WP2

DIGIT B1 - EP Pilot Project 645

Deliverable 12: Feasibility study and method for doing code reviews of free and open source projects in European institutions

Specific contract n°226 under Framework Contract n° DI/07172 – ABCIII

July 2016

DIGIT Fossa WP2 – Governance and Quality of Software Code – Auditing of Free and Open Source Software.

Deliverable 12: Feasibility study and method for doing code reviews of free and open source projects in European institutions

Author:

everis
an NTT DATA Company

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

EUI	European Institutions
EP	European Parliament
DG	Directorate General
FOSS	Free and Open Source Software
FOSSA	Free and Open Source Software Auditing

1 Introduction

1.1. Objective of this Document and Intended Audience

This document represents the deliverable 12 included within TASK-09: '*Draft of the Feasibility study'.*

The objective of this document is to provide a mechanism to analyse the feasibility of code review projects. This deliverable takes into account the results of TASK-06: '*Requirement for the code reviews and their validity for the European Institutions'*, TASK-07: '*Analysis of the methods for communicating the results of code reviews, targeting their automated communication',* and especially TASK-08: '*Design of the method for performing the code reviews for the European Institutions*'.

1.2. Scope

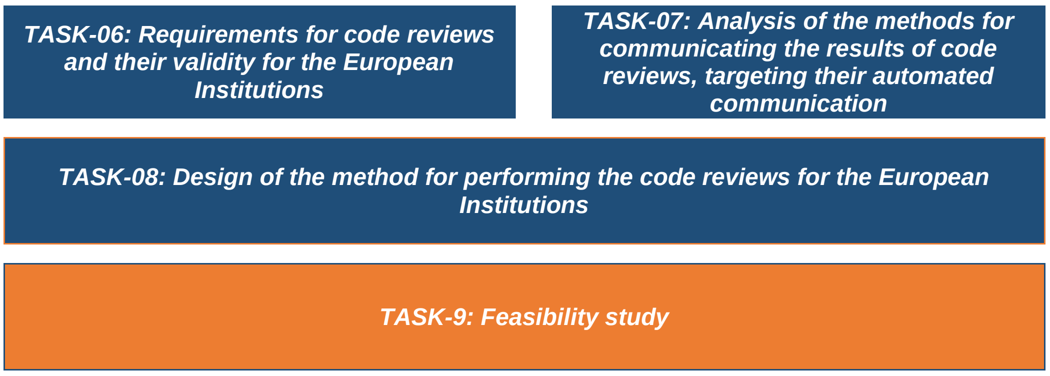
To entirely understand the scope of this document, it is necessary to understand the aim of the Work Package (WP) 2. WP2 encompasses four tasks:

- Task 6: Requirements for code reviews that aim to define the list of requirements for proper code reviews and their validity for the European Institutions, as well as to prepare an analysis of how they fit into the working methods of the European Commission and the European Parliament.
- Task 7: Analysis of the methods for communicating the results of code reviews, targeting their automated communication.

  Tasks 6 and 7 provide the requirements that the methodology defined in task 8 needs to fulfil. For this reason, deliverables 9 and 10 (output of tasks 6 and 7) are complementary.
- Task 8: Design of the code review process to be used in the European Institutions, taking into account the requirements defined in tasks 6 and 7.
- Task 9: Feasibility study of the method defined to perform code reviews, to be used in the European Institutions.

This document is the deliverable 12, the result of task 9 which covers the analysis of the feasibility study of the code review project, using the code review methodology described in Task 8.

Figure 1. WP2 Tasks


TASK-06: Requirements for code reviews and their validity for the European Institutions
TASK-07: Analysis of the methods for communicating the results of code reviews, targeting their automated communication
TASK-08: Design of the method for performing the code reviews for the European Institutions
TASK-9: Feasibility study

1.3. Document Structure

This document consists of the following sections:

- Section 1: **Introduction**, which describes the objectives of this deliverable, the intended audience and the Scope.
- Section 2: **Executive Summary**, which contains the main conclusions of this document.
- Section 3: **Methodological Approach to Building the Analysis**, which defines the approach selected to execute the task 9.
- Section 4: **Feasibility Factors and Project Limitations**, which provides the factors to analyse the feasibility of the code review projects, as well as the limitations.
- Section 5: **Case Study for Analysis,** which includes the information of the case study.
- Section 6: **Executing a Feasibility Study,** which describes how to conduct the feasibility study, using the case study.

1.4. Key Success Factors

The following factors are needed to ensure the success of this phase:

- Identifying the main aspects that affect the feasibility in any code review project.
- Defining a set of factors that ensure the feasibility of the project, based on the aspects previously identified.
- Elaborating a complete case study for analysis using the previous factors.

1.5. Deliverables

1 *Deliverable 9: List of requirements for code reviews*
2 *Deliverable 10: List of methods for communicating the results of code reviews*
3 *Deliverable 11: Design of the method for performing the code reviews for the European Institutions*

2 Executive Summary

This deliverable addresses the evaluation of the feasibility of code review projects. It provides a set of mechanisms to examine the feasibility, based on the project features and particular context.

This will be carried out by analysing the main project requirements (people, knowledge, tools, budget, benefits, legal issues, code for analysis and corresponding documentation, code review methodology, modules for analysis, project duration).

Any code review project has limitations, namely its scope, time and cost. These limits make the difference between the project being feasible or not.

The requirements are analysed using a set of factors, grouped by project aspects. Aspects are selected using the T.E.L.O.S approach, which comprises the following: Technical aspects, Economic aspects, Legal aspects, Operational aspects and Scheduling aspects.

Technical aspects evaluate the Knowledge Factor, which analyses if the project team has the knowledge required to execute the project; and the Tool Factor, which analyses if the project team has the necessary tools to perform the project.

Economic aspects evaluate the Cost Factor, which analyses the budget of the project against the Cost Limit, and the Benefit Factor, which analyses the reasons for carrying out the code review project.

Legal aspects evaluate the Tool License Factor, which analyses if the license of the code review tool adheres to the license terms.

As for the Operational aspects, several factors are evaluated: the Project Staff Factor, which analyses if the project has enough people to execute the project; the Stakeholder Factor, which analyses if the stakeholders are committed to the project; the Code and Code Documentation Factors, which analyses if the code is in the correct format for the analysis, as well as its documentation and The Formal Process Factor, which analyses if the code review project uses a formal methodology, and the Scope Factor analyses if the project scope is respected.

Finally, only one factor is evaluated in terms of Scheduling Aspects: the Time Factor, which analyses the project duration against the Time Limit.

Additionally, two more things are considered in this study: the automation of the code review and the criticality of the Open Source Software in the European Institutions. The automation of the assessment, reporting and results communication is addressed whenever the effects may improve the feasibility by reducing personnel costs, but it adds a cost for maintaining the automatic mechanism.

The Open Source Software code reviews conducted in the European Institutions will help improving the feasibility, as it will increase the benefit factor, by providing reasons to continue this as an ongoing activity

3 Methodological Approach to Building the Feasibility Study

In order to carry out the feasibility study, we are assuming that the code review method explained in Deliverable 11: *Design of the method for performing the code reviews for the European Institutions*, is valid for said Institutions. This study is based on the features of the code review method, especially on its flexibility and its scalability, which provide an adaptive approach for each code review project.

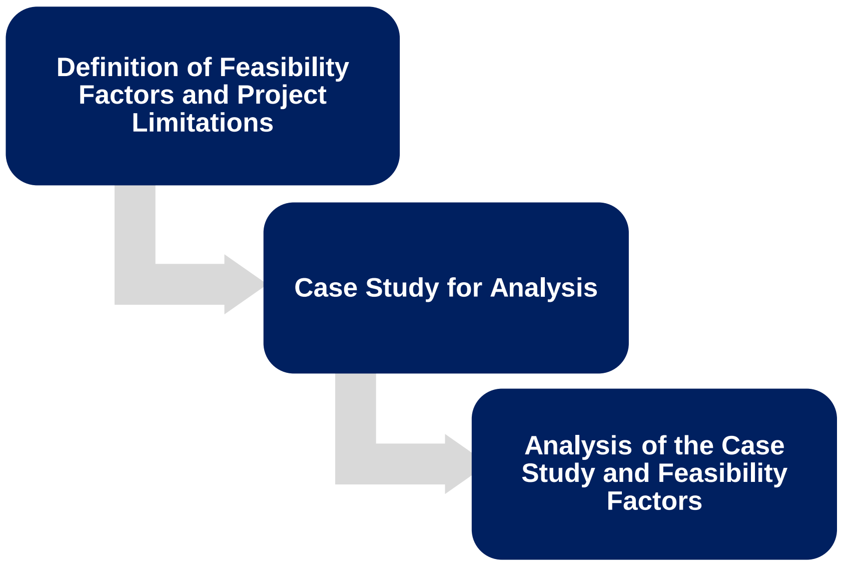
Additionally, each code review project is significantly different from the others in terms of scope and code under test. Since there is no static way to study the feasibility of any code review project, the approach selected uses a set of factors to assess the feasibility. It is important to note that by 'code review project', we refer to any project with the final objective of reviewing the code of any given software, and in which the term 'project' refers to the act of doing the code review, not the development of the software.

As a result, any code review project, even for any kind of organisation outside the European Institutions, has to be analysed in terms of its feasibility, taking into account the scope and particular context,.

In order to explain the factors that ensure the feasibility and how to use them, the methodological approach has been designed as follows:

- Definition of the factors that assess the feasibility of the project, as well as the limitations.
- Prepare a case study to analyse.
- Conduct the analysis of the case study by applying the feasibility factors. Additionally, the effect of the feasibility factors is also included.

Figure 2. Methodological Approach for the Feasibility Study– Steps


Definition of Feasibility Factors and Project Limitations
Case Study for Analysis
Analysis of the Case Study and Feasibility Factors

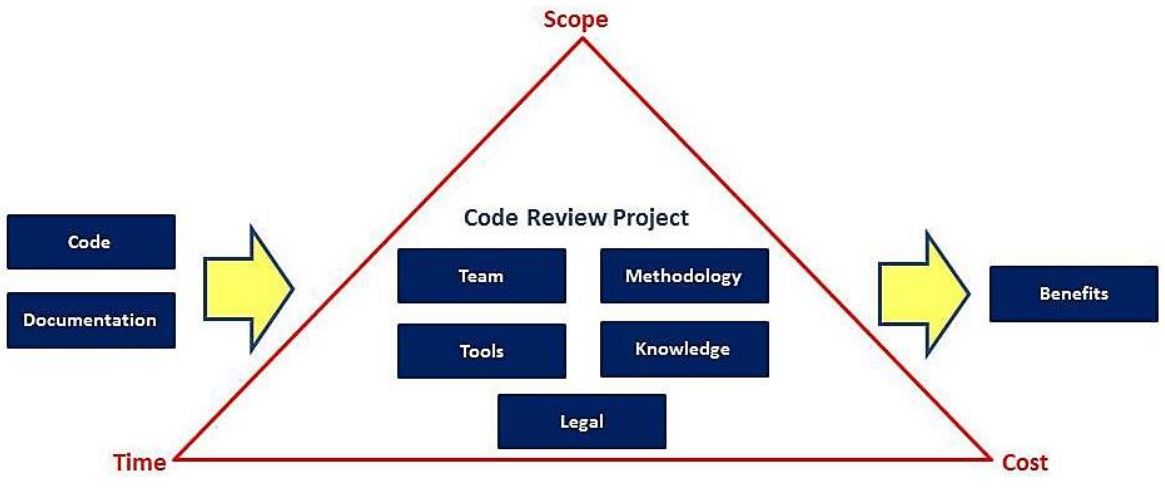
3.1. Definition of the Feasibility Factors to take into Account for the Analysis

In this section, we will provide a set of factors to assess the feasibility of the project, as well as the limitations. We obtained said factors by means of analysing the common features and needs of any code review project. These features and needs are as follows

- The team that will execute the project.
- The knowledge required to perform the project.
- The tools to use during the code review.
- The project budget and cost limit.
- The benefits of the project (the justification for carrying it out).
- The legal issues that could arise.
- The project input, the code for analysis and its documentation.
- A methodology to conduct the code review in an organised and formal way.
- The time required to execute the project, as well as the time limit.

The feasibility factors are a way to ensure that the code review project has everything that is needed to be feasible, and they are aligned with the needs and features mentioned in this subsection.

Figure 3. Code Review Project



3.2. Case Study for Analysis

The design of the code review has to be coherent and as realistic as possible. The information contained in the case study is related to the needs and features described in the previous subsection. This information comprises the description of a code review project:

- The team that is going to execute the project: the number of the people, their skills and knowledge, and their economic rates.
- The context of the code review project: the tools, the code review methodology, the time, the input of the project (code and its documentation), and the benefits.
- The economic aspects and the budget for the project.

3.3. Analysis of the Case Study and Discussion about the Feasibility Factors

The analysis will determine if the feasibility factors are fulfilled in the case study by analysing the information and context. Furthermore, the impact of these factors on the feasibility of the project is also analysed, as well as plausible solutions to fulfil them if they do not meet the requirements.

The goals of this analysis are:

- To provide an example of a feasibility study.
- To analyse the impact of the factors on the feasibility of the project, as well as their interdependencies.
- To provide a way to make a project feasible by modifying the feasibility factors.

4 Feasibility Factors and Project Limitations

The feasibility study takes into account several factors to analyse the methodology proposed in Deliverable 11: *Design of the method for performing the code reviews for the European institutions*.

These factors will provide a mechanism to study the feasibility of the code review method. The factors have been divided in 5 aspects, selected following the T.E.L.O.S. approach [1], as follows:

- Technical aspects, which address the knowledge required to execute the project and the tools to aid during the code review.
- Economic aspects, which assess the project budget and benefits.
- Legal aspects, which analyse the legal issues that could arise.
- Operational aspects, which consider several aspects of the project, namely:
    - The project team.
    - The project input (code and its documentation).
    - The methodology to carry out the project.
    - The project scope.
- Scheduling aspects, which evaluate the time required to execute the project.

These factors depend on the context of the code review project. They must be fulfilled according to the project limitations.

4.1.Project Limitations

There are limitations for the feasibility of the code review project. These limitations are common in project management [2], and they are used as parameters to measure the project status. When a project respects those constraints, it is in good status. If the project does not adhere to them, there is an issue with the status of the project.

Following the approach previously described, a code review project will be feasible if it adheres to the limitations. As we mentioned earlier, the limitations vary depending on the project.

The project limitations analysed in this study are as follows:

- **Time Limit**: the duration of the whole code review project, from the kick-off phase to the end of the project.
- **Scope Limit**: the part of the code that is going to be analysed, and how it is going to be analysed.
- **Cost Limit**: the budget of the project. The maximum amount of money available to execute the project.

4.2.Technical Aspects

Technical factors aim to evaluate the requirements to accomplish the code review from a technical point of view. These technical factors are determined by the technologies and tools necessary to execute the code review. They are as follows:

- **Knowledge Factor**: This factor analyses two different types of knowledge: the analysed programming languages and the security concepts of the code. This knowledge can be contained in the code review method documents, and/or be held by code review staff. This factor is fulfilled when the code review project team meets the knowledge benchmark desired. This degree of knowledge is required to execute all the project tasks (being able to examine any code and detect security flaws). It also evaluates the project team skills that are required to carry out the project.
- **Tool Factor**: The tool factor aims to analyse if the code review project team has the required tools to perform the code review, and if they are correctly installed and configured. This factor is satisfied if the code review project has the necessary tools and they are ready to use.

4.3.Economic Aspects

Economic factors evaluate the economic aspect of the project. Costs and benefits are the two factors to consider, and the way of analysing them is different.

While costs result from the sum of all project resources, benefits are quite hard to determine. Making estimations to quantify the benefits of any preventive measure is difficult, so the approach will consist in listing the benefits without a numeric estimation.

- **Costs Factor**: this factor evaluates the code review project costs, and it is an important element since the feasibility is calculated by adding up the different costs of the project (e.g. staff, testing tool, logistics, and any other costs that can be generated during the project execution). This factor is fulfilled when the total cost remains below the **Cost Limit** established.
- **Benefits Factor**: this factor addresses the justification for the execution of the project, and provides information about the benefits resulting from it. Due to the complexity of measuring the benefits of the project, the approach selected will be qualitative instead of quantitative.

  This factor is fulfilled if the project results achieve one or more of the following objectives:

  - Ensure that the software developed by third-party organisations (FOSS communities, software providers, contractors) is free of backdoors or malicious code.
  - Mitigate the risk of the software components or libraries by detecting vulnerabilities, allowing to choose the most secure option among the software libraries or components available.
  - Improve the security of the software applications under development.

  - Help FOSS communities to improve the security of their software, thus providing more secure FOSS software for EU citizens.

4.4.Legal Aspects

Legal factors aim to analyse the legal aspects of the code review project and their potential effects.

Due to the nature of this kind of projects, legal aspects have a minor effect on the feasibility of the project, because they are related to the software license of the testing tools.

- **Tool License Factor:** It aims to evaluate the software licenses of the tools. The tool involved in the code review has to adhere to the relevant licensing terms. If it is a proprietary tool, the license cost also has to be considered within the cost factor. This factor is fulfilled when the tool adheres to the license terms.

4.5.Operational Aspects

Operational Factors address the evaluation of the effects that operational aspects could incur in code review projects. Any project needs a set of elements; the project input, the processes to transform said input into the project output, and the people to perform the tasks. This is why these factors are related to people, project inputs and project processes.

The factors related to specific individuals are:

- **Project Staff Factor:** It analyses whether the project team includes the required number of people to execute the project. This analysis has to take into consideration the number of people, not their skills (which were already analysed in the **Knowledge Factor**). This factor is fulfilled when the project team has the required number of headcounts to execute the code review project.
- **Stakeholder Factor:** It analyses if the project has been able to engage key people in the project (e.g. software owners, sponsor (DIGIT), project managers), in addition to the code reviewers. They are necessary as they have to provide the software documentation and be available if any other requirements are needed. In the event that a critical vulnerability is detected, they will be contacted to determine how to approach the solution. This factor takes into account the engagement and commitment of the project stakeholders, who are necessary for the success of the project. This factor is fulfilled when those people are actively involved in the code review project.

With regard to the project input there are two factors to consider:

- **Code Factor**: Since this is a code review project, the code to analyse is the project input. This factor is fulfilled when the code is available in the proper format (source code, etc.) to perform the code review.
- **Code Documentation Factor**: It evaluates if the documentation of the code is available. Depending on the scope of the code review, the documentation required may be different in terms of module design or in the description of some functionalities. This factor is fulfilled once the code review team has been provided with that documentation.

The process-related factors are:

- **Formal Process Factor**: It evaluates if there is an established and documented process to execute the code review in a formal way. This process has to be flexible according to the project context. This factor is fulfilled when the code review team uses a formal methodology, like the methodology detailed in Deliverable 11: *Design of the method for performing the code reviews for the European Institutions.*
- **Scope Factor:** It analyses the scope of the code review project, what part of the code or what modules are going to be analysed and how (managed, defined or optimised). In a feasible code review project the scope has to fall within the **Scope Limit.** This factor is fulfilled when the scope of the code review (**Scope Factor**) is less than or equal to the **Scope Limit**.

4.6.Scheduling Aspects

This category contains the factor to analyse the length of the project. Time is an important resource that any project has to adhere to. If not, other project variables could be affected. Due to the variant context of the project, the factors have to be evaluated differently, according to the nature of the project.

- **Time Factor**: It evaluates the total duration of the project. As far as the feasibility is concerned, a code review project will be feasible if the length of the project is shorter than the **Time Limit** stipulated. This factor is fulfilled when the calculation yields a result shorter than or equal to the **Time Limit.**

5 Case Study for Analysis

To assess the feasibility of the code review method proposed in Deliverable 11: *Design of the method for performing the code reviews for the European Institutions*, we will use a case study with non-real data as a guide for future analyses. In order to explain how the study has to be conducted, the use of real data is not relevant because the objective is to describe the evaluation process.

5.1.Information for the Case Study

In this section we will introduce the case study, as well as the main aspects that impact the feasibility. This information will be analysed according to the factors presented in the previous section.

- **Software**: The software that is going to be analysed is an Open Source Java application, composed of 4 modules (‘A’, ‘B’, ‘C’, ‘D’).
  - Module ‘A’ has 4 Java packages, and 12 Java classes on average per Java package.
  - Module ‘B’ has 5 Java packages and 7 Java classes per package.
  - Module ‘C’ has 5 Java packages, and an average of 12 Java classes per package.
  - Module ‘D’ has 6 Java packages, and 11 Java classes per package.
  - For each Java class, the average size of the code is 100 lines.

Table 1: Description of the Software Under Test

Software Under Test				
Modules	**‘A’**	**‘B’**	**‘C’**	**‘D’**
Packages	4	5	5	6
Classes per Package	12	7	12	11
Total of Java Classes per Module	48	35	60	66

To sum up, the application has a total amount of 209 classes and 20,900 code lines. The source code will be provided to the code review team, as well as the assembled version of the code.

- **Code Review Team**: The team is made up of 2 analysts and an external security consultant to provide knowledge in relation to cybersecurity. This team has all the knowledge required about java and secure coding. The team will be managed by a Project Manager, who will dedicate 10% of their time to the project.

Due to the diversity of personnel rates in the European Union, we estimate the team member rate at **CostDay** (variable), regardless of the team member position. This rate also includes indirect costs.

The team will use the methodology defined in *Deliverable 11: Design of the method for performing the code reviews for the European institutions*, and the open source tool 'VisualCodeGrepper (VCG)' [3]. This tool will be correctly installed and configured to execute the code review.

- **Stakeholders:** The main stakeholders for this code review project will be the FOSS community that owns the application and DIGIT, who manages the code review project. The reason for this twofold engagement is that FOSS software is used as a critical component in some of the European Commission's applications. Furthermore, the FOSS community is interested in the project and committed to pursuing its success. They provide the documentation about the software under test via the FOSS community website.

- **Code Review Project**: The objective of the project is to analyse the application modules 'A', 'B' and 'D' using the 'Defined mode' of the code review method introduced in Deliverable 11: *Design of the method for performing the code reviews for the European Institutions*. Model 'C' will also be analysed, but using the 'Optimised mode'.

  According to the previous data, 149 classes will be analysed using the 'Defined mode', and 60 classes using the 'Optimised mode'.

  For this purpose, the analysis of controls is conducted as follows: when several (e.g. two or three) pieces of evidence are detected for a specific control, the control fails. The control will also recommend reviewing the rest of the module for that specific error. The reason is to avoid time-consuming tasks that do not provide value. Furthermore, not all the controls are applicable to all the modules and classes.

  In this case study, all modules and classes are going to be analysed because it is a small application. Nevertheless, the amount of efforts and resources to analyse the large ones may not be affordable.

  In the event that a big application has to be analysed, an efficient option would be to limit the scope by reducing the portion of the code to analyse especially in the manual review ('Defined mode' and 'Optimised mode'). If reducing the scope is required, in our experience, between 5% to 15% of the code suffices (it is a significant amount of code from a security point of view, taking into consideration input / output management, user management, error management, file management, cryptography, etc.) as a trustworthy sample to review the most critical aspects of the application manually.

  The estimated time for code analysis depends on several factors, namely:

  - The mode selected from the methodology ('Managed mode', 'Defined mode', or 'Optimised mode').
  - The documentation of the code (low quality, medium quality, high quality).
  - The complexity of the code (low, medium, high).
  - The difficulty to analyse the control. Some controls are faster to review than others.

In the current case study, the documentation is of high quality. After a preliminary analysis, the complexity of the modules is as follows:

- The complexity of module ‘D’ is low.
- The complexity of modules ‘A’ and ‘C’ is medium.
- The complexity of module ‘B’ is high.

Table 2: Complexity of the Software Under Test

	Software Complexity			
Modules	**‘A’**	**‘B’**	**‘C’**	**‘D’**
Complexity	MEDIUM	HIGH	MEDIUM	LOW

This complexity is estimated according to the number of categories of the methodology controls that are applicable (such as data/input management, authentication, authorisation, etc.) to the module (software) under test. The classification criteria is presented below:

- If categories 1 to 4 are applicable, the complexity of the module is low.
- If categories 5 to 7 are applicable, the complexity of the module is medium.
- If categories larger than 7 are applicable, the complexity of the module is high.

Table 3: Classification of the Complexity of the Software

Complexity	LOW	MEDIUM	HIGH
No. of Categories of the Methodology Controls	0 < n < 5	4 < n < 8	7 < n

If the code complexity is medium, one member of the code review team can analyse 2 Java classes per hour using the ‘Defined mode’ (200 code lines), and 1 Java Classes per hour using the ‘Optimised mode’ (100 code lines). If the code complexity is low, the speed of the analysis increases by 30%, while a high complexity decreases the speed by 30%.

However, these estimations assume that the quality of the documentation is high, which is correct for this case study. In the event that the quality of documents is different, the estimation will be larger:

- If the quality is medium, the estimations have to be increased by 50%.
- If the quality is low, the estimations have to be increased by 100%.

Taking into account the previous information:

- The analysis of module 'D' will take 8.5 hours (3 team members x 1.30 complexity factor x 2 classes per hour and 66 classes).
- The analysis of module 'C' will take 20 hours (3 team members x 1.00 complexity factor x 1 class per hour and 60 classes).
- The analysis of module 'B' will take 8.3 hours (3 team members x 0.70 complexity factor x 2 classes per hour and 35 classes).
- The analysis of module 'A' will take 8 hours (3 team members x 1.00 complexity factor x 2 classes per hour and 48 classes).

Table 4: Estimation Time of Analysis

	Software Under Test			
Modules	**'A'**	**'B'**	**'C'**	**'D'**
Total of Java Classes per Module	48	35	60	66
Team Members	3	3	3	3
Complexity Factor	1	0.7	1	1.30
Analysed Classes per Hour and Person	2	2	1	2
Effective Analysed Classes per Hour (team)	6	4.2	3	7.8
Estimated Time for Analysis (h)	48 / 6 = 8 h	35 / 4.2 = 8.3 h	60 / 3 = 20 h	66 / 7.8 = 8.5 h

The total number of hours needed amounts to 44.8 hours (8 + 8.3 + 20 + 8.5), or 5.6 days (counting 8 working hours per day and three people). This entails 1 week and 1 day (counting 5 working days per week). At this rate, the cost of the code review is **5.78CostDay** (including the project manager).

Additionally, according to the methodology, the assessment phase requires 7 additional days, but it will be executed by one team member, which costs **7.7CostDay**. The reporting phase will require 5 additional days and one more person, thus the cost of the reporting phase will be **5.5CostDay**. Thus the minimum budget for executing the code review project is **18.98CostDay**, and the minimum duration is 14 days (2 days for the code review + 7 days for the assessment phase + 5 days for the reporting phase).

5.2.Project Limitations

For this project, the code review project executor (in this case, DIGIT) has established the following limitations:

- **Time Limit:** The maximum duration of the project will be 1 month.
- **Scope Limit:** The scope is defined depending on the modules of the application under test, in this case 100% of the code will be reviewed as follows: modules 'A', 'B' and 'D' will be analysed using the 'Defined mode' of the methodology; while module 'C' will be analysed using the 'Optimised mode' due to its criticality.
- **Cost Limit:** The budget for the code review project will be **30CostDay.**

Taking into account the results of the previous information, we analysed if the project limitations were accomplished as follows:

- **Time Limit** is fulfilled since the code review project needs 2 weeks and 4 days, which is less than the project time limit defined.
- **Scope Limit** is fulfilled because no more code than the defined in the project limitations is analysed.
- **Cost Limit** is fulfilled since the established limit is larger than the project cost (**18.98CostDay**).

6 Executing a Feasibility Study

This section addresses how to conduct a feasibility study, taking into account the factors and limits explained in Section 4: Feasibility Factors and Project , and the information of the case study. In order to explain it, this section will provide an example to analyse the different feasibility factors, and their impact on the feasibility of the project.

6.1.Technical Factors

- **Knowledge Factor**: As we previously explained, this factor aims to analyse whether the project contains the knowledge required to achieve its ultimate goal. According to the information presented in the case study, there are different knowledge sources in this project.

  One is the code review project team, which encompasses the knowledge on programming languages (team analysts) and a broad knowledge about software security (external consultants). The controls and checks in the code review methodology are yet another source, where some knowledge about the security of the code is contained. In the light of the previous information, we can conclude that the current project, with its limitations, has the knowledge required and is thus feasible.

  As far as the skills are concerned, the profiles are suitable for the execution of the project (2 analysts, one security expert, and a project manager).

  In the event that there is a gap in the knowledge required about security or in the technologies involved, some measures will be taken to ensure the feasibility, provided that these measures adhere to the project limitations (scope, time, cost). Among the measures that can be taken are:

  - Incorporate a contractor (person or organisation) to fulfil the lack of knowledge (It affects the **Cost Factor).** FOSS communities are among the organisations that should be contacted to engage in the project, and which can also be contracted as they have knowledge about the technologies used.
  - Provide the team with documentation and/or training in the knowledge required (It affects the **Cost Factor** and the **Time Factor**).
  - If the above measures are not feasible, modify the project limits as needed (e.g. **Time Limit**, **Cost Limit**).

- **Tool Factor:** This factor analyses if the project team has the tools to execute the code review. Analysing the information of the case study, we see that the project already uses a FOSS code review tool, and therefore it does not generate further costs. Nevertheless, if the tools were not free, the cost should be added to the **Cost Factor.**

  In the event that a code review tool is not available, or is not installed or configured, this factor will not be fulfilled. In such a scenario a tool has to be procured, installed and configured. This results in additional time to be added to the **Time Factor**, and the cost of the tool (if any) and the project team rates have to also be added to the **Cost Factor.** The feasibility of the project could be affected if the **Time Factor** or the **Cost Factor** exceed their corresponding limits.

6.2.Economic Factors

- **Cost Factor**: This is one of the most important factors for the feasibility of the project. This factor is the sum of all project costs, and it is a measurement of the required efforts that the organisation executing the project has to make. As previously indicated, this factor has to remain below the **Cost Limit** in order to be fulfilled.

  In the case study, this factor (**29.36CostDay**) amounts to less than the limit (**30CostDay**), so it is fulfilled. If this factor exceeds the **Cost Limit**, there are two different strategies to follow: review the project structure to reduce the costs or change the **Cost Limit**.

- **Benefit Factor**: As previously mentioned, it is difficult to measure quantitatively the benefits of the project, so the approach for this factor is qualitative. In the case study presented, it is assumed that DIGIT would execute this code review as they are the ones seeking to ensure the security of the software under test, so the **Benefit Factor** was fulfilled (This benefit coincides with the benefit 'To ensure that the software developed by third-party organisations (FOSS communities, software providers, contractors) is free of backdoors or malicious code').

6.3.Legal Factors

- **Tool License Factor**: This factor analyses the license of the code review tool, as explained in Section 4. For the case study, the tool is a FOSS that can be used without restrictions (according to the license), thus this factor is fulfilled. For instance, for proprietary tools the cost has to be considered in the **Cost Factor**.

  If this factor is not fulfilled:

  - If the **Cost Limit** does not allow the cost of a proprietary tool or the software license is not compatible, another proprietary tool or a FOSS tool (with a compatible license) has to be selected.

6.4.Operational Factors

- **Project Staff Factor**: This factor analyses whether all the individuals required for the success of the project are taking part in it. According to the case study information, the project already has all the required headcounts to execute the project. In this factor we do not analyse the knowledge or skills of the code review team, we only analyse the number of team members.

  The main element that impacts this factor is the estimated amount of work (for instance, code files per hours) that can be done during the code review analysis. Depending on the number of headcounts involved, this estimation will be larger, and thus the **Time Factor** will be reduced. Nevertheless, the **Cost Factor** will increase as the number of team members increases.

  This factor will be not fulfilled if the number of team members is not enough to cover the tasks required before the **Time Limit** deadline**.** The measures to mitigate this problem will be to add new team members (provided that the **Cost Factor** is not greater than the **Cost Limit**), or to reduce the scope of the project.

Additionally, it is highly recommended to take into consideration previous code review projects for work estimations, as the estimation of the number of people needed for the project and the analysis of this factor will be more accurate.

- **Stakeholders Factor:** This factor analyses whether the main stakeholders related to the project are engaged in the code review project. In the case study, the FOSS community that owns the software is involved and committed to the project. DIGIT, the organisation executing the code review project, is also committed to the project. The stakeholders are part of the project in the sense that they are interested, involved and committed, then this factor is also fulfilled.

  In the event that this factor fails, several measures can be taken:

  - Promoting security awareness and the project benefits and communicating them to the software owner.
  - Promoting the project benefits to the organisation executing the code review (in our case study, DIGIT).
  - Improving the communication with the software owners and/or the organisation executing the code review, with regard to the evolution of the project.

- **Code Factor**: This factor analyses whether the code is available to perform the code review, and if the code format is correct. In the light of the information provided by the case study, the code is provided both in the source code format and in the assembled format, so this factor is fulfilled. If this factor is not fulfilled because the code is not provided in the correct format, then several measures can be taken:

  - Informing the stakeholders of the project (mainly the software owners), to fix this issue by providing new code.
  - Trying to engage the IT team that developed the software.

- **Code Documentation Factor**: This factor studies if the required documentation (design information, software modules, etc.) is provided. In the case study we saw that the documentation is provided, thus this factor was fulfilled. Nevertheless, in the case that this factor is not fulfilled, some measures can be considered:

  - Asking software owners and/or the IT team for additional documentation.
  - Dedicating code review team efforts to understand the unclear parts of the software. This definitely affects the **Cost Factor** and the **Time Factor,** and it may also affect the **Cost Limit** and/or the **Time Limit**.

- **Formal Process Factor**: This factor addresses the implementation of a formal methodology for executing a code review. Due to the use of the code review method explained in Deliverable 11, this factor is fulfilled in the case study.

  Any code review project that uses the FOSSA code review method will be fulfilled because it is a formal methodology. If they use other formal methodology this factor can also fulfilled, however the code review method has to be analysed to determine if it is a formal one.

- **Scope Factor**: This is a critical factor to analyse, since it affects several other factors (e.g. **Cost Factor, Time Factor**, etc.). The scope has to be clearly defined in any code review project, identifying what is to be analysed (which software modules fall within the scope and which do not), and the methodology/method. In the case study, the scope was clearly defined and it adheres to the **Scope Limit**, so this factor was fulfilled.

During the execution of the project, changes in other factors can affect the scope, so this factor has to be continuously assessed during the execution phase, together with other factors that may affect the project limitations.

Some measures are available in case this factor is not fulfilled:

- Redefining the project scope and the **Scope Limit,** provided that the rest of the factors are still fulfilled (Specially the **Cost Factor**, and the **Time Factor**).
- If the scope has to be reduced, the stakeholders should be contacted to reach an agreement on this regard.

6.5.Schedule Factors

- **Time Factor**: This factor analyses the project duration to ensure it does not exceed the **Time Limit** stipulated, as explained in Section 4.6. Taking into account the information of the case study, we can conclude that this factor was fulfilled, because the **Time Limit** was greater than the final estimation.

  Nevertheless, if this factor is not fulfilled, some measures can be taken:

  - Modifying the current **Time Limit** by extending the time to conduct the code review. However, this could affect the **Cost Factor** and the **Cost Limit.**
  - Reducing the **Time Factor** by means of rescheduling the project**,** however this could also affect the **Scope Factor** and even the **Scope Limit.**

6.6.Case Study : Feasibility Study Conclusion

Taking for granted the hereunder elements (cf. table), it is feasible to execute code review on the depicted case study.

Table 5: Feasibility Study Conclusion

Feasible Factors	Fulfilment	Comments
Knowledge Factor	YES	Project Team has the knowledge to execute the code review project.
Tool Factor	YES	Project Team has the tools to conduct the code review.
Cost Factor	YES	The estimated cost of the project is less than the **Cost Limit**.
Benefits Factor	YES	The code review project provides benefits that are considered as successful ones ('To ensure that the software developed by third-party organisations - FOSS communities, software providers, contractors - is free of backdoors or malicious code').
Tool License Factor	YES	The tool is a FOSS software, and the license is adhered to.
Project Staff Factor	YES	Project Team has the required number of people to execute the project.
Stakeholders Factor	YES	The key stakeholders are committed with the code review project.
Code Factor	YES	The code is available in the correct format to be analysed.
Code Documentation Factor	YES	A good quality documentation is provided to the code review team.
Formal Process Factor	YES	A formal methodology (described in Deliverable 11) is used to execute the code review project.
Scope Factor	YES	The project scope does not exceed the **Scope Limit.**
Time Factor	YES	The project duration is shorter than the **Time Limit.**

6.7.Other Considerations

Since some improvements can be implemented to enhance the feasibility in the particular case of the European Institutions and their usage of FOSS, additional considerations have to be added:

- ‘Automatic Communication of Checks and Results’ addresses the impact of automating the dissemination of results for code reviews on the feasibility of the projects.
- ‘Criticality of the Open Source Software for European Institutions’ analyses the impact of the usage of Open Source Software in the feasibility of the code review projects.

6.7.1.Automatic Communication of Checks and Results

In the example proposed in Section 5, the assessment and reporting phases are manual, resulting in an increase in project costs due to the fact that a team member has to carry out both tasks. The days needed for that purpose have an impact on the amount of **CostDay**s and on the duration of the project.

Nevertheless, both phases can be automated to some extent:

- In the assessment phase, the global assessment can be automated if it is done from an automatic process that evaluates each finding. If the project team has to evaluate only the findings, and the results are aggregated without human intervention, the project duration and budget will be reduced;
- In the reporting and communication phase, automation is a key feature that significantly improves the feasibility of any code review project. The generation of audit reports can be almost completely automated, provided that the results of the code review are ready for the automation. Some global analyses cannot be automated, but thanks to the collection of all results put together, the generation and submission of reports via email or via communication with other systems (e.g. wikis, Content Management Systems, etc.) will significantly reduce the time and days needed for the team to perform this phase.

The automated communication process is analysed in *Deliverable 10: ‘List of methods for communicating* the results of code reviews’, and the methods proposed can improve the feasibility of any code review project, allowing a faster execution of code review projects while consuming fewer resources (people, budget).

One way to do it is by developing an application to automate those tasks where the results of the ‘checks’ (checks of the methodology controls) used for each code file are added to the application, as well as the assessment (the global analysis). Then, the addition of the results, reporting and communication will be automatic while the global analysis will be done manually.

It is important to note that the application should be maintained, and this will be a cost to add in all code review projects, however this cost is significantly lower than not having it and increasing the exposure to security risks as a result.

6.7.2.Criticality of the Open Source Software for European Institutions

European Institutions use Open Source Software, directly or indirectly, via software libraries or components. Security is mandatory for the European Institutions due to the strategic importance of the information they manage. It is crucial that all their information systems are protected and secured correctly and at all times.

Software security is key for European Institutions. The recent disclosure of critical vulnerabilities in FOSS software being used by the European Institutions has raised the awareness on security, thus highlighting the importance of conducting code reviews.

The examination of the FOSS used by European Institutions should be mandatory to detect possible security flaws. Many European citizens and organisations use FOSS and therefore are exposed to these flaws as well.

Taking advantage of the 'open' nature of FOSS, the code can be examined to ensure the security of it. This way, European Institutions, European citizens and organisations can all rely on the software they use.

One last point to highlight is that the open source tool used in the code reviews should be code reviewed as well, as it is important to know if the used tool can be trusted to ensure reliable results

Though the cost of the impact of exploited vulnerabilities is quite difficult to assess before they happen, it may be much larger than the cost of the code review project. Additionally, many indirect costs (tangible or intangible) result after a cyber-attack, such as reputational damage to the organisation (e.g. European Institutions) or the actual cost of information disclosure. Taken the above into account, the benefit factor will be fulfilled and therefore increase the feasibility of the code review project

6.7.3. Findings and results of the code review project

The feasibility of the code review projects does not imply that all security bugs are found during the review. There are different reasons to justify that not all security bugs can be detected:

- Code reviewers analyse the code, so even with the use of automated tools, some bugs can be overlooked.
- The experience of the code reviewers is a factor to take into account, veteran reviewers will make less mistakes than novice ones.
- Some security flaws are quite complex to detect, as they require the analysis of different parts of the code while analysing a specific code review methodology control. These can be detected more easily during dynamic testing, such as penetration testing.
- To increase the detection rate of the security flaws, static and dynamic security testing should be carried out, as they are complementary.
- Some of the potential insecure code parts require a deep understanding of code security to be detected, so the experience of the code reviewer plays a crucial role in their detection.
- Code reviewers earn experience by practice, thus improving their knowledge in code security while reducing the time to review the already known bugs. As a result, feasibility of the code review projects is improved by reducing the project time and cost.

To conclude, code review projects may not detect all security bugs, so they have to be complemented by dynamic testing. Additionally, complex code can have bugs difficult to find that may require additional time and more expertise to be analysed, increasing the project duration and cost. Finally, as more code review projects are conducted, the feasibility factors can be optimised will in turn make the projects more feasible.

7 References

[1] Wikipedia, "TELOS," [Online]. Available: https://en.wikipedia.org/wiki/TELOS_%28project_management%29. [Accessed May 2016].

[2] C. C. a. T. Johnson, "A short course in project management," Microsoft, [Online]. Available: https://support.office.com/en-us/article/A-short-course-in-project-management-19cfed57-2f85-4a44-aadc-df8482d92688. [Accessed May 2016].

[3] NCCGroup, "VisualCodeGrepper - Code security scanning tool.," [Online]. Available: https://github.com/nccgroup/VCG. [Accessed May 2016].

[4] J. Stecklein, "Error Cost Escalation Through the Project Life Cycle," 2004. [Online]. Available: http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100036670.pdf. [Accessed July 2016].