FP7-ICT-2011-8

MARKOS

The MARKet for Open Source

An Intelligent Virtual Open Source Marketplace

WP6 - Front-end and Validation

D6.2.2.a
MARKOS evaluation report

Due date: 31.12.2013	**Delivery Date**: 24.01.2014
Author: Alicja Ciemniewska, PSNC	**Partners contributed**: ALL
Dissemination level: PU	**Nature of the Deliverable:** Report
Internal Reviewers: FRANCESCO TORELLI (ENG), THOMAS GORDON (FOKUS)	

VERSIONING		
VERSION	DATE	NAME, ORGANIZATION
0.1	**25.10.2013**	**ALICJA CIEMNIEWSKA, (PSNC)**
0.2	**04.12.2013**	**ALICJA CIEMNIEWSKA, (PSNC), ILKNUR CHULANI, JAMES AHTES (ATOS), FRANCESCO TORELLI, ROBERTO PRATOLA (ENG), THOMAS GORDON (FOKUS), ANTONELLA PASSANI, LUCA SIMEON, (T6), MASSIMILIANO DI PENTA (UNISANNIO)**
0.3	**30.12.2013**	**ALICJA CIEMNIEWSKA, (PSNC), FEEDBACK CONTRIBUTED FROM ALL PARTNERS**
0.4	**03.01.2014**	**ALICJA CIEMNIEWSKA**
0.9	**17.01.2014**	**ALICJA CIEMNIEWSKA (PSNC), JESUS GORROÑOGOITIA CRUZ (ATOS)**
1.0	**24.01.2014**	**ALICJA CIEMNIEWSKA (PSNC), REVIEWED BY: FRANCESCO TORELLI (ENG), THOMAS GORDON (FOKUS)**

Executive Summary:

The overall goal of the validation task (T6.2) is to conduct a fit-for-purpose evaluation that will assess the MARKOS system in real life scenarios with involvement of FLOSS developers to assess the MARKOS scientific and technical solutions satisfy the needs and expectations of users. Validation refers then to the evaluation of the prototype from users' perspective, and aims at answering the question: "Does the MARKOS system meets users' needs?".

Traditionally most of the testing effort occurs after the requirements have been defined and the coding process has been completed, but in the agile approaches most of the testing effort is on-going. Software evaluation with users should occur at each stage of the software development life cycle, starting when the first requirements are defined. In line with this approach, the first year validation activities started in M6, when the first requirements specification was ready.

This report describes the results of validation actions that were conducted up to end of M15 of the project life cycle. The validation activities summarized in the report can be divided into 3 stages, focusing on validating the MARKOS concept on the basis of different project artefacts i.e.: requirements defined in terms of usage scenarios (M6-M7), early system mock-ups (M9) and first year version of the software prototype (M13-M15).

The report includes results of the analysis of the collected feedback on MARKOS initial requirements and first prototype. The summary delivers not a batch of data to interpret, but a set of 39 actionable findings that the development team can act upon.

It should be mentioned that the synthesizing of needs requires combining ideas from many points of view and a proper balancing between the original vision and the received feedback. Leveraging user feedback to validate the ideas, design and implementation does not mean that all comments raised by the users will be applied. The feedback needs to complement the scope of the project, adding value to an original vision of what the product should look like. Also the way to implement some findings is still under analyses.

D6.2.2.a – MARKOS evaluation report

Table of Contents

1. INTRODUCTION

1.1. Background

The MARKOS project aims to facilitate the software developer's discovery and analysis of open-source software suitable for his/her technical and legal needs.

To this end, MARKOS realizes a prototype of a service and an interactive application providing an integrated view on Open Source projects available on the web, focusing on functional, structural and licensing aspects of software code.

With respect to the offered capabilities, MARKOS aims to support users as follows:

- support search of specific components using expressive query mechanisms based on software dependencies other than usual features such as language, operating system etc.
- facilitate the understanding and comprehension of the software from a technical point of view by a more user-friendly navigation and browsing of the code
- allow a more efficient and accurate analysis of licence compatibility and to provide well-founded legal arguments
- facilitate the collaboration between different projects (upstream/downstream ticketing)

MARKOS wants to offer developers and analysts a solution for choosing the Open Source components that suit their needs, allowing them to learn how to integrate or extend such artefacts and more generally to foster easy adoption of Open Source software.

To fulfil these challenges, the MARKOS project intends to realize the prototype (MARKOS system) of an automatic service providing an integrated view on Open Source software available on the web, focusing on functional, structural and licensing aspects of the software code released by the projects.

The overall technical objectives have been translated into functional requirements, as described in D1.1.1a [1]. The specification includes a set of user requirements organized in themes, epics, features and user stories. During the first year of the project only part of the functionality will be implemented and thus available for validation.

1.2. Deliverable objectives

This document collects the results of the analysis of user feedback. It considers the evaluation performed from the beginning of the MARKOS project until month M15.

The overall goal of the validation task is to conduct a fit-for-purpose evaluation that will assess the MARKOS system in real life scenarios, with the involvement of developers that use FLOSS dependencies (projects, libraries, etc.), to assess whether the MARKOS scientific and technical solutions satisfy the needs and expectations of users. Validation refers to the evaluation of the

prototype from users' perspective, and aims at answering the question: "Does the MARKOS system meets users' needs?". Therefore this tasks focuses on validation with users, checking if the system is appropriate for the purpose it has been designed and meets the business and operational needs; as consequence, the system can be used in real-world scenarios, e.g. by OSS communities.

A common approach to the evaluation has been described in D6.2.1a [1]. That deliverable describes the high-level validation plan and the set of validation activities that will be conducted during the project life-cycle defining the objectives, metrics and methodology to evaluate the fulfilment of users' needs by the MARKOS system.

1.3. Structure of document

The document is divided into four main chapters, including this introduction.

Section 2 provides a summary of the validation activities performed since the beginning of the project up until M15.

Section 3 describes results of the analysis of the feedback provided by users.

Section 4 presents the conclusions of the document.

1.4. Related documents

The interested reader should refer to a document D6.2.1a "MARKOS evaluation: requirements, metrics, methodology and scenarios", where the evaluation plan is presented and D1.1.1a "Definition of storing, browsing and querying requirements" where the MARKOS initial requirements are defined.

2. SUMMARY OF VALIDATION APPROACH AND ACTIVITIES

Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed. In the agile approaches most of the testing effort is on-going. Software evaluation with users should occur at each stage of the software development life-cycle, starting since the first requirements are defined.

The original plan was to start the validation efforts after the release of the first prototype. However, at the beginning of the project development the consortium decided to adapt some techniques coming from the world of agile methodologies. In line with the approach, the first year validation activities started in M6, when the first requirements specification was ready, in parallel to defining the holistic approach for validating the MARKOS service prototype, described in deliverable 6.2.1a, released at month M12.

The activates carried out during the first validation period, i.e. since the beginning of the project up to the M15, included three main validation activities at different stages of MARKOS development, taking as a basis different project deliverables. As soon as the initial usage scenarios were ready, described in D1.1.1a, a survey was conducted in M6-M7. Shortly afterwards, in M9 first front-end mock-ups were created to visualize the requirements and a focus group workshop to discuss them with developers was organized. Starting from mid M13, for the purpose of validation, the project provides an online prototype service where users can use the MARKOS features they need for their everyday working activities. It enabled the pilot assessment to be conducted in M14-M15. Meanwhile, the MARKOS prototype was also showed at the ICT 2013 Conference, which resulted in gathering some additional feedback and demo website traffic data was collected and analyzed.

The prototype was validated according to the holistic evaluation approach defined in D6.2.1a. In the evaluation plan, typical categories of software requirements where used as a basis to specify detailed objectives of user acceptance validation are as follows:

- Functionality: to assess whether the capabilities of MARKOS prototype meet users' needs;
- Information / content: to assess whether the information provided by MARKOS system is of the kind users are looking for;
- Usability / design: to assess whether the prototype meets needs with respect to interaction design and information navigation;
- Performance: to ascertain whether the prototype meets basic performance requirements;
- Benefits**:** to assess the perceived value of MARKOS system and the benefits it provides to users.

Until the release of the first software prototype, the initial two validation activities have not covered all the objectives, focusing mostly on the functionality specified in term of requirements and the expected benefit of the MARKOS.

3. VALIDATION ACTIVITIES RESULTS

Below are summarized the results of the analysis of the collected feedback on MARKOS initial requirements and first prototype. The summary delivers not a batch of data to interpret, but a set of 39 actionable findings that the development team can act upon.

All the findings are described using a common template, presented below:

Finding #No	**<finding summary>**		
Reasoning	<description of finding and data and comments (e.g. from survey or workshop) the finding is based on>		
Reported on	<date>		
Aspect	<aspect/objective which the finding contributes>	**Priority**	<priority (low, normal, high, urgent)>
Assigned To	<role in charge of implementing the finding (e.g. Scenario Leader Component Leader or Work Package Leader)>	**Timing**	*<planned time of implementing the finding> to be filled by assigned role*
Way of implementing	*<Description of how the finding will be addressed, if it will be adopted and implemented> to be filled by assigned role*		

It should be mentioned that the synthesizing of needs requires combining ideas from many points of view and a proper balancing between the original vision and the received feedback. The feedback needs to be analyzed, to consider different options for new features and to decide what is possible to apply and what has to be discarded. Leveraging user feedback to validate the ideas, design and implementation does not mean that all comments raised by the users will be applied. The feedback needs to complement the scope of the project, adding value to an original vision of what the product should look like. In particular, the innovative and research nature of the project implies that some comments, although accurate, cannot be taken into account, for example due to the timing constraints or lack of enough research potential.

Therefore, for each finding, a description of a way of implementing the finding is provided, as a result of discussion on applying the user feedback to the MARKOS development plan. Also it should be highlighted that the way to implement some findings is still under analyses.

3.1. Survey on scenarios

3.1.1. Summary

The survey was conducted in M6-M7 (March-April 2013). This first validation questionnaire was prepared based on the usage scenarios defined in D1.1.1a. The goal was to collect the first feedback on initial requirements defined for the MARKOS system.

This questionnaire was addressed to people involved in software development process and/or using open source components. It consisted of background demographical questions (contact information, organisation, role in a project) and questions related to the 4 main usage scenarios. For each scenario, there were a few specific questions, asking technical details (different for each scenario) and eight general questions (common for all scenarios), related to the overall potential usefulness of such functionality, its impact on the working routine and suggested improvements.

The survey was sent to MARKOS consortium partners targeting people from other projects, not personally involved in MARKOS. We received responses from 13 people.

The results, based on answers collected, have been used to prioritize the implementation tasks for the first prototype.

3.1.2. Findings and Recommendations

The most important finding from the analysis was getting to know how important the specific features of the designed prototype were to the users. The users were asked to point out which 3 steps were most important for them. On the basis of these responses, we created features rankings for each scenario. The features ranked highest were assigned high priority in the development plan. In particular, some features which were initially scheduled to be released in the second version of the MARKOS system were re-scheduled to be provided in the first year prototype. Examples of such features are: ”Restrict the search to implementations that adopt non-reciprocal licenses” or “Find all implementations” because they were of highest priority to users (see Finding #3).

Another common comment requested a very simple design. Several users independently suggested the user interface design to be as simple as possible.

The summary of responses and the analysis are described in detail in Analysis of feedback from survey on usage scenarios. Below a summary of all findings from the first survey is provided, together with the agreed upon way to implement the feedback.

3.1.2.1. Findings for Scenario 1 (software search) functionality:

Finding #1	**Users need very simple user interface and search form**

Reasoning	Most of the users responded they need only simple form-based queries. Also, the comments suggested keeping the user interface very simple and even indicated some design solutions for hiding more advanced options.		
Reported on	M6-M7		
Aspect	Functionality, Usability	**Priority**	High
Assigned To	Scenario 1 Leader	**Timing**	*Y1*
Way of implementing	*We are going to provide a simple and clean search form (Google-like) which can optionally show filters, which are hidden by default, to formulate more precise queries.* *Additionally, we will offer an advanced search form for more complex searches, so that the users will have different views to choose among according to their skills and familiarity with the MARKOS system.*		

Finding #2	**Users would like to be allowed to search (or restrict/filter the search) for all the entity types and the relationship types listed in the survey**		
Reasoning	The software entities that the users are most interested to search for are: APIs, packages, interfaces, classes, and programs/applications. The most interesting relationships between software entities are: implementation, function/method invocation, import, used/using projects and inheritance. However all software entity types and relationship types listed in the survey were selected as interesting to search for by some users.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 1 Leader	**Timing**	*Y1/Y2*
Way of implementing	*Taking into account the finding we are going to focus in Y1 on the entity types: Package, Interface, Class. Additionally, by the end of the first year we are going to support searches for APIs.* *Concerning the programs/applications we will try to introduce this concept in the next release, while for the moment we will provide a similar concept, i.e.*		

	projects. *In the end we are going to provide in the first release a way to navigate relationships like 'implementation' and 'inheritance'. We will add the relationships to navigate 'used/using projects' and maybe even the 'import' one in the second release.*

Finding #3	**The features "find all implementations" and "restrict the search to implementations that adopt non-reciprocal licenses" are of highest priority to users**		
Reasoning	The features "find all implementations" and "restrict the search to implementations that adopt non-reciprocal licenses" were indicated by the users as the most important steps of the usage scenario.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Urgent
Assigned To	Scenario 1 Leader	**Timing**	*Y1*
Way of implementing	*We have put major effort into proving already in the first release the "Restrict the search to implementations that adopt non-reciprocal licenses" and even the "Find all implementations" features, which was initially scheduled to be released in the second version of the MARKOS system.*		

Finding #4	**It would be useful for users to add some metrics or information related to the quality of the software implementation e.g. test coverage**		
Reasoning	Users commented that some information related to quality of the implementation would be useful. These could be implemented for example with metrics related to percentage of test cases covered or by adding some kind of social networking allowing browsing of user comments and using them for ranking of search results.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Low

Assigned To	Scenario 4 Leader	**Timing**	*Partially in Y2*
Way of implementing	*It is not in the goal of the project to collect metrics on the quality of the code (a part from metrics like the number of software releases using another one).* *However, in Y2 we are planning to enable user comments in the form of unstructured annotation. Users may use the comments as a mean for evaluating the quality of the code released by the project.*		

Finding #5	**The capability to make SPARQL queries is not required by users**		
Reasoning	Most of the users (almost 80%) responded they don't need SPARQL queries. Simple form-based queries are enough. There were also some comments that any "strange query language" would potentially not be used.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 1 and Validation Leader	**Timing**	*to be elaborated in Y2*
Way of implementing	*This is related to one of the project objectives described in the DoW (task 1.2), that consists in implementing a LinkedData access point, other than a semantic store. Such an access point is intended to be used by data integrators to develop innovative IDE plug-ins or semantic web tools.* *Usually, a Linked Data access points offer a linked data browser and a SPARQL endpoint.* *This finding potentially has been raised since the users involved in the validation process, until now, do not seem to be interested in the implementation of IDE plug-ins or semantic web tools. The findings should be further elaborated. To get a more reliable understanding of this finding during the next validation session we will try to involve Data Integrators. The target users of the SPARQL interfaces, of course, are not the same as for the MARKOS main frontend, since the SPARQL interface are used only by people skilled in semantic technologies and interested in reusing the data collected by MARKOS.*		

3.1.2.2. Findings for Scenario 2 (browsing/code reuse) functionality:

Finding #6	**User would like to be able to browse the software components at the level of the entire library, down to the level of a method or even source code fragment.**		
Reasoning	All the capabilities suggested in the code reuse scenario-specific questions (i.e. entire library, package, class, class cluster, method or source code fragment) are almost equally important for the end users.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 2 and Front-end Leaders	**Timing**	*Y1/Y2*
Way of implementing	*This capability is already provided in Y1. The only missing part is the ability to view the source code. This will be done in Y2.*		

Finding #7	**The feature for showing all invoked methods and for showing all the components/projects and libraries from which the code originates are of highest priority to users.**		
Reasoning	The features “shows all methods invoked”, “all the components and the projects from which the code originates” and “(shows that) some code is a library” were indicated by the users as the most important steps in this usage scenario.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	High
Assigned To	Scenario 2 Leader	**Timing**	*Y1/Y2*
Way of implementing	*In Y1 we will provide within-project dependencies and (some) inter-project dependencies.* *The accurate analysis of inter-project dependencies and kinds of connectors used (e.g., static linking, dynamic linking, fork, etc.) will be done in Y2.*		

Finding #8	**Looking for similar source code elements in other projects would be a useful feature**		
Reasoning	Looking for similar source code elements in other projects would be of help for about 60% of the interviewees. Users commented that it can be useful not only for finding a similar component with another open source license, but also for giving users some insight into the code's reliability or quality.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 2 Leader	**Timing**	*Y2*
Way of implementing	*This is already foreseen for Y2, however only at the level of granularity of whole source code files (otherwise it won't scale up).*		

Finding #9	**Users want to be able to look for similar source code elements in other programming languages**		
Reasoning	Implementations in other programming languages than originally searched for are interesting for all responders. Most of them potentially would use such code as a starting point and re-implement it in the programming language they need (77%) or use it as an external (native) library and integrate through some kind of a middleware (23%).		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	High
Assigned To	Scenario 2 Leader	**Timing**	*Y2*
Way of implementing	*Such features are planned for Y2. The user interface will provide the possibility of searching APIs independently of the implementation language. However, we won't compare the source code (i.e., the implementation) of artifacts developed with different languages, as it might not make sense.*		

Finding #10	**MARKOS should enable browsing source code and method/class level comments (e.g. Javadoc)**		
Reasoning	The source code and method/class level comments were Indicated as the most important source of information to determine whether a method is suitable to fulfil a given piece of functionality.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	High
Assigned To	Scenario 2 and Front-end Leaders	**Timing**	*Y2*
Way of implementing	*The ability to view the source code will be done in Y2.*		

Finding #11	**Linking some external documentation, if available, to the source code component would be a very useful feature**		
Reasoning	Some external documentation on the component was indicated as a very important source of information.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 2 Leader	**Timing**	*will be analysed in Y2*
Way of implementing	*This finding is currently outside the scope of the project. However, we can consider looking for collaboration with other projects in Y2.* *We are investigating this for other (research) purposes. If successful, we will be able to provide StackOverflow discussion links to source code components. However, very likely this can happen only on-demand, as StackOverlflow limits the number of queries we can ask.*		

3.1.2.3. Findings for Scenario 3 (license verification) functionality:

Finding #12	**It is needed to simplify the licensing usage scenario and properly explain the licensing analysis features for people without legal background**		
Reasoning	As end users of the MARKOS system are mainly software developers and system designers a special focus should be put on providing the legal information in clear and simple way for people without a legal background. Users commented that the scenario is too complex and some steps are not clear (in particular steps: "can create a new profile" and "creates a new legal profile tailored to their view of the governing law "). Most of the users do not consult lawyers before selecting an Open Source license to use with their software.		
Reported on	M6-M7		
Aspect	Functionality, Usability/design	**Priority**	High
Assigned To	Scenario 3 Leader, Validation Leader	**Timing**	*Y1/Y2*
Way of implementing	*Markos provides a simple, fully automatic method for checking license compatibility issues, called the LicenseChecker that has been implemented in Y1. Users can do a more careful, thorough legal analysis using the LicenseAnalyser, but this is entirely optional. Users who do not consult lawyers when choosing licenses should reconsider their practice.* *More attention will be also put on licensing issues and MARKOS licensing features in the next validation activities, and an effort will be made to simplify the user interfaces and make the LicenseAnalyser easier to use by software developers and system designers.*		

Finding #13	**Users need a simplified analysis of license compatibility issues**
Reasoning	The need for a simplified analysis is claimed by 85% of interviewed. They responded such a check would be useful to them, even though a fully automatic, simplified check of licenses compatibility issues cannot take into consideration the specific copyright laws of the country of the users. Users also commented that as software developers it would be difficult to them to construct argument graphs from scratch, however they are interested in performing some default analysis and understanding (possibly changing) its argumentation/parameters.

Reported on	M6-M7		
Aspect	Functionality	**Priority**	High
Assigned To	License Checker Leader	**Timing**	*Y1*
Way of implementing	*The license checker will actually provide such a functionality, i.e., it will* - *Point out license incompatibilities with a simple, textual explanation, and* - *Given a source license and a usage mode, allow the query and browsing to return only compatible components*		

Finding #14	**The features for obtaining an argument graph, notifying about available analysis and defining legal profiles are of highest priority to users.**
Reasoning	The following steps were indicated by the users as the most important in this scenario: - "obtains an argument graph showing dependencies between various legal assumptions and conclusions about license compatibility and other legal issues" - "warning was produced automatically (…) after the PerfectPM software was indexed" Also the feature: "can create a new profile, with the help of the legal office, either "from scratch" or by modifying an existing profile" was assigned a high priority by a 40% of the survey responders. These comment is a bit opposite to the ones of finding #12 and #13. Further investigation lead to a conclusion that such functionality is important for users that consult a lawyer on software license issues.

Reported on	M6-M7		
Aspect	Functionality	**Priority**	High
Assigned To	Scenario 3 Leader	**Timing**	*Y2*
Way of implementing	*These features, like all the features mentioned in the scenario, are all on the agenda to be implemented in Y2. How they are implemented will be explained*		

	in the design deliverables.

3.1.2.4. Findings for Scenario 4 (user comments) functionality:

Finding #15	**All feedback features suggested in the usage scenario are almost equally important to users**		
Reasoning	Users are generally positively open to providing feedback about the quality of information and their experience with project and software component. Users are also willing to check user feedback and change the information about their project if necessary and give feedback to the reviews of MARKOS users concerning their project. All features suggested in the usage scenario (browse and write comments/reviews on a library or a project and see reviewer reputation) are almost equally important to users.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 4 (User Comments) Leader	**Timing**	*Y2*
Way of implementing	*User comments in the form of unstructured annotation will be implemented in Y2. The following functionality is planned for registered users:* - *to submit a comment about a given project* - *to browse the comments provided by other users* - *to evaluate the comments posted by other users* *To make the checking of the comments easily for the users, the idea is to have a ranking system that enables the users to rank the comments which afterwards will be classified according to this ranking.*		

Finding #16	**A tagging system to easily rate and tag code quality would be a useful feature**
Reasoning	The users commented that apart from comments and natural language reviews, a more formal tagging system for describing code "quality" would be useful.

Reported on	M6-M7		
Aspect	Functionality	**Priority**	High
Assigned To	Scenario 4 (User Comments) Leader	**Timing**	*to be reconsidered in Y2*
Way of implementing	*Such features were not planned originally, but since the users seem quite interested in it, we might re-consider them in Y2.*		

Finding #17	**Ranking results based on user ratings and comments would be a useful feature**		
Reasoning	Some users suggested that browsing user comments could be a useful source of information related to the quality of the implementation and could be used for ranking search results.		
Reported on	M6-M7		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 4 (User Comments) Leader and Scenario 1 (Software search) Leader	**Timing**	*to be reconsidered in Y2*
Way of implementing	*A ranking system will be developed allowing the software developers to evaluate the comments which afterwards will be classified according to this ranking. The idea here is to use the comments as a means for evaluating the project metadata. The comments will certainly be a good source of information.* *The features for ranking the search results were not planned originally, however we might re-consider them in Y2.*		

3.1.2.5. Other findings:

Finding #18	**The role of Software Consumer is not clear to users**

Reasoning	Almost all system stakeholders' roles (as defined in D1.1.1a) have been represented by the responders. The users were able to select more than one role and usually did. The only one which were not selected by any responder is the role of Software Consumer. It is quite surprising as in the requirements document it is assumed all other roles are somehow inherited from the Software Consumer, because they use i.e. consume software in different ways. This may indicate that the Software Consumer role or its name is not completely clear and was misunderstood by the users.		
Priority	High		
Reported on	M6-M7		
Aspect	Other	**Priority**	High
Assigned To	Requirements Leader	**Timing**	*Y2*
Way of implementing	*Currently, the Software Consumer is defined as an 'abstract' role that can be made more concrete as one of Developer, Software Architect or Software Integrator.* *Given the misleading interpretation manifested by the final user we will change the definitions of the user roles in the next release of the requirement specification (D1.1.1b)*		

Finding #19	**The survey did not reach project stakeholders such as Legal experts or Data Integrators**		
Reasoning	The two kinds of stakeholders which were not represented, i.e. Data Integrator and Legal Expert, are defined in D1.1.1a. These are not the direct users of the MARKOS service, but other people potentially interested in the project. It is important to involve such users in future validation activities.		
Reported on	M6-M7		
Aspect	Other	**Priority**	High
Assigned To	Validation Leader	**Timing**	*Y2*
Way of implementing	*To get a more reliable information during the next validation session we will try to involve all stakeholders, in particular try to include Data Integrators and Legal Experts in the validation.*		

Finding #20	**All scenarios are perceived as having important impact on the user working routines**		
Reasoning	For the large majority of the respondents, all the scenarios will have an impact on their working routine. This is particularly true for Scenarios 1 and 2, that will simplify the work of the respondents and will allow then to perform their activities in a quicker way. For the majority of respondents also Scenario 3 will simplify the work of MARKOS users. For half of the sample, Scenario 4 will also have an impact in simplifying their work, but will also have an impact on the quality of their work.		
Reported on	M6-M7		
Aspect	Benefit/impact	**Priority**	Normal
Assigned To	Dissemination Leader, Validation Leader	**Timing**	*Y2*
Way of implementing	*We can use this information for the dissemination material, using as "testimonials" the results gathered from the survey. For example "80% of the respondents declare MARKOS will speed up their work and make it easier". However, to make the statistic more trustworthy we need to involve more users in the validation.*		

Finding #21	**All usage scenarios are almost equally important for users**		
Reasoning	All usage scenario impacts the daily routine of most users and will be helpful in work activities. Scenario 1 is perceived as the most useful and is also the one that may generate the highest cost savings. In turn, the license usage scenario is perceived as slightly less often used, but also the scenario with the most potential for saving time.		
Reported on	M6-M7		
Aspect	Functionality, Benefit/impact	**Priority**	Normal
Assigned To	Dissemination Leader	**Timing**	*Y2*
Way of implementing	*Features for all usage scenarios should be implemented in parallel.* *This information will be spread to the users through the different means supporting the dissemination campaign, to attract people in testing/validating*		

	the prototype.

Finding #22	**The time estimates show important savings**		
Reasoning	We asked the respondents to estimate the percentage of time saving produced by the different scenarios. The percentage of time saving is between 36% for Scenario 4 and 48% for Scenario 3.		
Reported on	M6-M7		
Aspect	Benefit/impact	**Priority**	High
Assigned To	Exploitation Leader and Validation Leader	**Timing**	*Y2*
Way of implementing	*The data on time savings, if confirmed by a larger sample, can be used for the development of the impact analysis of MARKOS by describing the cost saved at a company level. This can be helpful in defining business models too.* *Time savings should be an aspect of validation always taken into account and verified with users. Therefore, it is recommended to introduce questions related to time saving in future surveys/workshops in order to get as much feedback as possible on this issue.*		

3.2. Focus group workshop

3.2.1. Summary

In M9 a group interview, conducted as a focus group discussion, had been organized during the third MARKOS plenary meeting in Poznań PSNC.

The goal, similar to the first survey, was to understand users' opinions and feelings about the MARKOS system concept and their preferences to add value to their current development workflow for searching and analysing OSS. However, this time the idea was not to collect individual responses, but collect feedback in an interactive group setting where participants are free to talk with other.

The guide used for conducting the workshop is described in detail in

Appendix B: Guide for focus group workshop.

The system was presented to the users by the MARKOS technical coordinator with a slide show presenting MARKOS features including the early mock-ups of the system.

The workshop involved five external users (software developers) and five representatives of MARKOS project.

Figure 1 Focus group workshop

The warm-up questions focused on current experiences of users on the area of searching and analysing open source software which MARKOS tries to consolidate and improve. The current workflow can be summarized as follows:

- Google is used for OSS solutions and alternatives search (actual Google search engine, not Code Search), as well as other sources such as koders.com or ohloh.com.
- Developers also use social-oriented web channels, such as Stackoverflow or devrates.com ,seeking real-life validation from their peers.
- Developers then integrate the software using existing documentation, looking for original authored source docs, and experience from their peers on social channels.
- Licenses are analyzed by developers with limited legal knowledge, through organization's policy and/or legal channels, or through a variety of web references for known compatibility issues.

3.2.2. Findings and Recommendations

In the workshop the two remarkable findings were a) the great importance of the social aspects and b) the need for an aggregator of OSS related information. The users look for information in many different sources, but one of the most important are the comments from other users which have used a particular software component, to learn from others experience as much as possible. The latter aspect reinforces the intersection of the analytical approach of the MARKOS system

with the wider search-based solutions that exist to some extent (e.g. Ohloh, Blackduck, etc.), although shifted towards peer-based analysis and feedback. For this purpose, they use services like StackOverflow, articles done by other developers on how to use a library, or code snippets showing how a library can be used. Developers share their experiences regarding the simplicity/difficulty to use a particular component, the stability/quality of the software, etc.

As an outcome of the focus group workshop analysis, MARKOS is considering to extend the functionalities regarding the management of user feedback, not only improving the collaboration among projects but also the collaboration among users.

Here is a list of findings from focus group workshop:

Finding #23	**Users would like to use MARKOS to crawl social feedback on projects and be an aggregator of OSS related information (e.g. for peer analysis)**		
Reasoning	The social trend in OSS development is important to consider. Developers rely on StackOverflow and other services for peer community evaluation and recommendations that offers real-life experience and validation. As an aggregator of OSS assets, a MARKOS service could be expected to include this community and feedback content in some form, as well. It would bring together existing community feedback, rather than starting yet another source e.g. by integrating data from existing services such as StackOverflow, devrates.com or YouTube etc. It would be great to find a consolidated set of opinions, feedback and evaluation to get an „objective" view on the software.		
Reported on	M9		
Aspect	Functionality	**Priority**	High
Assigned To	Scenario 4 (User Comments) Leader	**Timing**	*will be re-considered in Y2*
Way of implementing	*It is not one of the MARKOS goals to integrate social assets. On the other side, a first evaluation of StackOverflow integration shows a bunch of challenges to be overcome. In addition to that, in Y2, MARKOS will build its "own" social asset which is the comments database that will be developed as a part of the annotation framework.*		

Finding #24	**Users would like to know extensive authoring and dependency information in search/browsing**

Reasoning	The more information that is pooled from portals, the better. The developer looks for a variety of non-technical info, as well (e.g. author info, size of community, popularity and usage by other (well known) projects etc.), so that "trusted" sources can be tapped in the future. Also links to similar or alternative projects and components would be useful.		
Reported on	M9		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 1 & 2 Leader	**Timing**	*to be analysed in Y2*
Way of implementing	*It's not one of the MARKOS goals to integrate information about the community participating in the software development which is a feature already offered by some complementary platforms like Ohloh.net. Yet, since the users seem quite interested in it we may reconsider referring to the results of those platforms. To be analysed in Y2.*		

Finding #25	**Users would like an easy way to see if their project is compliant legally with other software**		
Reasoning	The ability to see a simple "is my project compliant" validation can be an added-value (and matches the Governance-oriented offerings out there now). Developers often lack the knowledge of license conflicts and can unintentionally break policy and/or introduce liability issues for their organization.		
Reported on	M9		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 3 (License) Leader	**Timing**	*Y1*
Way of implementing	*The LicenseChecker already provided this service in Y1, fully automatically, provided that the project that the user's project is on a public forge which has been crawled and analysed. However, users need to be aware that the LicenseChecker cannot be trusted to provide a correct legal analysis. It is intended only as a heuristic for helping users to decide whether or not conduct a more careful analysis, preferably with the help of a lawyer, using the LicenseAnalyser.*		

Finding #26	**Users would browse via licensing compatibility**		
Reasoning	Users (developers), once identifying which license-related guidelines are compatible with their needs, could benefit from using MARKOS' filtered search and browsing to find compliant software based on license policies, instead of shifting through various incompatible results. A possibility to seek for licensing categories (e.g. reciprocal, Apache community) or organize the license search by guidelines/policies (commercial use), not licenses themselves, would be useful. Also it would be useful to filter out the code without any license.		
Reported on	M9		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 3 Leader	**Timing**	*Some to be implemented in Y1*
Way of implementing	*We are going to include in the first release of the MARKOS system a set of filters to help the users in selecting licenses by categories like academic and reciprocal.* *Additionally we are going to provide a way to retrieve software according to its usage compatibility with a given license template.*		

Finding #27	**Users would benefit from knowing of license updates**		
Reasoning	Related to other findings, developers would benefit from alerts on license updates to software they have already adopted/integrated, to avoid compliance and liability issues.		
Reported on	M9		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 2, Crawler and Front-end Leaders	**Timing**	*Partially in Y2*

Way of implementing	*Downstream notification planned in Y2 includes license updates, though, the flag is not license specific. The project is flagged when metadata changes, including (but not limited to) licenses. Making it license specific entails some issues, e.g.:* *1) adding notification complexity on the front-end so we should ask if effort is reasonable* *2) reliable information on licenses does not come from the crawler but from code analysis so we could end up with unreliable notifications* *The feature will be further analysed and re-considered again in Y2 whether it will be very useful to users to include it or general notification mechanism will be enough.*

Finding #28	**It would be useful to know the test coverage**		
Reasoning	How many projects use unit tests and what is the coverage is a useful information adding to confidence and to improve experience of developers by learning by example. Supporting finding #4		
Reported on	M9		
Aspect	Functionality	**Priority**	Normal
Assigned To	Scenario 2 (browsing/code reuse) Leader	**Timing**	*to be analysed in Y2*
Way of implementing	*Such features were not planned originally, but since the users seem quite interested in it, we might re-consider them in Y2.* *It is possible to extract information about availability of test cases. Further analysis whether we could also provide test coverage is to be decided (and probably unlikely), mainly because of the need to compile and run the application, which would require too much time.*		

Finding #29	**Users would find it useful to have a local implementation of MARKOS (e.g. running in local infrastructure, implemented within IDE, etc.)**

Reasoning	Developers mentioned they could benefit from an Eclipse plug-in with the MARKOS feature set, especially for the integration work. It would be interesting to have an Eclipse plugin so that the user can analyze his own code locally and before publishing it		
Reported on	M9, M14-M15		
Aspect	Functionality	**Priority**	Normal
Assigned To	Front-end Leader	**Timing**	*Post-project*
Way of implementing	*This is considered very relevant, but out of project scope.* *For this purpose MARKOS provides LinkedData access point. Such an access point is intended to be used by data integrators to develop innovative IDE plug-ins or semantic web tools.* *The extensibility of the MARKOS API could be provided as part of future research done using MARKOS as a baseline.*		

3.3. Pilot prototype assessment

3.3.1. Summary

The pilot assessment was conducted in M14-M15 by partner organizations. It was a first validation activity based on a running software prototype.

The pilot assessment goal was to perform an internal preliminary check in a controlled environment of MARKOS consortium partners to ensure the MARKOS prototype is technically ready and sufficiently self-explaining before publishing it to wider audience. For this purpose it was needed to introduce some new users-evaluators from partner organizations to try the prototype and the feedback tools.

Apart from functionality and benefit, the validation approach at this stage included also aspects such as the information/content provided by the prototype, its usability in terms of design and navigation and meeting basic performance requirements.

Each Partner was asked to invite a few colleagues from her/his institution to try the MARKOS prototype and provide feedback. The feedback was provided in direct contact with a person from MARKOS team, through feedback message form/box, or by filling the online questionnaire (https://docs.google.com/forms/d/1aqzZl2KQXEEf-D6Uvv3YBS5YWrsOc7tfYmpiDvIxl-8/viewform).

Altogether, 11 users , with technical, research and business background, took part in the pilot prototype assessment. They provided feedback by email (5), an online survey (4) and interviews (2). The comments have been grouped according to five aspects (see

Appendix C: Pilot assessment comments): functionality, information/content, usability, performance and impact/benefit to users. Finally, for each group based on comments the findings were specified.

3.3.2. Findings and Recommendations

Altogether the pilot assessment resulted in 9 findings. The most important finding is that the MARKOS prototype needs a bit of polishing, especially in terms of improving its content i.e. the knowledge base, but also some features and design elements need to be improved to make it fully ready. All the received issues and drawbacks reported by users were submitted to the issue tracker system and planned to be addressed in nearest sprint.

Due to this fact, the release of the public demo and public field trials was postponed, in order to address the most critical reported issues in features and design, add more "help in action" and also add more projects to the repository.

With respect to the general idea, MARKOS is found a useful tool of interest to users. Among the features, the license compatibility analysis was often mentioned as a very valuable tool.

List of findings from pilot assessment:

Functionality

Finding #30	**The first year prototype require polishing before publishing**		
Reasoning	Pilot users got the impression that not all features work properly e.g., several links e.g. in lower bar didn't functioned and click on method/attribute leads to an empty abstract view		
Reported on	M14-M15		
Aspect	Functionality	**Priority**	Normal
Assigned To	All	**Timing**	*M15*
Way of implementing	*Due to this fact, the run of field trial will be delayed to the next sprint release in order to introduce the improvements in features.* *All the received issues and drawback reported by users submitted to the issue tracker system are planned to be implemented in nearest sprints.*		

Finding #31	**A link to the original source is missing**

Reasoning	Having found the project/API/implementation/library, etc. he/she wanted, then what? There should be some action to allow user to get the code e.g. at least a link to go find it on its forge.		
Reported on	M14-M15		
Aspect	Functionality	**Priority**	Normal
Assigned To	All	**Timing**	*Y2*
Way of implementing	*This feature will be implemented in Y2*		

Finding #32	**A new useful feature would be to add social capabilities to involve legal experts to judge on licenses**		
Reasoning	It could be very useful to have an opinion from legal experts on tricky questions about licences. This could be accomplished by extending social capabilities over simple comments, for example in the form of query and legal answers (and maybe a score). Scores and answers should be provided only by recognized experts (users authorized to provide legal responses should be profiled so)		
Reported on	M14-M15		
Aspect	Functionality	**Priority**	Low
Assigned To	License Leader	**Timing**	*to be analysed in Y2*
Way of implementing	*Will be analysed in Y2, possibly using structured surveys.*		

Finding #33	**A new useful feature would be a general picture about a project and its dependencies**		
Reasoning	It could be nice to have an additional functionality that show briefly (in a class diagram or something like that) the main dependencies between java classes (or the main java classes). In general it would be useful to have a general picture about a platform/project, before going to see more specific things. For example, before using apache.lucene and briefly observed the hierarchy and the main		

	classes responsibilities.		
Reported on	M14-M15		
Aspect	Functionality	**Priority**	Low
Assigned To	All	**Timing**	*to be analysed in Y2*
Way of implementing	*Will be analysed in Y2.*		

Finding #34	**The users need "help in action"**		
Reasoning	Not enough help is embedded directly into the prototype itself. Such a unique app, even if oriented towards a developer and "speaking their language", should focus on being a self-explanatory experience. Especially, it is hard for developers to understand the results of the license compatibility check and analysis tool (e.g. the argumentation “con” was not clear). Apart from that, users pointed several elements that need more explanation to them: e.g. “Find compatible licenses”, "Show only public interfaces (APIs),,. Explanation of the terms in the "Abstract view,,. A common comment from users was to include tooltips.		
Reported on	M14-M15		
Aspect	Functionality	**Priority**	High
Assigned To	All	**Timing**	*Y2*
Way of implementing	*To provide more help in action, we will design and implement of a user manual integrated within the MARKOS frontend. For each page there will be a collapsible help box changing instructional text depending on the page content.* *Additionally, tooltips will be added to any button and link in the UI. This functionality is planned in Y2.*		

Information and content

Finding #35	**The users need access to more projects, of different nature, to be added to**

	the repository in order to recognize the potential of such service.		
Reasoning	The knowledge base available at the end of first year (136 Java projects, mostly from Apache foundation) was not representative enough. The limited stack of data made it difficult for the users to conceptualize and try some of the features (e.g. the code view did not displayed the content, difficult to analyze license compatibility, if most of listed projects use the Apache 2.0 license). The knowledge base need to include at least several hundred projects of various nature (e.g. different licenses).		
Reported on	M14-M15		
Aspect	Information	**Priority**	High
Assigned To	Code Analyser	**Timing**	*Y2*
Way of implementing	*Due to this fact, the run of field trial will be delayed to the next sprint release in order to add more projects to the repository.* *It is planned to provide several hundred projects from SF.net in the nearest Sprints of Y2.*		

Usability

Finding #36	**Need for improvements in interface design, several quirks found**		
Reasoning	Users reported several quirks in the design and navigation and suggested some improvements. Reported issues: feedback link hard to see and survey difficult to get (very unlikely to be completed); click through to methods in content explorer refreshes position. Suggested improvements: keep visible filters in "simple search" mode; group results based on selected filters (by tabs or trunks), unify look and feel of Carneades components with MARKOS style. Legal issues should be made more understandable, since the users, being developers, do not want to go into deep details.		
Reported on	M14-M15		
Aspect	Usability/design	**Priority**	High

Assigned To	All	**Timing**	*Y2*
Way of implementing	*The most irritating quirks, the user complained about will be implemented in the nearest sprint, before publishing the prototype.* *There will be continuous work in Y2 on making the interface more user friendly. For example the design of the License Analysis tool will be unified with MARKOS layout.*		

Performance

Finding #37	**The performance of the prototype is acceptable**		
Reasoning	In the current configuration setup, the performance is acceptable. Sometimes the service is a bit slow, however, it is actually a good demo. Only little experience, no problems for now.		
Reported on	M14-M15		
Aspect	Performance	**Priority**	Normal
Assigned To	*N/A*	**Timing**	*N/A*
Way of implementing	*No action needed.*		

Impact / benefit the MARKOS will provide

Finding #38	**The users find MARKOS an interesting and useful tool**
Reasoning	Users pointed out different features (searching, filtering, license analysis) to be of interest to them. The survey responders unanimously agreed that MARKOS provides added-value to their current workflow to discover, analyse and integrate OSS, help searching of software components and facilitate the understanding of software. Most also agreed that MARKOS provides a more efficient and accurate analysis of licenses than the legal tools currently available and provide enough information to make decisions on whether to use a particular open source software component. Among the MARKOS features, license compatibility analysis received much interest and is often foreseen to be the most valuable feature. Users generally agree that it helps in searching OSS

	and to understand a project/platform.		
Reported on	M14-M15		
Aspect	Functionality, Benefit	**Priority**	Normal
Assigned To	*N/A*	**Timing**	*N/A*
Way of implementing	*No action needed.*		

3.4. Interviews on first prototype on ICT

3.4.1. Summary

In M14, the MARKOS project was present at the ICT 2013 Conference within two different events:

- The Exhibition Booth "Open Source Software: Peeking Backstage".
- The “Get Together Booth” "Future services to facilitate FLOSS development and adoption by EU research and business communities"

Both events have been organized in collaboration with other EU Funded Projects: OSSMETER, RISCOSS and PROSE.

The Exhibition Desk, in particular, has been an opportunity to showcase the first MARKOS Prototype to the European research community and collect first feedback on the proposed service over the three full days of the conference.

The features presented could be summarized as the following: filtering on the kind of software entities (classes, methods, packages, libraries), their relationships (inheritance, inclusion, linking), their license models (reciprocal or not reciprocal, compatibility with specific license), and showing results at semantic level (different copies of the same entity replicated in different projects are identified and shown once with their attributes).

Especially the first day, several people passed by and asked for information about MARKOS and the prototype. All of them were particularly interested in the license analysis part and not so much in the feature regarding the code querying/browsing.

Representatives from Academia, Public Administrations, Industry and Policy Makers attended the event and a sample from all the categories passed by the MARKOS exhibition desk. It is worth mentioning, anyhow, that the sample showing more interest into the joint exhibition desk

and in particular in MARKOS has been the one from Industry and Small and Medium Enterprises. This consideration, again, highlights how the need for tools and services providing an integrated view of FLOSS projects is felt of paramount importance in FLOSS adoption and integration within enterprises and how MARKOS can support them in concrete.

3.4.2. Findings and Recommendations

Concerning the collected feedbacks, among the things discussed about with the people, the only interesting feedback is about those topic we are well aware of:

- It would be interesting to have an Eclipse plug-in so that the user can analyze his own code locally before publishing it, which consolidates and complements the finding F#29: Users would find it useful to have a local implementation of MARKOS
- Legal issues should be made more understandable, since the users, being developers, do not want to go into deep details. This consolidates the findings: F#13 Users need a simplified analysis of license compatibility issues and complement F#36: Need for improvements in interface design, several quirks found

3.5. Prototype website traffic analysis

3.5.1. Summary

At the end of M13, the first MARKOS demo has been released and launched internally to the consortium (but not to the public). Since then, a website traffic analytics software (Google Analytics) was setup, in order to get some insight on the usage of the system. The tracking code was included in the source code of the GWT part of the interface.

The website analytics tool allowed us to collect web traffic information during the M14-M15 dissemination and validation activities, including the pilot assessment and demo of MARKOS at ICT 2013 Conference. Metrics collected over the two months has been summarized below:

- There was 186 visits from 62 unique visitors during the given period.
- An average time spent using the MARKOS service was over 6 minutes
- There was 4,371 page views of all pages, among them 72% page views of the MARKOS homepage, 319 page views related to searching and 825 page views related to browsing software components (the tracking code was not included in the license analysis part, therefore no statistics were gathered on the usage of this component)
- Almost one-third of the searches was done using the “exact match” option enabled and over 40% of the searches were performed with applied filters, among them for Keywords (2% of all searches); licenses (10%), and entity types (42%).
- Among entity types most commonly applied filters were: project (24%), API (21%), API implementation (19%), Class (15%), Interface (12%), Library (6%), Annotation and Constructor (3%). Enumeration, Exception, Method, Package filters were not used.

- Users searched mainly for terms provided as examples in the tutorials: onlinetour, log4j, commons, javax.swing.event. Other searches included terms such as: utils, sql, apache, File, java, git, log, proxy, xerces, "expert system", class fourier transformation, ERP, jdyna, image.

- Users viewed 118 software projects, among the most often viewed were e.g.: onlinetour (28%), apache accumulo (20%), java sdk (13%), apache ant (4%)

- There was 90 page views of libraries: htmlwolf-1.1 (13%), apache lucene core (12%), cdc-agui-swing-layout (8%) and 184 page views for API: java (41%)

- There was 51 page views of the advanced search page, and 16 queries executed from this page. Average time spent on the page was below 30 s.

3.5.2. Findings and Recommendations

Although the prototype was not officially published during this period, the metrics show that the MARKOS was broadly used and well validated through MARKOS Partner organizations.

The examples of the collected metrics provided above, show that metrics from a web analysis tool may provide valuable data on the usage of particular features. However, the controlled nature of the pilot assessment and small sample of users disallows to use the results to make general statements on using MARKOS.

Due to the fact that the license analysis part is not implemented using the GWT framework, the tracking code was not included in that part. The mechanisms and configuration of the web analytics tool based on page URLs is not enough to track and measure the metrics related to license features. The following finding was defined:

Finding #39	**Web analysis tool need more advanced setup and configuration**		
Reasoning	The examples of the metrics from the web analysis tool provide valuable data on the usage of particular features. Due to the fact that the license analysis part is not implemented in GWT framework, the tracking code was not included in that part. The mechanisms and configuration of the web analytics tool based on page URLs is not enough to track and measure the metrics related to license features. A more advanced configuration of the web analysis tool is needed.		
Reported on	M14-M15		
Aspect	Other	**Priority**	Normal

Assigned To	Validation leader, License Component Leader.	**Timing**	*Y2*
Way of implementing	*The web analysis tool will configured to allow tracking in order to track and record user interaction with particular UI elements. For this purpose EventTracking code will be added to MARKOS site.*		

4. CONCLUSION

In summary, during the first fifteenth months of the MARKOS projects we conducted several validation activities on different early stages of the project lifecycle. The activities included" a survey on the initial usage scenario, a focus group workshop on requirements and mock-ups and a pilot assessment of the first year MARKOS software prototype.

Altogether almost 40 people, mostly with a technical background, have provided direct feedback to MARKOS requirements and prototype. The group consists of people from the MARKOS Partners organizations, but not personally involved in the MARKOS project, people from organizations in close relationship with MARKOS Partners (for example input to validation was received from Bitergia, which is going to join the MARKOS consortium) or completely external users met and interviewed during dissemination activities.

On the basis of the users comments 39 actionable findings has been defined.

In this report summarizing the validation task up to M15, most of the findings specified based on the collected feedback considers the functionality of the MARKOS system. These findings can be used to direct the further development of the MARKOS prototype. Since the very beginning we also try to include in the validation mechanisms to identify and measure the benefits the MARKOS can provide to users. These will be evaluated even more in the second validation phase.

The first survey on usage scenarios resulted in specifying the priorities for implementing requirements. Rankings for each scenario feature have been created and the features ranked highest where assigned high priority in the development plan. In particular, feature which was initially scheduled to be released in the second version of the MARKOS system (e.g. "Restrict the search to implementations that adopt non-reciprocal licenses" and "Find all implementations") have been rescheduled to be provided in the first year prototype.

The focus group workshop resulted in two remarkable findings "great importance of social aspects" and "the need for an aggregator of OSS related information". As an outcome of the focus group workshop analysis, the MARKOS team has been considering and analysing the possibility to extend the functionalities regarding the management of user feedback including for example some links to the related social assets. The final decision has not been made yet.

The most important finding from the pilot assessment of the first year MARKOS prototype is that before publishing the prototype openly to users it needs a bit of polishing, especially in terms of improving its content i.e. the knowledge base, by adding more projects to the repository, but also some features and design elements, including adding more help. Apart from the direct comments from users, the demo website traffic analysis tool provides metrics showing that the MARKOS was broadly used during the pilot assessment.

With respect to the general idea, MARKOS has been found to be a useful tool of interest to users. Among the features, license compatibility analysis was often said to be the most value-added tool.

Below is a summary of validation results according to specific objectives:

Objective	Quantitative data and metrics	Qualitative description of results
O.1 Functionality To assess if the capabilities of MARKOS prototype meet users' needs	**Metrics from first survey:** - Number of requests to change or remove a feature: 11 **Metrics from pilot assessment:** - Number of defects reported: 5 - Number of enhancements reported: 11	The users have not complained about the functionality. Some of the advanced options such as advanced search, the SPARQL interface or construction of argument graphs are not of direct interest for the majority of users, being developers. However, these features are addressed to advanced users, such as developers of IDE plug-ins and semantic applications (in the case of SPARQL) and lawyers (in the case of argument graph), or interesting for research projects. In particular user's requests for social features allowing to share experience and opinions on software components quality, discuss quality and license issues.

O.2 Information To assess if the information provided by MARKOS system is the kind the users look for	**Metrics from pilot assessment:** - Users subjective level of the perceived usefulness of information in selecting OSS: 3,8 (in a scale 1 to 5)	This aspect was not validated in the first survey and focus group workshop, as the repository did not yet exist. During the pilot assessment of the MARKOS prototype, users complained about the limited number of projects and information available in the MARKOS repository and that the amount of information was not enough to recognize the potential of the service. This was one of the most important reasons to postpone the public launch of the prototype and public field trial.
O.3. Usability To assess the prototype meets needs with respect to design and navigation	Metrics: - Usability problems reported: 2 - Number of improvements suggested: 3	At this point, in a very early prototype stage, we did not focus on the usability aspect. However, the comments related to this part received from users have been taken into consideration. Users made a few comments about design and navigation, in general opting for simplicity of the design. Based on the comments, two defects and three improvements have been identified and reported after the pilot assessment of prototype. A usability focused workshop is planned in the sixth quarter of the project.
O.4 Performance To ascertain the prototype meets basic performance requirements	Metrics: - Performance problems reported: 0	No problems have been reported during the pilot assessment. There was only one comment from one user that the prototype is sometimes a bit slow.

O.5.Benefit the MARKOS will provide
 	
To assess the perceived value of MARKOS system and the benefits it provides to users)
	Metrics from first survey:
	
	- User perceived helpfulness in relation to current work activities: (, in a scale of 1 to 6)
	  o Scenario1: 4,38
	  o Scenario2: 4,54
	  o Scenario3: 3,85
	  o Scenario4: 4,15
	  o Avg. for all usage scenarios: 4,23
	- Estimated percentage of time saving:
	  o Scenario1: 44%
	  o Scenario2: 45%
	  o Scenario3: 48%
	  o Scenario4: 36%
	
	Metrics from pilot assessment:
	
	- Users subjective level of usefulness w.r.t. to MARKOS capabilities and attributes: 3,95 (avg. in a scale of 1 to 5)
		Users expressed very slight differences in judging the usefulness of MARKOS functions. All usage scenarios are almost equally important for users (ranked above 4 in a scale 1-6 in the first survey) and useful. Also during the pilot assessment there were no big difference in the perceived usefulness of the different MARKOS capabilities.
		
		The most controversial is license analysis, by some users almost ignored, by others indicated as most value-added part.
		
		The benefits users see from using the MARKOS system are: simplification of the activities, time saving, but also increase in quality of work and knowledge.

5. REFERENCES

[1] D1.1.1a "Definition of storing, browsing and querying requirements", MARKOS project, 2013

[2] D6.2.1a "MARKOS evaluation: requirements, metrics, methodology and scenarios", MARKOS project, 2013

APPENDIX A: ANALYSIS OF FEEDBACK FROM SURVEY ON USAGE SCENARIOS

Below are summarized the findings of the first survey collecting feedback on MARKOS initial requirements. The survey was responded to by 13 people from the MARKOS partners organisations, but not personally involved in the MARKOS project.

The summary delivers not only a batch of data to interpret, but also a set of actionable findings that the development team can act on. Firstly the results of the scenario specific questions are summarized for each scenario. Then, the general questions are summarized and compared for all scenarios. User comments are quoted in italics.

Usage Scenario 1: Software search description

Antonio is a software architect. He has just designed a new Knowledge Management System and needs a triplestore to implement the persistence layer of his system. He wonders if there is an open source component that implements a standard interface such as Sesame or Jena. Antonio tries to use Ohloh.code, a well-known code search web service, to find open source components implementing the Sesame API (Java package org.openrdf). He writes the name of the API (i.e. the package name) and hits "Search". He finds 5.652.359 results! A lot of choices? No, a lot of noise! Most results refer to the same component and most components just use the Sesame API but do not implement it.

Francesco, one of Antonio's colleagues, suggests trying MARKOS, a new web service for open source development. Antonio accesses to the MARKOS service as a GUEST, (1)[writes the name of the API (i.e. the package name), selects exact match], (2)[Java language] and (3)[API] as constraints, and then click on "Search". MARKOS finds a few APIs containing the specified name "org.openrdf". Antonio (4)[looks at them] and (5)[selects the only one coming from] the Project called "Sesame".

Antonio clicks on (6)["find all implementations"] and (7)[browses the first result]. He immediately (8)[sees] that only few interfaces of Sesame are implemented by that result. Antonio (9)[asks to order the results by percentage of implemented interfaces] and (10)[to restrict the search to implementations that adopt non-reciprocal licenses].

Antonio finds few software products that implement almost 100% of the Sesame API. Now he is interested to understand which is the most adopted product in other Open Source projects. Antonio asks to (11)[order the results by "number of projects which use this implementation"]. Antonio finds an implementation used by several projects. He (12)[looks at the projects using it] and notes he knows some of them. Antonio decides to include in his architecture the implementation used by the projects he knows.

Analysis of feedback on Usage Scenario 1: Software search functionality

For most of the users (77%), to search for software entities, there is no need to allow SPARQL queries, as the simple form based queries are enough.

The software entities that the users are most interested to search are: API, Package, Interface, Class, Program/Application. These were selected by more than half of the responders. However, each mentioned software entity type was chosen at least once. See Figure 2.

The relationships between software entities that the users are most interested in are: implementation, function/method invocation, import, used/using projects and inheritance. The least interesting relationships between entities are part-of/nesting and revision history. See Figure 3.

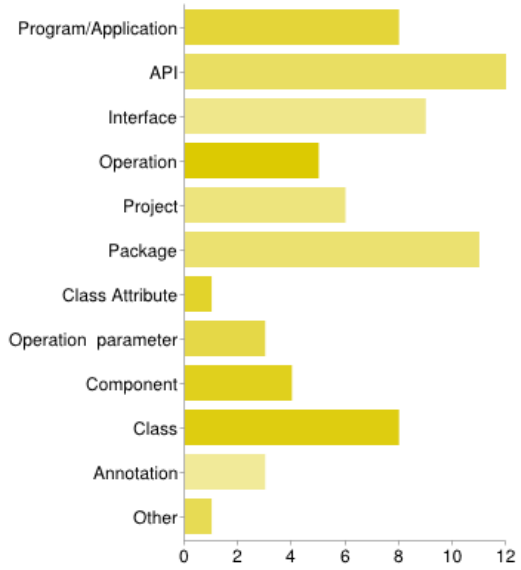


Figure 2 Importance of entities type

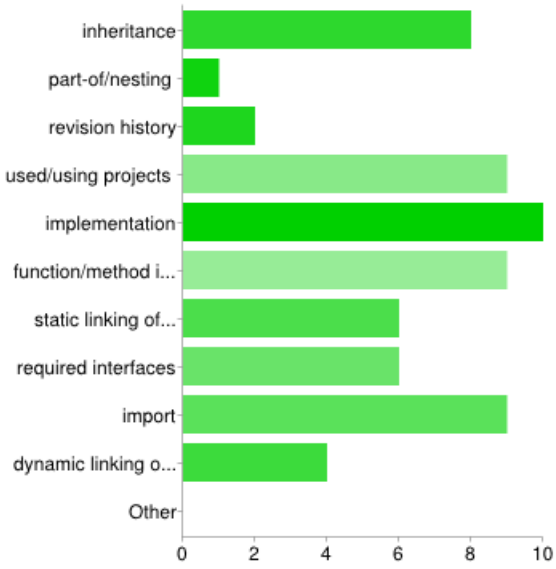


Figure 3 Importance of relationship types

The users were asked to indicate the three most important steps. To summarize the importance, a ranking was created based on all responses, starting from most important step. The top three steps from the list are:

- Step 6. “find all implementations” (selected 8x)
- Step 10. “restrict the search to implementations that adopt non-reciprocal licenses” (8x)
- Step 7. “browses the first result” (5x)

For clarity, we provide here the full step description only for the top three steps from the list. Further order is as follows: 2, 11, 1, 9, 4, 5, 8, 12, 3.

Also users could point to steps that should be removed or changed. As the question was optional all responses are included in the summary. The users suggested to:

- Remove step 2 (“select Java language as constraint”) and step 3 (“select API as constraints”) as “*from a user perspective the system should know that "org.openrdf" is an API written in Java”.*
- Change step 1 (“writes the name of the API (i.e. the package name), selects exact match”) because “*Sometimes a developers do not have any API name in mind.. she just writes a natural language query”.*
- Change step 9 (“asks to order the results by percentage of implemented interfaces”) for the reason that *“for standard APIs, it's also important to know % test cases covered”.*

Other improvements suggested by users include ideas such as: making the search interface as simple as possible with only one input (like in Google search engine), with a possibility to add further restrictions after the first search; suggesting the user with further restrictions, (*“e.g. search engine could guess the language of required library”*); avoid *“strange query language”;* hiding advanced search options under some "Advance search options..." button; adding some kind of social networking for ranking of search results and prioritizing the results that were well ranked by other users or the ones used by the MARKOS user in the past; filtering the results by the number of known open issues; showing project statistics about release dates to see if the project community is active; and allowing browsing of user comments related to quality of the implementation.

There are also questions that indicate that the scenario text may not be entirely clear:

- *Will everyone know what a "triplestore" is?*
- *What's the difference between an "API" and an "interface" - i.e. why does the org.openrdf "API" have a "% implemented interfaces" property (the "I" in API stands for "interface")*

Some of the suggested improvements, e.g. the social networking features, are already addressed by other usage scenarios. The ones related to the information at the project level,

such as filtering by the number of open issues and showing project statistics, or requiring research in natural language queries are out of the scope of the MARKOS goals.

The comment related to removing step 2 (“select Java language as constraint”) and step 3 (“select API as constraints”) as “*from a user perspective the system should know that "org.openrdf" is an API written in Java”.*

Usage scenario 2 (browsing/code reuse) description

Roberto is a Software Developer, he is contributing to BeSmart, a BI open source project, and he has to add a feature to import xls files.

He knows that the same feature is offered by an OS component from project JData. He would like to reuse the corresponding code, but avoiding to import unrelated libraries and unneeded features. He needs to understand the dependencies of the method he is interested to. But he cannot see at the dependencies by browsing the code on the web.

The analysis of dependencies requires to import the full JData code in his preferred IDE. Also the projects that JData depends on must be imported (as the code Roberto needs could be imported/inherited by them). Roberto has no much time and he does not know how long does it take to do all that.

Roberto consults Massimiliano, an expert in software analysis, that suggests to try MARKOS, a new web-based environment that allows to navigate the dependencies in open source projects at a global level.

Roberto (1)[browses on MARKOS the project JData] to (2)[find the method] he is interested to. He notices that, differently from existing code search engines, MARKOS allows to highlight and browse components, interfaces and dependencies, instead than working at granularity of source code files and directories.

Once identified the wanted method xlsImport, Roberto clicks on “find dependencies”. MARKOS (3)[shows him all the other methods invoked] and (4)[the components used] by xlsImport, and (5)[all the components and the projects from which the code originates]. Roberto notices that MARKOS has traversed several projects6) and that the needed code was not in the projects directly referred by JData, but in some of their ancestors.

In some case MARKOS (7)[presents different guesses on code provenance] , because a source code element is similar to other source code elements found in different other projects, and Roberto has to decide which is more reliable, but it is much better than analyse dependencies without any help.

Roberto notices that (8)[some code is a library] that contains also other interesting features, so he decides to reuse the full library (and all other libraries he depends on). Some other code is placed in libraries that are less useful, so he decides to copy and reuse just the (9)[classes] he needs.

Analysis of feedback on Usage scenario 2 (browsing/code reuse) functionality

The level of granularity of performing reuse (i.e. entire library, package, class, class cluster, method or source code fragment) do not vary very much. All responses are chosen almost equally often. See Figure 4.

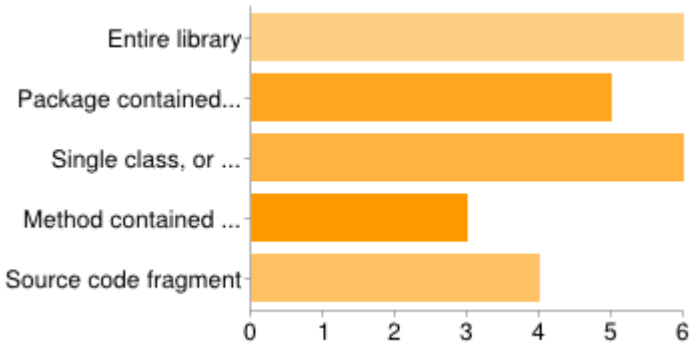


Figure 4 Level of reuse granularity

Looking for the same or similar source code elements in other projects would be of help for about 60% of the interviewees. Most of them find it helpful because they can find equivalent code released under different licenses. In turn implementations in other programming languages than originally searched for are interesting for all responders.

With respect the premier sources of information to determine whether a method is suitable to fulfil a given piece of functionality all sources are indicated as important (most have the average note higher than 4,5 in the scale 1 to 6). The ranking is the following:

- Source code,
- Method/Class level comments (e.g. Javadoc)
- Any external documentation if available, (all above with notes higher than 4.5)
- Class definition and method signatures
- Inline comments.

The top three most important features of the MARKOS system in this scenario are:

- Step 3. “shows him all the other methods invoked” (selected 8x)
- Step 5. shows “all the components and the projects from which the code originates” (7x)
- Step 8. shows that “some code is a library” (6x)

Further order is the following: 4, 2, 6, 7, 1, 9.

The only step that should be changed is step 7 regarding the guesses on code provenance, because it is not clear for the users. The comment was: “*unclear what is intended*”.

Additional comments on suggested improvements are:

- *It would be useful to link somehow the search results with some code samples.*

- *Knowing which projects use code is also useful because it can give some clue as to the code's reliability/quality*

Usage Scenario 3 (license verification) description:

Silke is a software architect responsible of the licence quality of OpenIT.net, a software house specialised in the development of open source software. She needs to check if the PerfectPM software is violating any licence.

Usually Silke works with the technical lead and the legal office of her company to evaluate the legal risks, but the PerfectPM software has become so large, with many dependencies on other components, that it is becoming increasingly difficult for them to evaluate the risks comprehensively. After a while she learned about MARKOS from a blog.

Silke (1)[searches for "PerfectPM"] on MARKOS. Silke finds the PerfectPM software on Markos and goes to the "licence analysis" section. Silke notices the warning on the license analysis page, suggesting that the EPL license used by the PerfectPM may be incompatible with the GPL license of the software entities used by PerfectPM. This (2)[warning was produced automatically by the LicenseChecker of the Markos system after the PerfectPM software was indexed by the CodeAnalyser of Markos].

Silke wants to analyse the license compatibility issues more thoroughly so she (3)[clicks the "Analyse" button.] In an interactive dialogue with Markos, Silke enters further facts about the PerfectPM software, if necessary, and (4)[obtains an argument graph showing dependencies between various legal assumptions and conclusions about license compatibility and other legal issues.]. Silke can (5)[select a legal profile containing particular assumptions about the governing copyright law to have the argument graph evaluated with these assumptions]. If Silke is not able to find a suitable legal profile for her jurisdiction, she (6)[can create a new profile, with the help of the legal office, either "from scratch" or by modifying an existing profile].

Silke's analysis of PerfectPM has helped to clarify the potential legal risk. The PerfectPM software uses a EPL license, but is dynamically linked to an "EffortManager" component that uses a GPL licence. The EPL license is not compatible with the GPL licence. This might not be a problem if dynamic linking does not cause the PerfektPM software to be derived from the EffortManager, but Silke has selected a legal profile which follows the legal opinion of the Free Software Foundation, which does consider linking to create derivative works.

(7)[A link to the argument graph can be published and stored in the MARKOS repository, along with metadata about the software analysed, the user who performed the analysis and the date of the analysis.] The argument graphs for the analyses which have been carried out for a software entity are listed in the software's "legal analysis" section. Silke wants to keep her analysis of the license issue of the PerfectPM software private for the time being, at least until the issues have been resolved, and thus decides not to publish the link.

Silke discusses the problem with the technical lead. He uses the MARKOS site to find the license of the EffortManager and sees that it is part of a GPL project. His team had copied just

the object code of the EffortManager from another project that, perhaps erroneously, also used the EPL license. But a thorough analysis of the other project would be required to determine if their use of the EPL was incorrect, since this would depend on how they used to the EffortManager. Merely including a copy of the object code of the EffortManager with their program probably would not require them to use the GPL for their own work.

Silke discusses the legal issues with the legal office of her company. The legal office is of the opinion that dynamic linking does not create works according the law they believe would govern the case, should they be sued by the owner's of the EffortManager software. With the aid of the legal office, Silke (8)[creates a new legal profile tailored to their view of the governing law.] Silke then (9)[performs a new analysis of the licensing issues, using this new profile.] The information she entered during the previous analysis did not need to be re-entered, since it was retrieved from the Markos repository.

The new analysis, using the revised legal profile, provides arguments justifying the publication of the PerfectPM software using a EPL license, as desired by Silke's company. Nonetheless, Silke decides against publishing this analysis on the Markos server. The legal department of her company recommend “letting sleeping dogs lie” by keeping the analysis private until someone, such as the owner of the EffortManager software, makes an issue out of the use of the EPL license.

To be on the safe side, the technical lead searches MARKOS for another component with the required features, but with a GPL license. MARKOS was able to find a couple of possible alternative components, but a closer evaluation of the alternatives lead OpenIT.net to accept the legal risks by continuing to use the EffortManager, which the they consider to be best from a technical perspective.

Analysis of feedback on Usage Scenario 3 (license verification) functionality:

Most users (85%) claim it is important or very important to be able to check whether the Open Source license they have chosen for some project is compatible with the licenses of the components used by the project.

Even though a fully automatic check of licenses compatibility issues, based on a simplified, global model of copyright law cannot take into consideration the specific copyright laws of the country of the users, 85% claim such a check would be useful to them.

All users would prefer to check the license compatibility issues before publication of the software.

Although more than a half of responders said that their company has its own legal staff to assist them with licensing issues, 62% of them does not typically consult a lawyer before selecting an Open Source license to use with their software. 70% claims they have a basic understanding of copyright law, which in their view is sufficient for selecting open source licenses without professional legal advice.

The time typically needed to analyse licensing issues and choose an open source license for your projects is in most cases several hours to days.

Most users stated an interest in having the option to publish an analysis of the licensing issues of their software (left diagram), and use this option to publish the analysis even if it shows that the project might have licensing issues (right diagram).

Figure 5 Interest in publishing license analysis (left diagram) and publishing analysis with licensing issue (right diagram)

The top three steps most important to users:

- Step 4. “obtains an argument graph showing dependencies between various legal assumptions and conclusions about license compatibility and other legal issues” (selected 10x)
- 2. “warning was produced automatically (…) after the PerfectPM software was indexed” (6x)
- 6. “can create a new profile, with the help of the legal office, either “from scratch” or by modifying an existing profile” (5x)
- Further order of step importance: 5,7,1,3,9,8.

Steps pointed out to remove or change are: 2, 6 and 8 (“creates a new legal profile tailored to their view of the governing law”), possibly because they are not completely clear to all users.

Additional user comments and suggested improvements:

- *This scenario is too complex. It describes too many things at once…*
- *The "warning" automatically generated by the system must be based on an "analysis" ... so I would expect to be able to view that analysis (in detail) first, and then maybe change some of its parameters ... it's not clear if this is what is intended by step 3, or whether step 3 is about creating a new analysis from scratch ?*
- *Trusting legal opinion from 3rd party seems to be highly risky here. Would still need to complete analysis and validate the kind of license and how it impacts my business*

Usage Scenario 4 (User Comments) description:

Robert is a software developer who uses MARKOS to search a software library for his current project.

After evaluating the found library, he (1)[writes a review pointing out strengths and weaknesses of the library] and the quality of the meta-information about the library provided by MARKOS. As a regular user, he has registered at the MARKOS system as software consumer and has written, already, a number of such reviews.

Constanza, who is interested to use the same library, (2)[scrolls through the comments on it] and finds Robert's comments useful, not only because it is informative and well-researched, but also because (3) [Robert has a good reputation].

Dirk, who is the responsible project manager for the library, is registered as a software consumer, too. He regularly (4) [checks the reviews of his project] in MARKOS and (5) [improves the information] about and the (6) [functionality of the library] if necessary.

Analysis of feedback on Usage Scenario 4 (User Comments) functionality

No matter what their role is in a project, about 80% of the interviewed expressed their positive (or possible positive) attitude towards providing feedback on all the suggested subjects, i.e.

- provide feedback (reviews) about the quality of the information offered by MARKOS
- provide feedback (reviews) about their experience with the projects/libraries you have retrieved from MARKOS
- check user feedback and change the information about their project if necessary
- give feedback to the reviews of MARKOS users concerning their project

The top four most important features of MARKOS with respect to this scenario are the following:

- 1. "to write a review pointing out strengths and weaknesses of the library" (selected 7x)
- 2. "to scroll through the comments on it" (7x)
- 3. to show a reviewer "has a good reputation" (7x)
- 4. "to check the reviews of a project" (7x)

None of the step should be removed, but one comment pointed out that step 1 should be formalised.

Other comments:

- *open text reviews are better than nothing, but a more formal tagging system for describing code "quality" would be even more useful*
- *I would like to know: how stable the code is, how many years it's been in development/use, how many people worked on it, how reliable/efficient it is, does it carry security risks (e.g. what are the conditions when it shouldn't be used), etc ...*

- *Sometimes results of the Markos analysis can be inaccurate. Other than comments, Markos should give users the possibility to manually insert corrections to the results of the analyses.*

Analysis of general questions and comparing scenarios

We included in the survey also some background and general questions which results will not have a direct impact on the technological development of MARKOS but will be useful for defining overall planning activities, exploitation strategy and for supporting dissemination activities, especially for the engagement of a user community. It is important to remember that this survey has been conducted with a restricted number of people and internally to the consortium. Still this preliminary data can inform our dissemination and exploitation activities, especially if confirmed by the next validation activities that will engage external users.

Firstly, the users were asked to select the role they play in the software development process. It was possible to choose multiple roles per one person. The roles indicated by the responders are mostly the technical ones (Figure 1). None of the responders has a role of Legal Expert or Data Integrator. Also none of the respondents selected a role of Software Consumer.

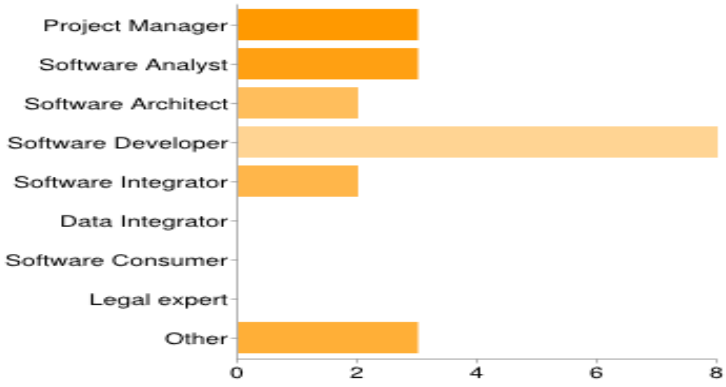


Figure 6 Users roles

We asked in which way the various scenarios could impact the working routine of the users. Figure 7 reports the results obtained.

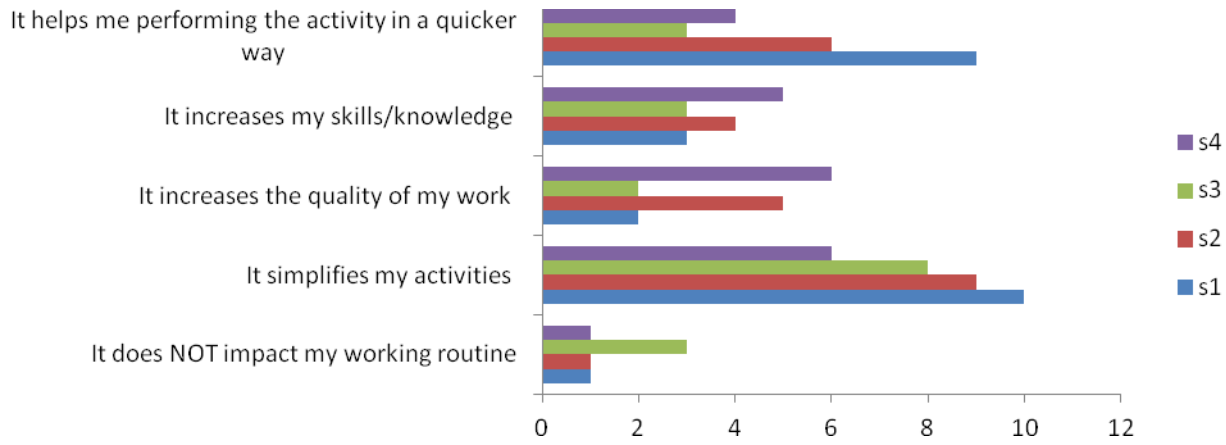


Figure 7 Impacting working routine

Then, we asked to what extent the respondents perceive as useful the MARKOS scenarios. Respondent should select a value from 1 to 6, where 1 is "unhelpful" and 6 is "extremely helpful/must have".

As reported in the table that follows, scenario 2 is the one perceived as more useful, followed by scenario 1 and 4. Scenario 3 is the one that scored the lowest; the value – however - is close to 4, which indicate a certain degree of usefulness of the scenario. The lower results obtained by scenario 3 can be explaining by the job profile of the respondents: in fact we do not have in our sample any legal experts for which this scenario can result more useful than for other users typologies.

Question	Scenario 1	Scenario 2	Scenario 3	Scenario 4
How well will the described functionality be helpful in relation to you in your work activities?	Avg 4,38	Avg 4,54	Avg 3,85	Avg 4,15

Table 1 Scenario usefulness

Finally, we asked the respondents to estimate the percentage of time saving produced by the different scenarios. The percentage of time saving, as shown in the table that follows is between 36% for scenario 4 and 48% for scenario 3. Scenario 3 is the one that occurs with a lower frequency in the user every-day life, but it is the most time consuming (average 13 hours) and produce the higher time saving for single operation. So, even if the scenario is perceived as less useful in relation to work activities, if compared to the others, it is also perceived as the one that will produce the higher time saving. Of course, due to the fact that the related activities are performed with a lower frequency that the others (6 times a year versus 15 times of the activities related to scenario 1), the time saving is not very high in absolute value even if it is higher that the one produced by scenario 4.

Question	Scenario	Scenario	Scenario	Scenario

	1	2	3	4
How long does it take to perform the described activity without MARKOS?	Min: 1 h Max: 40 h Avg.: ~7 h	Min: 1 h Max: 16 h Avg.: ~4 h	Min: 1 h Max: 48 h Avg.: ~13 h	Min: 1 h Max: 6 h Avg.: ~3 h
Please estimate how many times a year you would use a system providing the type of activity described in the scenario.	~15	~22	~6	~15
Please estimate the percentage of time saving produced by this scenario.	~44%	~45%	~48%	~36%
Calculated cost saving	4.620 Euros	3.960 Euros	3.744 Euros	1.620 Euros

In fact, if we use the average values for the hours needed for performing the individual activities (Time for doing: Td) and the average value of the times a specific activity is performed (n. of times: Nt) we can calculate the time saving produced by each scenario. If we, then, attribute a monetary value of a single hour of the user (COSTh), which can be equal to 100 Euros, we obtain the cost saving for scenario. Of course the value of 100 Euro would need to be verified and differentiated for the different figures using the scenarios, but at this point we are only interested in understanding the difference in cost saving produced by the various scenarios in relation to one-each-other.

By using the following formula we can calculate the cost saving for each scenario.

$$\text{Cost Saving} = (\text{COSTh}*\text{Td})*\text{Nt}$$

In this way we can see that scenarios will create a cost saving as given in the last row of the table. All these values refer to the cost saving for a single user. Of course these values are approximated, but still we can see how Scenario 1 is perceived as the most useful and also generate the highest cost savings. In order to validate these results we will need to: a) use a more appropriate proxy for the cost for one hour of work and b) enlarge the sample of respondents.

APPENDIX B: GUIDE FOR FOCUS GROUP WORKSHOP ON MARKOS SYSTEM REQUIREMENTS

Purpose
To understand users opinions and feelings about MARKOS system concept and to discover what users want from the system. To assess how users will use the system and what aspects are important for them.

Welcome

- Thank you for coming and being part of the group. We are grateful for your time.
- In the MARKOS project we are building a system to support the reuse of code in Open Source software.
- The discussion that we have today will help us to design a system that caters to your needs
- We need your input and want you to share your honest and open thoughts with us.

Ground rules

- We would very much like to record these discussions to help us remember them and so that we do not miss any of the issues and ideas you give us.
- We will use your ideas in our report but your names will be kept confidential and you will remain anonymous. So please feel free to openly express your thoughts and opinions.
- WE WANT YOU TO DO THE TALKING. We would like everyone to participate. I may call on you if I haven't heard from you in a while.
- THERE ARE NO RIGHT OR WRONG ANSWERS Every person's experiences and opinions are important.
- Speak up whether you agree or disagree. We want to hear a wide range of opinions.
- It okay that we record this discussion?

Introduce team

- As a first step we should introduce ourselves. Please you start and we will follow. My colleague here will prepare name tags to help us remember your names.

Warm-Up Questions [15 minutes]	Probes	Notes
Please tell us your name. When you introduce yourselves, could you also tell us your role in an organization/project? Tell us about your experience with open source software.		Prepare name tags for participants and facilitators
Introduce the team		

When developing a software do you use third party code and open source components (now and in the past)?	Do you search for source code fragments or libraries? Do other people in your team reuse OSS in this way?	
Where do you search for such open source components?	What portals. Services do you use? What you really like about this page?	
What tools do you use to analyze the software you are going to reuse?		
What aspects are important for you when you search for an open source code to reuse?		
Do you check if the OSS component is compliant with your license? How do you deal with license issues of open source software? Is there anybody else in your organisation that take care about licensing issues?		

MARKOS System Concept Evaluation, part I [30 min]	Probes	Notes for moderators and facilitators
Francesco (and Luca), is going to give a short presentation of the MARKOS system and the first visual presentation of some of the features of the system. During the presentation please note on the post notes the most important keywords that come to your mind. Once he has finished the presentation, I would like to hear your opinions and feeling about the system, its features and the interface.		

What we'd like to do now is hear your opinion about specific product features so that we can modify this product to better meet your needs.	Any free comments? Can you talk about that more? Help me understand what you mean? Can you give an example?	
Would you be interested in using such system? Why or why not? How the system will influence your work activities and what activities in particular?	What would be the greatest benefit to you?	
What are the key features that you liked about the product that we just described?	Which product features are most appealing for the system? Which features would you use? How do you compare the system with the ones you use currently?	
Is the functionality provided enough to make decision whether to use a particular open source software component?	Why or why not? What other information is needed? What other aspects are important for you?	Try to understand what types of information is needed to make a decision?
What would be the greatest benefit to you?	What would be the effect on your work?	
If a colleague or a friend would search for an open source code would you suggest this system to them?	Who can benefit from such system and how? Who apart from already defined roles i.e. project manager, software architect, software analyst, software developer , software integrator, data integrator	
What at are the things you would like to change about the system we described?		
Are there any outstanding questions about the system that you have?	Are there any other comments?	
Please describe a typical scenario you		

would use the system.		

APPENDIX C: PILOT ASSESSMENT COMMENTS

Below are collected all the comments, received from users during the pilot assessment activity, grouped according to five aspects: functionality, information/content, usability, performance and impact/benefit to users.

Comments on MARKOS functionality

- *I had an impression that not all functions are working (e.g. code view)*
- *Good idea ...*
- *Most added value is the tool for license compatibility analysis*
- *I don't understand the result of the analysis. In particular the argumentation "con" is meaningless to me*
- *Primary home / search page: who is this for? This isn't clear from the portal. It's presented on the website for multiple roles, but the system seems either of a generic design for all (not necessarily a criticism), or oriented towards one in particular with other roles' interfaces pending.*
- *Once I find the project, API, implementation, library, etc. that I want... then what? Shouldn't one at least be given a link to go find it on its forge? It seems the use case is missing its last step...*
- *"Search results for: http://markosproject.eu/kb/API/2053383988-190447014" ... shouldn't it repeat my query?*
- *I went through the demo a couple of times and found it to be interesting.*
- *All-in-all the search seemed to be useful and the license analyzer was helpful. I had to hunt around for a couple of terms that would produce some results (ended up using apache), but I really liked that part.*
- *Filtering for me is important as I might be looking for different aspects to include in a project.*
- *The Analyze license feature/workflow was a little confusing (carneades), but I didn't spend a lot of time on it. (apache antunit 1.1)*
- *Several links aren't functioning, e.g. upper and lower bars (still to be implemented, but should probably be hidden) [Jim: - Perhaps add a link in the top bar to the demo page in the project website, as well.]*
- *The license compatibility output is wrong: It lists GPL-2.0 as compatible to a GPL-3.0 project. This is of course not correct, as GPL-2.0 and GPL-3.0 are intentionally incompatible licenses. GPL-2.0+ would have been compatible as GPL-2.0+ allows to apply GPL-3.0 to the code. It is however not sure whether all GPL-2.0+ code could be put under GPL-3.0 without asking the author, as in Europe you are not allowed to sign a contract where you don't know the conditions before signing. So code that was put*

under GPL-2.0+ may not be compatible to GPL-3.0 if there is a European author and the last release of the code has been done before GPL-3.0 was published.

- *It could be very useful to have a judge from legal experts on tricky questions about licences. This could be accomplished extending social capabilities over simple comments, for example in the form of query and legal answers (and maybe a score). Scores and answers should be provided only by recognized experts (users authorized to provide legal responses should be profiled so)*

- *It could be nice to have an additional functionality that show briefly (in a class diagram or something like that) the main dependencies between java classes (or the main java classes). I mean, in general it is useful to have a general picture about a platform/project, before going to see more specific things. For example, before using apache.lucene i briefly observed the hierarchy and the main classes responsibilities...*

Content & Information comments:

- *Lack of enough "help in action" i.e. hints embedded directly into the prototype itself: Tooltips would be useful when using a particular function/feature e.g. I've got a list of compatible licenses but I don't know what it means in practice*
- *Linked Data page... not clear what this is for and what one is supposed to do with it; explanation is needed.*
- *It tells me how many projects have been analyzed, but of what forges, communities, types, etc.?*
- *Examples are a nice addition, like the tooltips these are key and should stay in future iterations, as well.*
- *Lack of tooltips, which is very common for modern web apps... it would be very difficult for a tutorial to replace this. Other web apps sometimes lack in-app explanation because of a common accepted understanding (e.g. webmail), but such a unique app, even if oriented towards a developer and "speaking their language", should focus on being a self-explanatory experience. [For example, in abstract view, some of the characteristics could use a bit more explanation... they're apparent but much based on assumption of understanding; if it's really an analysis tool, explanations are best put within the portal itself, rather than the extensive documentation that would be more oriented to someone looking at MARKOS's open-source. (i.e. the difference between a user and an adopter)]*
- *Filters in search (e.g. API implementation , API, etc.) should some type of explanation.*
- *http://demo.markosproject.eu/ lists 136 projects to be available, but searching did not help to find more than 9 example projects.*
- *MARKOS claims to be language independent, but all listed projects are using java.*
- *MARKOS seems to be intended to analyze license compatibility, but all listed projets except one use the Apache 2.0 license.*
- *I am choosing to send you the note instead of the survey because I don't agree with the wording about "helpfulness" at this point. The limited stack of data made it difficult for me to conceptualize/try some of the features.*
- *I recognize that this is a demo/pilot but saw some search results come back with blank in the abstract, which will get fixed when the bring in more data.*
- *Search results (for the example "apache accumulo")*
    a. *What is "Show only public interfaces (APIs)"? Check mark is taken out again when I open a library, when I click on the check mark a library of "test" (in my example) appears ... Is that library public? (I think no)*

   b. *Explanation of the terms in the "Abstract view", e.g. what is "Manifestation" (time of the module name? time path?)*
   c. *in "SoftwareClass": tab "Source code" without content*
   d. *in "SoftwareClass": click on method/attribute leads to an empty abstract view*
   e. *in "Interface": click on method/attribute leads to an empty abstract view*
   f. *Button "Analyze Licenses"; why new window?; Explanation about the tool something flimsy, what is called for example "per ☑"? There also "contra" is there?*
   g. *Headline about Abract view (e.g. "accumulo. ("Provenance: apache accumulo") hasn't changed, is it meant?*
   h. *Search options under "Advanced search form" part unclear: "Filter by" part is closed, if no result is delivered to the search; "Filter by Keywords": how are multiple keywords I choose from the popular keywords, are linked? "And"? "Or"?*
- *Why not realize a guide with specific example of search. I mean, for people that don't know very well concepts like Licence analyzer*

Comments on usability / navigation

- *GUI is not very intuitive (tooltips would be useful)*
- *... but GUI is not intuitive enough*
- *Filters in "simple search" mode I would keep visible, then I know what should be in result list*
- *I would group results (by tabs, or trunks) to easier select e.g. libraries from other results that match the query*
- *Feedback link hard to see, more attention should be given to it if this is for validation. Survey even more difficult to get to, very unlikely it will be completed.*
- *The parallel launch of Carneades happens without warning, as if it's a completely parallel solution separate from MARKOS [Jim: Even if it is separate software, it still should be presented as part of the same solution; some explanation should come with that "Analyze license" button... with so much space in the Abstract View it seems possible.]*
- *[Filtering ...](Honestly with the bare bones search it would be fine to leave it exposed and not collapsed.)*
- *The interface for browsing seems to be straight-forward, although I ran into some quirks in the interface (click through to methods refreshes position, etc.)*
- *no pagination in search results*
- *search filters should not be collapsed when one of them is selected and search is performed*
- *search results categorisation - tabs maybe? Seperate for programming languages, libraries, constructions, etc.*

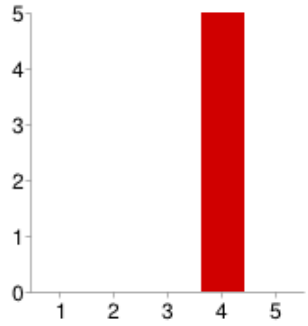
Performance related comments

- *Some times it is a bit slow. However it is actually a good demo Only little experience, no problems for now.*

Responses and comments on Impact

Does MARKOS provide added-value to your current workflow to discover, analyse and integrate OSS?

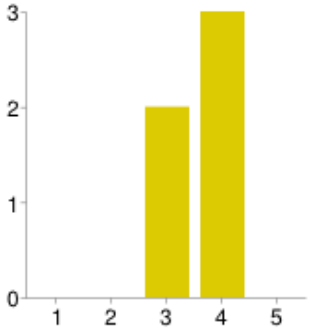
- *In general I never looked at licenses, so why I need such tool. However, I think I could use such tool in future.*
- *Currently no workflow is established on this issue*
- *Even if I've never been too much committed in licences, I think that a good analysis on licences could be important to better understand the topic and to use the software more properly*



1	0	0%
2	0	0%
3	0	0%
4	5	100%
5	0	0%

Does MARKOS help support you in the searching of software components by its expressive queries?

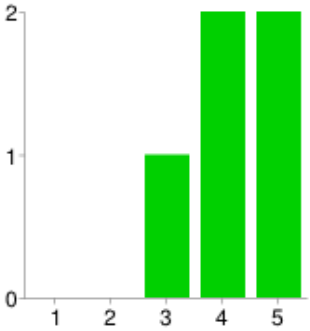
- No additional comments



1	0	0%
2	0	0%
3	2	40%
4	3	60%
5	0	0%

Does MARKOS help facilitate your understanding of software components, their structure, interfaces and dependencies?

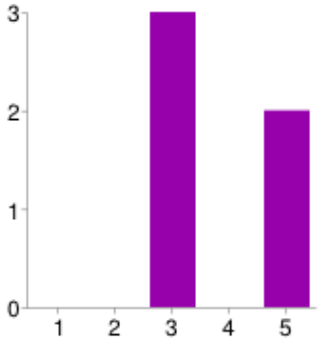
- *The separation of the concepts of a library and its manifestations is very useful*



1	**0**	0%
2	**0**	0%
3	**1**	20%
4	**2**	40%
5	**2**	40%

Does MARKOS provide a more efficient and accurate analysis of licenses than the legal tools currently available to you?

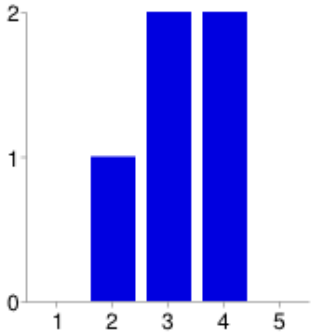
- *Maybe more efficient, but not more accurate*



1	**0**	0%
2	**0**	0%
3	**3**	60%
4	**0**	0%
5	**2**	40%

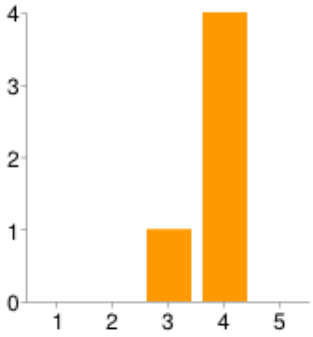
Does MARKOS facilitate the collaboration between different open-source projects?

- *In my opinion more aspects lead to this result: code/licence search is a minor part of the problem*



1	**0**	0%
2	**1**	20%
3	**2**	40%
4	**2**	40%
5	**0**	0%

Does MARKOS provide enough information to make decisions on whether to use a particular open source software component?



1	**0**	0%
2	**0**	0%
3	**1**	20%
4	**4**	80%
5	**0**	0%

Please explain in which way the MARKOS capabilities can impact your working routine and what benefit it could bring.

- *MARKOS can help to understand a project/platform and make decision between different platform/project that are implemented to fulfill the same problems*

- *Markos could be very useful in the eaerly phases of design and development, to prevent issues (both legal and technical) on final deliverables. This is critical especially in software projects where Public Administration is the committer.*

- *semantical code/API search*