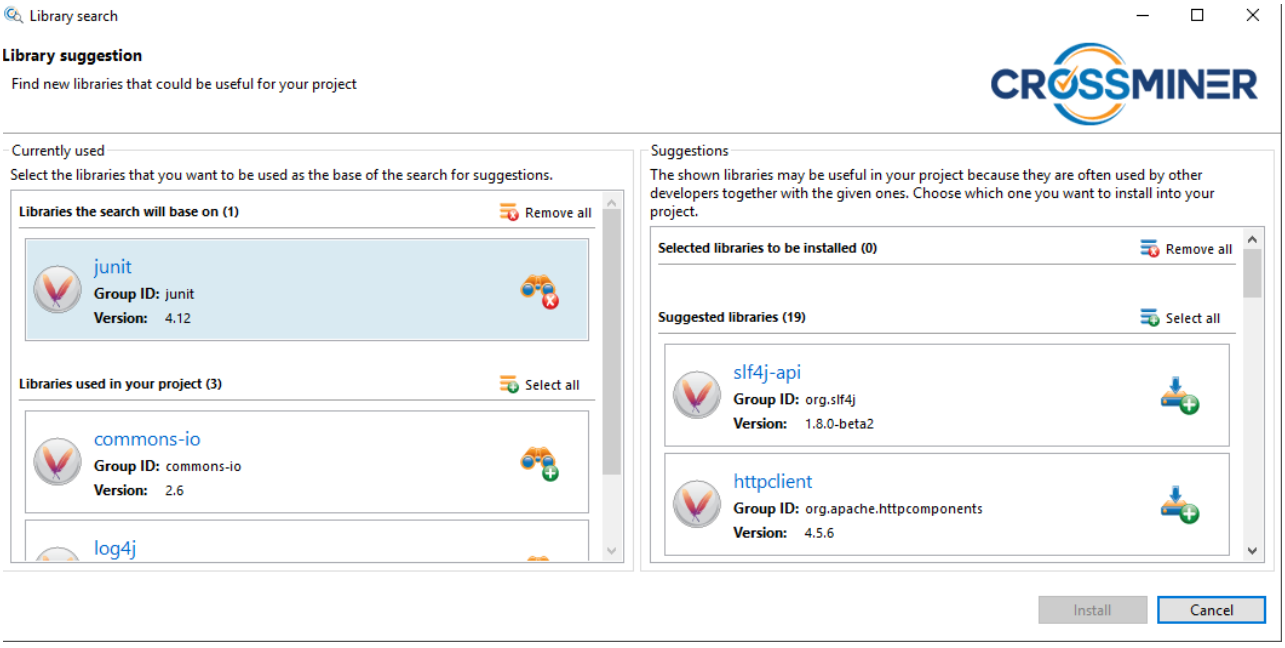

Figure 7: Library search dialog – results

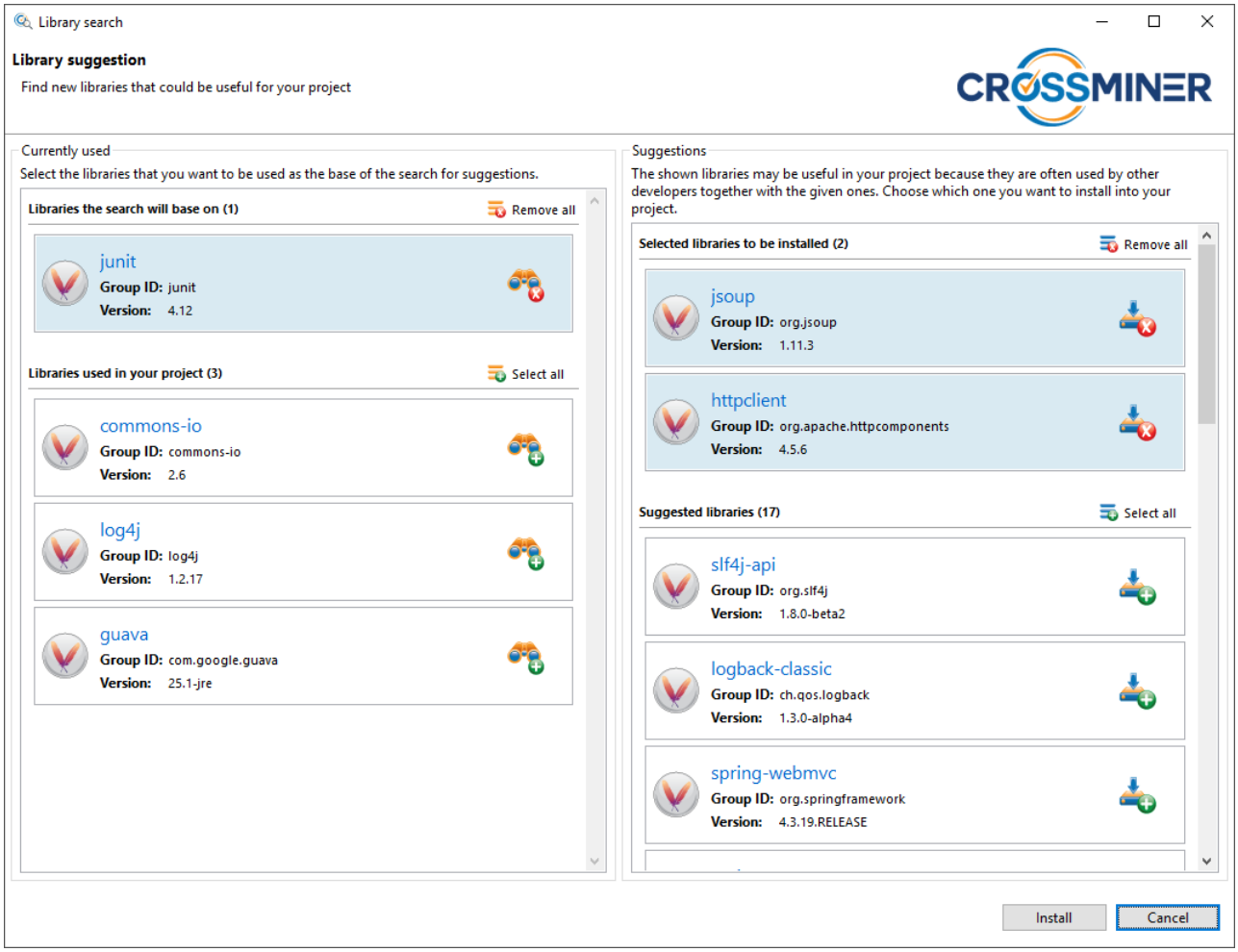

Figure 8: Library search dialog – selected to install

The recommended libraries are refreshed every time the search parameters (i.e. the set of libraries that are used as the base of the search) are changed. In case the library recommendation cannot be retrieved from the CROSSMINER server, a *Try again* button will be shown to perform a new request (as shown in Figure 9).

After selecting the set of libraries that you want to use in your project, by clicking on the *Install button* you can add them to your project. They will show up among the dependencies in the `pom.xml` of the project.

3.2.1.3 Handling Changed and Deprecated APIs

Third party libraries are prone to change and evolution. To help the developers to adapt their project after these changes, we defined an other sub-category of recommendations, namely when their goal is to provide information about modified or deprecated interfaces. A new version of a used library can be selected, and the CROSSMINER Eclipse IDE Plug-in will retrieve a list of those elements that cannot be used in the same way as in the previous version. The usage of these items will be marked in the sources of the project under development.

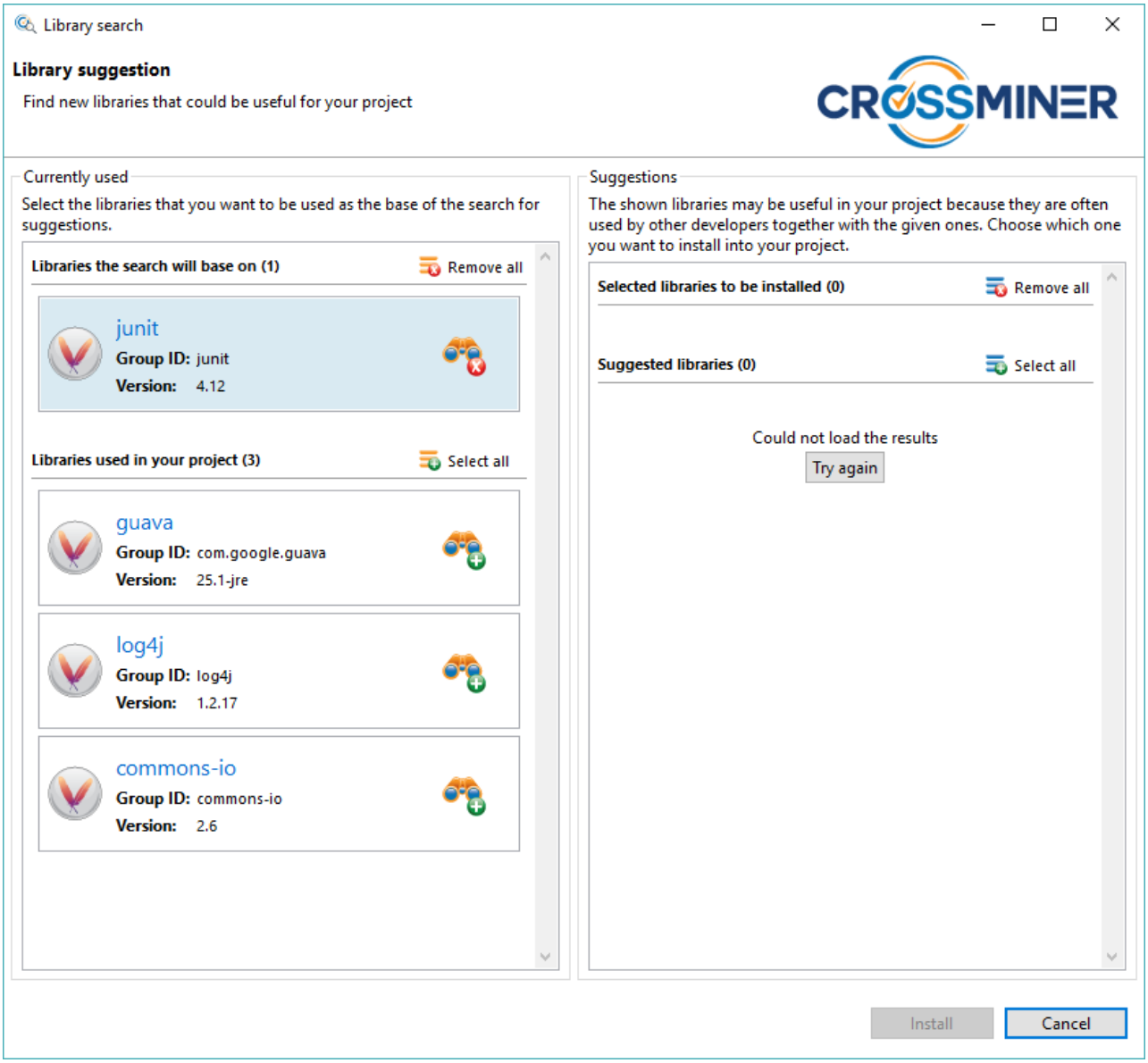

Figure 9: Library search dialog – try again

3.2.2 Source Code Based Recommendations

In this section we elaborate features related to those recommendations whose subject entities are present in the source code of the project under development. They usually retrieve some code chunk, which could be annotated to ease further understanding.

3.2.2.1 Code recommendation Code recommendations are used to retrieve code snippets which may be useful for you by showing an example how to use a specific library function or just presenting a pattern that can be followed. These snippets can be requested by selecting a chunk of code in an open Java source code editor and by clicking on the *Request code recommendation* option in the

CROSSMINER sub-menu of the context menu, or by using the *Ctrl+Shift+X C* key combination. After the *Code recommendation* view has shown up, the suggested patterns for the selected pattern will be displayed on the left side in a hierarchical representation. The results are grouped by the file they are relevant to, and by the query which represents the request for the recommendations at a given time using a given part of the file. This can be seen in the Figure 10.

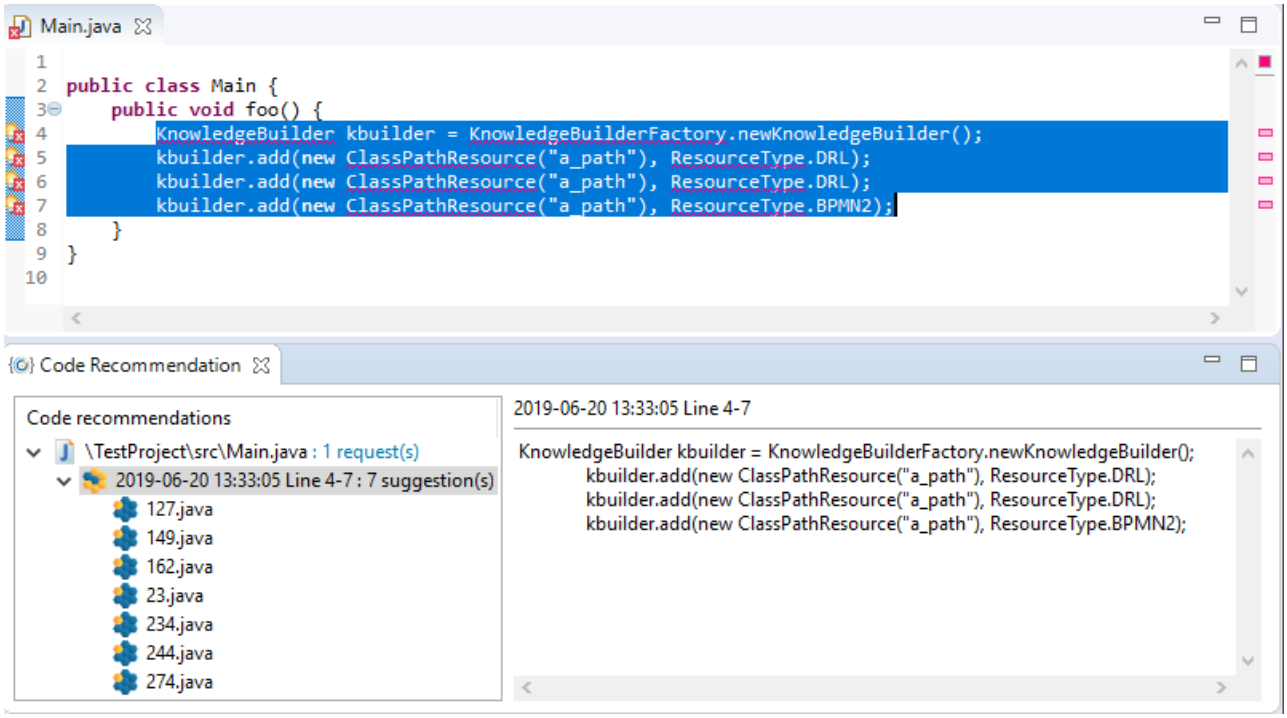

Figure 10: Code recommendation view – suggested patterns

By selecting a pattern in the hierarchical tree view, the recommended snippet will be shown in the preview view on the right side of the *Code recommendation* view. Here, you can examine the given code, copy a part of it, or, by using the *Insert snippet at cursor* button you can paste it into the active editor at the cursor's position or replace the selected code chunk with it. This functionality is available only when there is an active editor opened. The preview view can be seen on the Figure 11.

You can also remove the recommended code snippets from the results by opening the context menu over them and choosing the right option for you. Basically there are 4 options that you can go with. If you select a file you have the option to drop all the recommendation that are related to that file. If you select a recommendation query, you can drop all results for that request. If you select a code recommendation you can drop the items one-by-one. Or, in all cases there, is an option to drop all the recommendations, and this will clear the results view.

By double clicking on a file you can easily navigate to it and request the Eclipse to open it.

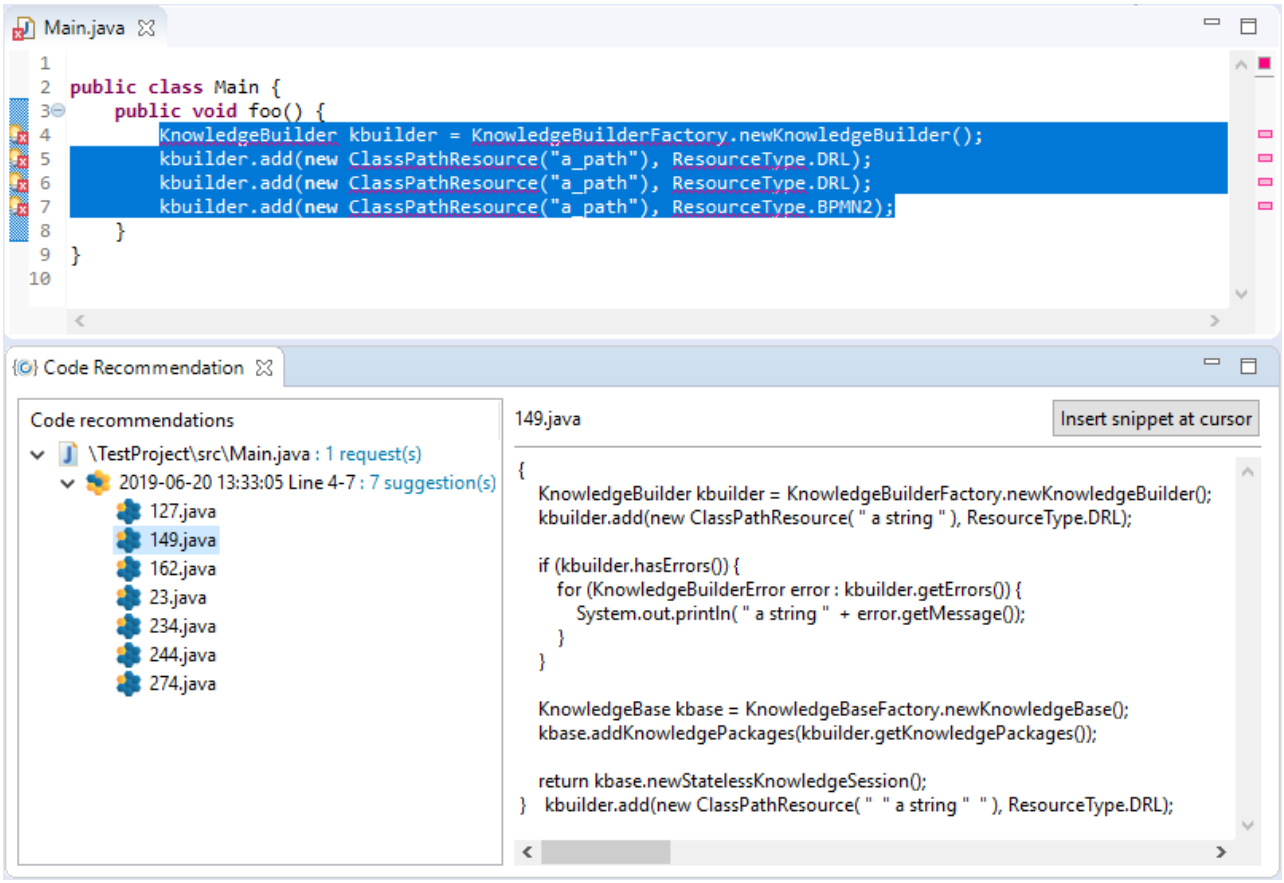

Figure 11: Code recommendation view – preview snippet

3.2.3 Text Based Recommendations

There are tons of documentation and discussion available for various topics, which could be useful for the developers. Those recommendations which yield some natural language documents (or reference to them, e.g. Q&A posts, blogposts, documentation) are called *text based recommendations*.

Developers frequently read different blogs, forums, API documentation, post questions and read discussions about how to solve different problems in different contexts (e.g. with different libraries). As the amount of this online-available information grows, it becomes a greater and greater challange, and takes more and more time to find the relevant sources. This part of the work is automated by the CROSSMINER platform. The goal is to save time for the developers by mining different sources, filtering the information, and providing only the most relevant discussions or documents.

3.2.3.1 API documentation and Q&A posts This feature is used to show you some documentation or forum posts where you may find some useful information about the API calls that you use in your code. To initiate a request for these recommendations, you have to select a code chunk in the active editor and click on the *Request API documentation and Q&A posts* option in the CROSSMINER sub-menu of the context menu. After this, the results are shown in a particular view, where you can examine these results, copy the link for them or by clicking on the title they are opened in

an external browser. This view can be seen on the Figure 12. Note, that the query works for any code chunk, however, the more information (code context) is given to the knowledge base, the more relevant recommendations will be returned.

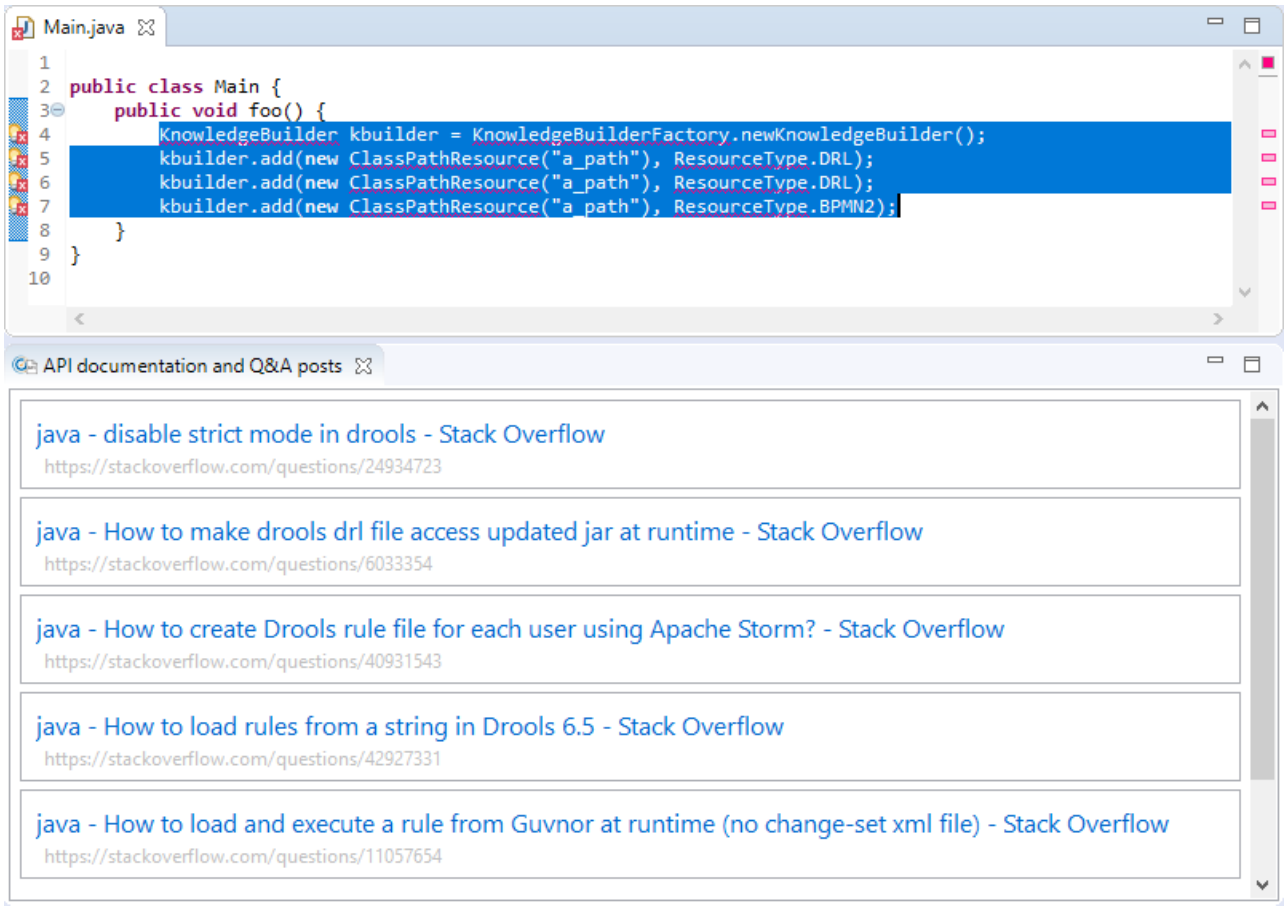

Figure 12: API documentation and Q&A posts view – results

3.2.3.2 API Usage Developers could use these recommendation to get more information about the features and their usage of a 3rd party API, for example offical pages, documentations, and samples.

3.2.3.3 Handling API changes There are several forum threads and change reports, which could ease the migration between different versions of the same library.

3.3 User Activity Monitoring

The main components of this scenario is the recording of developer interactions (events), computing process related metrics and sending them to the CROSSMINER server for further processing. All of the events are stored in a local database and only the metrics are sent to the CROSSMINER server. None of the stored events contain information about the user, so there is no way to identify someone from these data. All data sent to the server are logged on the client side, however, a client-side data browser is not part of the deliverable. The stored database can be investigated by the corresponding database administration tools.

The CROSSMINER server can calculate high-level metrics of the development process, and the *Web-based Dashboard* (deliverable D7.9, developed in task T7.4) is able to show calculated metric values to developers. Users are able to authenticate themselves, after which they will be able to configure their own metrics set, and enable the server to compute the information they are interested in.

3.3.1 List of Collected Events

Document event Our plug-in uses it to detect all keypresses in the editor and store which file is affected. Properties of the event type include *count*, *timestamp*, and *type*. For example, such events are recorded during the implementation of a new method for an existing class.

Part event It stores information about life cycle of given parts (part activating, part deactivating, part closing etc). The event has a *timestamp* and a *type* property, i. e. describing the behaviour of the given part, and is connected to the affected part. For example, this kind of event is recorded when you work on a existing class and select something in the *Package Explorer*.

Window event It stores information about window life cycle the same way as the *Part event* handles information about parts. This event also has *timestamp* and *type* properties. For example, this event is recorded when you open an another Java source code.

Eclipse close event The closes of Eclipse are stored in this even type. As this is the last event when the eclipse is shutting down, we use it as a milestone capturing events between two Eclipse application launches. It has a *timestamp* attribute.

Launch event It stores information about code building and launching. The event has a *timestamp* and contains information about the *type* of launch mode. For example, the event captures if a build was started in debugging, test, or normal mode.

Idle event When there is no new event in the database for a pre-set amount of time, we insert this to indicate the user is not using the IDE. The event has a *timestamp* property.

Project Structure Build event This is a special event that is in charge of mapping the hierarchy of the project. It creates the graph representation of the project structure in the database. It is stored when we open Eclipse or open a project.

Search event It stores information about the user indicated searches in the IDE. The event has a *timestamp* and a *search word* property.

Resource element event It stores information about the file saving and deletion. The event has a *timestamp* and a *type* property and connects to the affected file.

Class path event It stores information about class path changing (entry added to or removed from the class path). The event has *timestamp* and *type* properties. This is recorded, for example, when you add a library to your classpath.

CROSSMINER event These events are recorded when the plug-in is used. It needs for metrics which measure the plug-in usage. It is recorded, for example, when you use CROSSMINER Library Search function.

3.3.2 Categorization of Event Related Components

Event detection requires additional, non-event resources to be stored in connection to the events. These resources are important for the grouping of events and contain extra information for metric calculation. This additional information enables us to examine events in different scales of the project.

File It helps the CROSSMINER Eclipse IDE Plug-in to capture files affected by events and, in later stages of the development, enables the calculation of metrics associated with file usage in the project. This resource stores the *name* of the File.

Part The Parts represent different parts of the Eclipse application. It enables the CROSSMINER Eclipse IDE Plug-in to track the most active parts of the Eclipse application and represents the basis for grouping different types of events. It stores the *title* of the Part.

Window The windows represent the parent Eclipse application window, and it can also be used to group items.

Package The Package will represent the Package, and will be connected to the File, Package or Project items representing the package structure. It stores the *name* of the Package.

Project The projects will represent the projects, and will be connected to the File or Package items representing the representing the project structure. It stores the *identifier* of the Project.

3.3.3 List of Recorded Metrics

To transform the event chain into metric values, we use two similar strategies. Both strategies process subsequent events and compute metric values based on the number or the property values of certain events. The two strategies differ in how the start and end events of the sequence are determined. The first, so-called time-based strategy uses fixed time windows to compute the metrics. In this strategy, all metrics are calculated from an hour slice. We call the second strategy milestone-based strategy. In this case, instead of the time-based windows, we use specific events to split and limit the event chains on which the metric values are calculated. These milestone events are the *Eclipse close event, Launch event* and *Save event*.

Eclipse Search Usage We use it to express the amount of user searches in IDE.

File Access Rate Intend to express the count of Java source file opens and focuses.

GUI Usage Rate Used to express the rate of graphical interface interactions by the user.

Modification Rate Used to express the rate of file modification by the user.

Testing Rate It is used to express the amount of test execution by the user.

Working Time We use it to express the amount of time while the user works on different Java files.

Scava Library Usage The purpose of this metric is to analyse the plug-in provided library-related features.

Scava Search Success This metric helps to analyse the success of the plug-in provided library search function.

Scava Search Usage This metric is to analyse the plug-in provided search function.

	Hour	Eclipse Close	Launch	Save
Eclipse Search Usage	✓	✓		
File Access Rate	✓	✓		
GUI Usage Rate	✓	✓		
Modification Rate	✓	✓	✓	✓
Testing Rate	✓	✓	✓	✓
Working Time	✓	✓	✓	✓
Scava Library Usage	✓	✓		
Scava Search Success	✓	✓		
Scava Search Usage	✓	✓		

Table 1: Metric calculation for different windows.

As Table 1 shows, we calculate different metrics for windows. The reason behind this is that some of our milestone events are project specific but for example the GUI Usage Rate is undefined for project, because the event is related to the IDE not to the current project.

3.4 Settings and Customization

All the plug-in related settings can be found in the preference page of the plug-in. To open this page select the *Preferences* item under the corresponding menu in Eclipse menu bar.

3.4.1 Integration Related Settings

To properly use the plug-in, you have to set some of the settings which is required for integration. As Figure 13 shows you have to enter the Knowledge Base server address, a port and the Web Dashboard's base path.

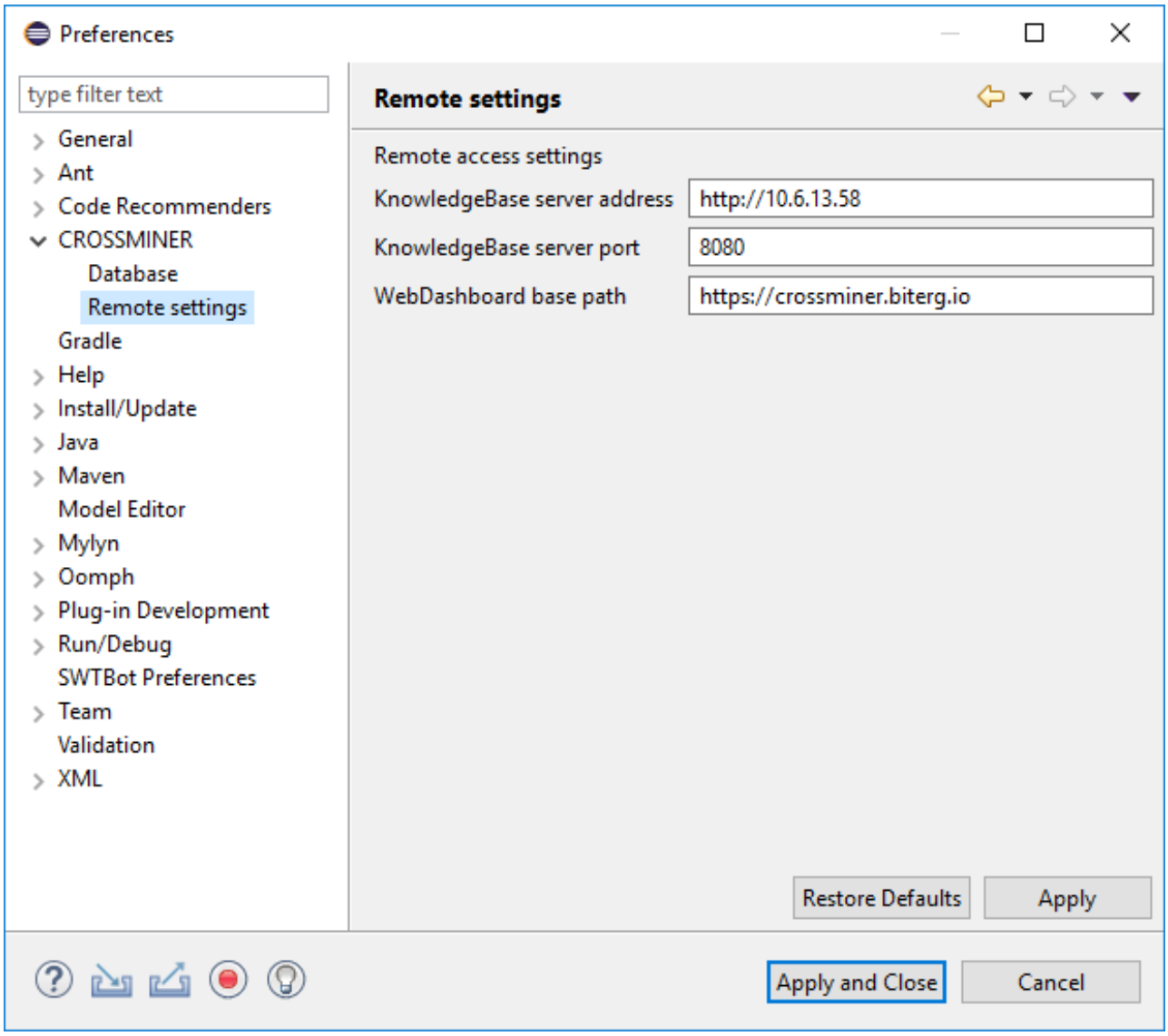

Figure 13: Plug-in remote settings

3.4.2 Process Metric Related Settings

To control which metrics are to be computed or which events are to be collected, navigate to the *User activity events* page under the Preferences' CROSSMINER page. As on the Figure 14 can be seen there are two columns. In the first column there are the events that can be collected by the User Activity Monitoring system and in the second column there are the metrics that can be calculated from the collected events.

Any of these two groups can be enabled or disabled, but it may have some consequences. For example, if the collection of an event is disabled, then no metrics that depend on that event can be calculated, so they will be disabled too. On the other hand, if all the metrics are disabled that would use a specific event, then that event would be useless to collect, so it will be disabled too.

These dependencies can be seen by hovering the mouse over the items in the columns. For example, by hovering over an event, if there are some metrics that would use that event to calculate their values then these metrics would appear over a blue background at there place. This can be seen on Figure 15. The behaviour is similar when the mouse is hovered over a metric, but at this time the events that are used to calculate that metric are highlighted with the blueish background.

To enable the collection of an event, there must be at least one metric enabled that depends on that event.

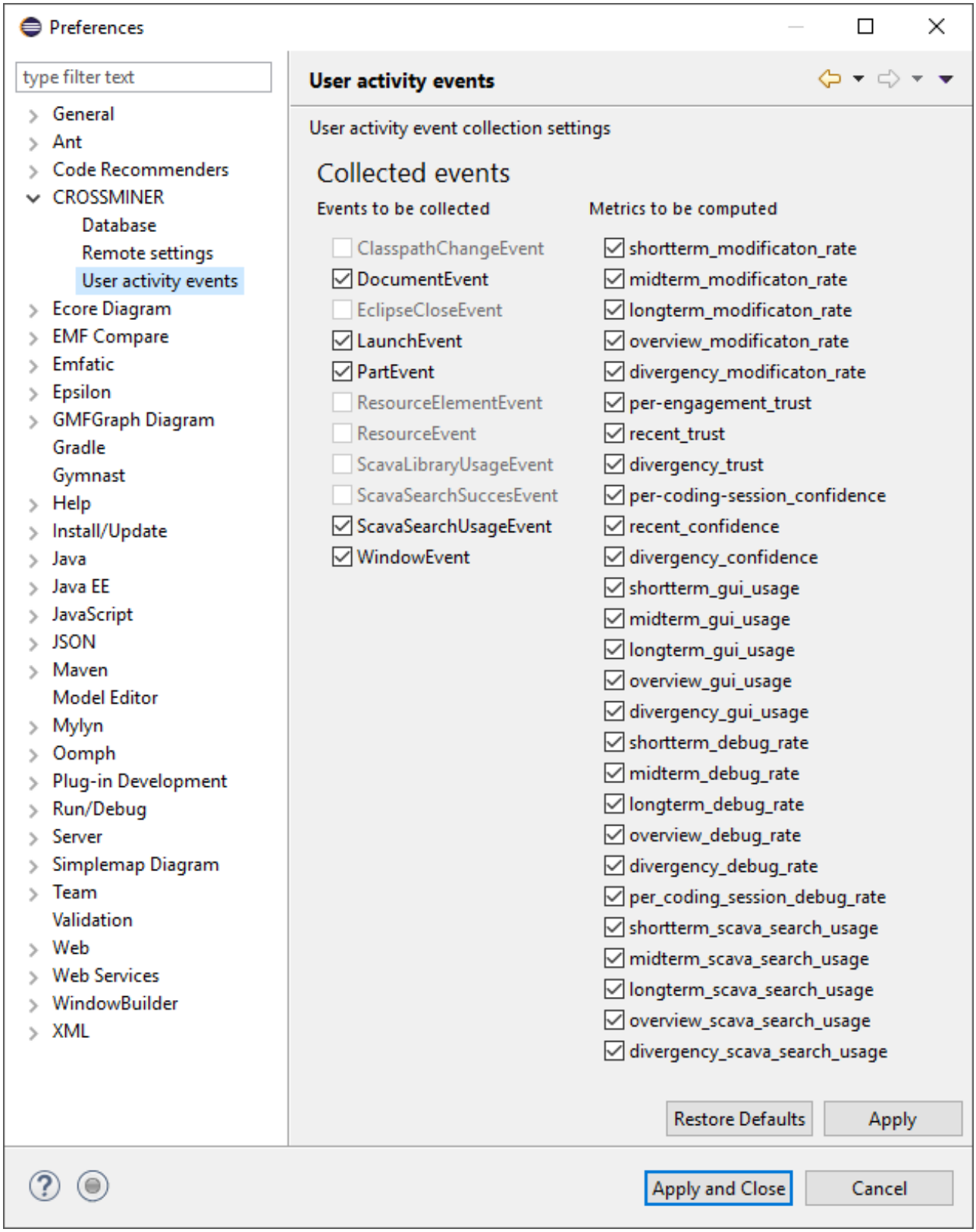

Figure 14: Plug-in remote settings

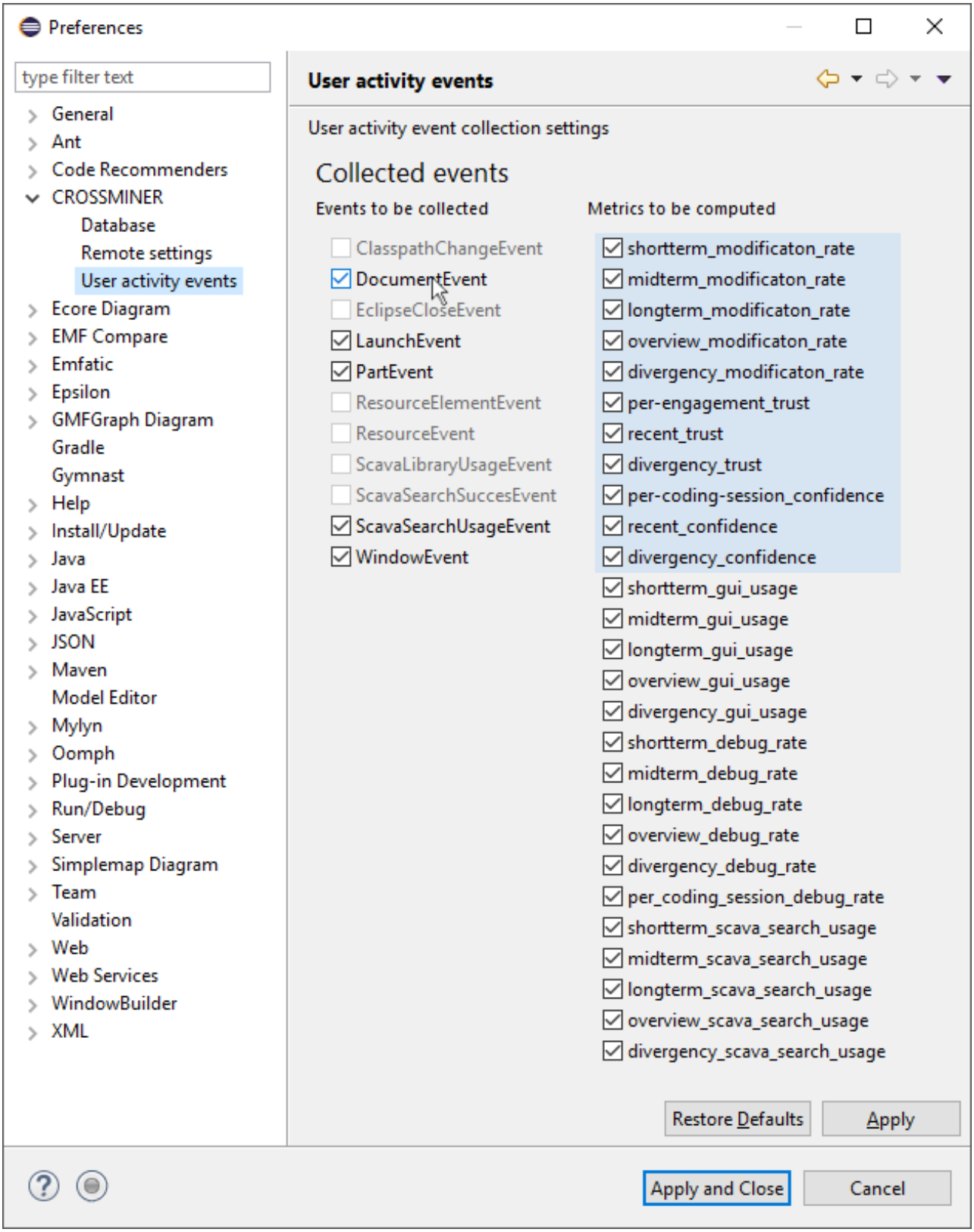

Figure 15: Plug-in remote settings

4 Requirements coverage

In this deliverable, the final version of the IDE plug-in was prepared, and as such it has all required functionality. In the following subsections we show how the technical (Section 4.1) and use case (Section 4.2) requirements related to the CROSSMINER Eclipse IDE Plug-in and defined in the Project Requirements document (deliverable D1.1) are covered by the final version of the plug-in. In the last column of the tables an empty circle (○) denotes that the requirement is minimally (or not) covered, a half-filled circle (◐) denotes that it is partially covered, and a filled circle (●) denotes that it is mostly (or fully) covered. Requirements marked with a cross (×) became invalid for some reasons during the development.

4.1 Technical requirements

D74	The IDE shall provide a settings interface to the user, where the different properties of the CROSSMINER IDE plugin (like server address and port, global settings for recommendation queries, etc.) can be checked and changed. So the user can configure the plugin.	SHALL	●
D75	Recommendation query shall be initiated from the menu, from context menus, and from the toolbar in the IDE. So the user can easily start working with the recommendations.	SHALL	●
D76	The IDE shall use a view to list the code recommendations for the examined project. The view shall provide hierarchical grouping of listed items, filtering by meta-properties, and navigation.	SHALL	●
D77	The IDE shall provide an interface, where recommendations against a new (to-be-used-in-the-project) library can be given. So the user can describe features and functionalities they wants to perform using an external library. The user can also give some constraints, e.g. minimal age or users of the library.	SHALL	●
D78	The IDE shall provide an interface to the user, where libraries that match some recommendations are listed with their selected set of details in filterable and sortable form. So the developer can easily compare the libraries and choose those best fit to their expectations.	SHALL	●
D79	The IDE should provide the ability to the user to mark a library proposed for a query as “not appropriate”. This information can help in the improvement of the knowledge base.	SHOULD	●
D80	The IDE shall provide an interface where known information from a single library is shown. This interface can help the user to check details of the library, and reach out for more information through links (e.g. to project home page).	SHALL	●

D81	The IDE shall provide an interface to the user, where recommendations for replacing a library used in the current project with some alternative libraries can be given. The user can select the library currently used in the project to be replaced and can give further recommendations against the alternatives.	SHALL	●
D82	The IDE shall provide the user the ability to initiate library version check. So the user is notified if some libraries used in their project have new versions.	SHALL	●
D83	The IDE shall perform library version checks on the libraries used in the current project. The check is performed on project load, and only marks the upgradeable libraries. *Instead of marking the libraries, a dialog is initiated.*	SHALL	◐
D84	The IDE shall be able to show if a library used in the current project has a new version that satisfies some pre-determined criteria set globally for library upgrades by the user. So the user can see the relevant results of the library upgrade search. *Performed with library upgrade checks.*	SHALL	◐
D85	The IDE shall provide an interface where the details of an available library upgrade can be shown. These may include the new version number, release date, number of users, number of bugs, estimated impact of the upgrade, etc.	SHALL	●
D86	The IDE should provide the ability to initiate a library upgrade that marks all those places in the source code that needs rework due to the change of the library version.	SHOULD	●
D87	The IDE shall provide an interface that explains the steps of how to upgrade a library used in the project to a new version. *For the handled library types, upgrade is performed autonomously.*	SHALL	◐
D88	The IDE may perform the steps of a library upgrade autonomously (if the user requested the upgrade).	MAY	●
D89	The IDE may provide an interface where description of a feature to be implemented and libraries that should be used to implement it can be given. So the user can ask for recommendations how to implement features using specific libraries.	MAY	○
D90	The IDE may provide an interface to show recommendations on how to implement a feature using some specific libraries.	MAY	○
D91	The IDE may insert a recommendation in the code if the user accepts it and requested it on a code position.	MAY	●
D92	The IDE should provide the ability to the user to select a code snippet or a file and ask for code recommendations for it. So the user can check whether there is a better practice to solve her problem.	SHOULD	●
D93	The IDE should be able to show recommendations assigned with code elements. So the user can see what the knowledge base suggested for a code snippet.	SHOULD	●

D94	The IDE may be able to process recommendations and perform code migration autonomously. So the user can point to a recommendation and the IDE replaces the old code to the new one.	MAY	●

4.2 Use case requirements

U154	Eclipse users can invoke code recommender via an easy shortcut	SHALL	●
U155	Eclipse users can easily deactivate the analysis *This requirement became invalid due to the chosen technology of implementation.*	SHALL	×
U156	Eclipse IDE proposes a dedicated view or perspective for recommendations	SHALL	●
U157	Eclipse IDE proposes filtering options for recommendations	SHALL	●
U158	Eclipse IDE plugin can be installed via the Marketplace	SHALL	◐
U159	Eclipse IDE plugin can be installed via an update site	SHALL	●
U160	CROSSMINER IDE notifies if a new version of a third-party API used by the project on which the developer is working is available *On request.*	SHALL	◐
U161	CROSSMINER IDE notifies if a new version of a third-party API used by the project on which the developer is working breaks backward compatibility	SHALL	●
U162	CROSSMINER IDE is able to offer the use of the newest version of a third-party API utilized in the project	MAY	●
U163	CROSSMINER IDE is able to identify and navigate to those places that became suspicious for changing behaviour after the third-party API version used in the project has changed	SHALL	●
U164	CROSSMINER IDE is able to mark the usage of deprecated third-party APIs in the source code the developer is working on.	SHALL	●
U165	CROSSMINER IDE offers a list of community discussion forums concerning the use of a changed third-party API element	SHOULD	●
U166	CROSSMINER IDE offers code examples for deprecated third-party API usage points	SHOULD	●
U167	CROSSMINER IDE provides the ability for the developer to give feedback on the usefulness of the advises in the given situation	SHALL	●
U168	CROSSMINER IDE is able to notify the developer about third-party API changes that are in design or development phase *Handled in the same way as the API changes of "main" versions, depends on if the server analyses development versions of the API.*	MAY	●
U169	CROSSMINER IDE is able to provide an overview of the impact of a third-party API change on the project the developer is working on	MAY	●

U170	CROSSMINER IDE provides the ability for the developer to ask recommendations for a code chunk previously marked by the CROSSMINER IDE as suspicious	SHOULD	●
U171	CROSSMINER IDE provides the ability for the developer to ask recommendation for an arbitrary code chunk or code element	SHOULD	●
U172	CROSSMINER IDE provides developers code templates and example of codes related to the usage of the API of a specific project	SHALL	●
U173	CROSSMINER IDE assists developers to migrate the current version of a third-party jar to the new version by providing a list of required changes, advises and code templates	SHALL	●
U174	CROSSMINER IDE provides suggested alternatives to the usage of third-party jar which offer the same range of services as a jar used in current project	SHOULD	●
U175	CROSSMINER IDE assists developers to address a deprecated API by proposing an alternative for obtaining the same behaviours of the code	SHOULD	●
U220	User and admin documentation is embedded into the UI	SHALL	●
U225	Plugin supports the latest supported release of Eclipse	SHALL	●