WP1

DIGIT B1 - EP Pilot Project 645

Deliverable 3: **Analysis of Software Development Methodologies Used in the European Institutions**

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

January 2016

DIGIT Fossa WP1 – Governance and Quality of Software Code – Auditing of Free and Open Source Software.

Deliverable 3: Analysis of Software Development Methodologies Used in the European Institutions

Author:

everis
an NTT DATA Company

Contents

List of tables

List of Figures

Acronyms and abbreviations

EI	European Institutions
EP	European Parliament
DG	Directorate General
FOSS	Free and Open Source Software
FOSSA	Free and Open Source Software Auditing
OS	Operating System
SDLC	System Development Life Cycle
SEO	Search Engine Optimization
WP	Work Package
API	Application Programming Interface
ESAPI	Enterprise Security Application Programming Interface

1. Introduction

1.1. Objective of this Document and Intended Audience

This document represents the deliverable 3 included within TASK-01: Analysis of software development methodologies used in the European Institutions.

The objective of this document is to analyse the software development methodologies, tools and best practices used in the European Institutions' projects that were selected and prioritised in Deliverable 1.

This document is targeted at the DIGIT areas interested in the study of the software development methodologies, related practices and tools used in the European Institutions (European Commission and European Parliament).

1.2. Scope

The analysis covers the European Institutions' projects selected and prioritised during the development of Deliverable 1 and whose project sponsors were interviewed

Throughout the document, the term "European Institutions" refers to the projects that fall within the defined scope.

1.3. Document Structure

This document consists of the following sections:

- Section 1: **Introduction**, which describes the objectives of this deliverable, intended audience and Scope.
- Section 2: **Methodological Approach to building the analysis**, which describes the steps followed to conduct the analysis of the different methodologies, tools and best practices used in the European Institutions' FOSS projects selected, according to the scope.
- Section 3: **Software development methodologies, best practices and tools** used in the European Institutions.
- Section 4: **Analysis** of the identified software development methodologies used in the European Institutions.
- Section 5: **Bibliographical** references.
- Section 6: **Annexes**.

1.4. Key success factors

All the steps described in Section 2 - Methodological approach to building the analysis, will ensure the fulfilment of the key success factors related to this deliverable

- To have a complete stock of methodologies used both in the European Institutions and in the FOSS communities that were selected for this project
- The Best practices will include a variety of typologies: technical, organisational and about the governance and quality of open source software (e.g.: synchronisation with FOSS; guidelines for secure software development; secure integration and interoperability of different components; sustainable ways of FOSS governance and professional services).

2. Methodological Approach to Building the Analysis

The goal of this document is to analyse all the information gathered during the interviews and the documentation that is relevant for the purpose of the study. This analysis will provide valuable information from the perspective of the European Institutions´ projects identified with regard to:

- Software development methodologies in use
- Best practices in use
- Tools in use
- Release management
- Incident management
- Security aspects related to software development
- Additional necessities identified by stakeholders and their points of view regarding how European Institutions can contribute to ensure that the widely used critical software can be trusted.

2.1. Selection of Projects, Engagement with the European Institutions and Information Gathering

Deliverable 1 provided a list of 15 projects to be analysed. Out of the 15 projects, 11 were from the European Commission and the remaining 4 from the European Parliament.

For this step, the following activities were conducted:

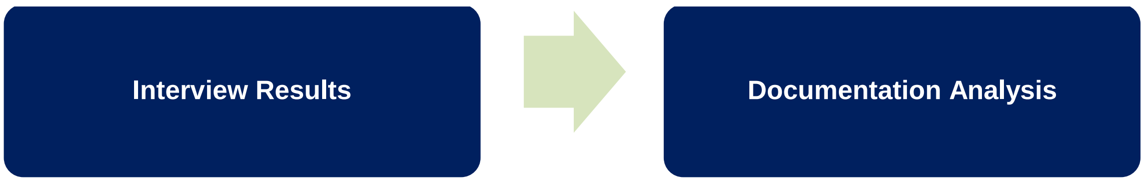
- Deliverable 1 provided a list of 15 projects to be analysed.
- To engage with the project owners, the Project Officer from DIGIT sent an executive summary explaining the importance of the FOSSA project, and requesting their availability for an interview to gather information on their particular project.
- The Project Officer from DIGIT developed an interview planning).
- 14 out of 15 projects were covered during the interview rounds.

The information gathering of the 14 projects was conducted in 12 interviews

2.2. : Information Classification and Filtering Process

The following figure shows which information sources were used to conduct the analysis.

Figure 1. Methodological approach to build the analysis- Information sources



Interview Results: During the interviews with the European Institutions´ project owners, we used a questionnaire to obtain the relevant information for the study. Since the interviews were conducted as an open discussion, the information gathered was filtered and classified to conduct the analysis. For this purpose, a spread sheet was created to count the number of projects using a specific methodology, practice or tool under analysis. Only common criteria were taken into account and the analysis does not include what is particular to a project, unless it is relevant for the study. After filtering and classifying the data, each methodology, practice or tool was compared with other projects showing the percentage of usage within the projects analysed.

- **Documentation Analysis**: In order to complete the information related to the methodologies, best practices and tools identified, public documentation was analysed in order to fulfil the aspects mentioned above.

2.3. Analysis of the Information

Sections 3 and 4 of this document are structured following two main purposes:

- **Software development methodologies, best practices and tools used in the European Institutions**: For each of the methodologies, best practices and tools gathered from the interviews, a form is developed in order to complete the information about each variable.

- **Analysis of identified software development methodologies, best practices and tools used in the European Institutions**: This section is structured following four main points to conduct the analysis:

    - **Software Development Lifecycle**: It contains the analysis of methodologies, practices and tools used within the different phases of the project development.

    - **Quality Assurance and Maintenance**: It analyses the methodologies, practices and tools used to ensure the sustainability of the projects and their quality.

- **How European Institutions contributes to FOSS Communities**: It analyses the real contribution of the projects and teams to FOSS communities.

- **Relevant opinions and advices from interviewees**: It contains interviewees' personal opinions and pieces of advices expressed during the interviews.

The usage of each analysed variable is represented by a numeric value and a percentage. To represent these numbers, we used two different approaches:

- **Tables**: Represent the percentage of usage for the total number of projects analysed. It is important to note that the variables are not mutually exclusive; therefore, a project can use one or more of them.
  To calculate this percentage, we used the following formula:

$$\%usage = nCoincidences * 100 / nProjectsAnalysed$$

Pie charts: These charts allow a quick reading of the results since the percentages of usage are represented graphically. The variables analysed using this approach are exclusive; therefore, a project can only use one of them.

3. Software Development Methodologies, Best Practices, Frameworks, Libraries and Tools Used in the Projects Analysed from the European Institutions

3.1. Methodologies Used by the Analysed Projects During the Software Development Lifecycle

M1.	**Methodology Name:**	**PM^2**

Use	**Objectives**	**Benefits**
Project management.	The objective of this methodology is to enable Project Managers to deliver solutions and benefits to the European Commission through the effective management of projects	• A project Governance Structure tailored to the European Commission • PM2 process guidelines. • Artefact Templates. • A set of effective Mind-sets.

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. PM, developer, etc.)	Project manager
Related Methodologies, Best Practices and Tools	PMBOK
Related Technologies	

Project Using This Methodology	Project 1	X	Project 5	X	Project 9	
	Project 2	X	Project 6		Project 10	
	Project 3	X	Project 7	X	Project 11	X
	Project 4		Project 8	X	Project 12	X

M2.	**Methodology Name:**	**PM for EP**

Use	**Objectives**	**Benefits**
Project management	The objective of this methodology is to enable Project Managers to deliver solutions and benefits to the European Parliament through an effective project management methodology	Project management governance, guidelines, templates and artefacts tailored to European Parliament projects

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. PM, developer, etc.)	Project manager
Related Methodologies, Best Practices and Tools	PMBOK
Related Technologies	

Project Using This Methodology						
	Project 1		Project 5	X	Project 9	
	Project 2		Project 6	X	Project 10	
	Project 3		Project 7		Project 11	
	Project 4	X	Project 8		Project 12	

M3.	**Methodology Name:**	**Scrum**					
Use		**Objectives**		**Benefits**			
Software development		Manage the software development using an iterative and incremental agile method		This methodology maximizes the team's ability to deliver quickly, to respond to emerging requirements and to adapt to evolving technologies and changes in market conditions.			
SDLC Phase Where It Is Used							
Analysis	Design	Development		Testing	Deployment		
X	X	X		X	X		
Roles (i.e. PM, developer, etc.)		Scrum Master, Product Owner, Development Team Member					
Related Methodologies, Best Practices and Tools		Agile					
Project Using This Methodology	Project 1	X	Project 5		Project 9	X	
	Project 2	X	Project 6		Project 10		
	Project 3		Project 7	X	Project 11		
	Project 4	X	Project 8		Project 12	X	

M4.	**Methodology Name:**	**Agile@EC**

Use	**Objectives**	**Benefits**
Software development	Manage the software development using an iterative and incremental agile method.	An agile method tailored to European Commission projects

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	

Roles (i.e. PM, developer, etc.)	Business analyst, system architect, test architect, project manager, tester, developer
Related Methodologies, Best Practices and Tools	Agile
Related Technologies	

Project Using This Methodology	Project 1		Project 5		Project 9	
	Project 2	X	Project 6		Project 10	X
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

M5.	**Methodology Name:**	**Kanban**

Use	**Objectives**	**Benefits**
Software development	Drive the software development process through well-defined and limited phases, which ensures the completeness and the quality when an artefact goes to the next phase.	Improve the quality and classification for the development of artefacts and to reduce bottlenecks.

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. PM, developer, etc.)	No existing roles. (The help of an agile coach)
Related Methodologies, Best Practices and Tools	Agile
Related Technologies	

Project Using This Methodology	Project 1	X	Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7	X	Project 11	
	Project 4		Project 8		Project 12	

M6.	**Methodology Name:**	**Waterfall**

Use	**Objectives**	**Benefits**
Software development.	In this methodology, the software development activity is divided into different phases and each phase consists of series of tasks with different objectives. All these phases are linked and they have to be executed in the right order.	It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process. Phases are processed and completed one at a time and they cannot overlap.

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. PM, developer, etc.)	Project manager, Business analyst, architect, developer, tester, release manager
Related Methodologies, Best Practices and Tools	Waterfall
Related Technologies	

Project Using This Methodology						
	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	X
	Project 4		Project 8		Project 12	

M7.	**Methodology Name:**	**RUP@EC**							
Use		**Objectives**			**Benefits**				
Software development.		It is based on the Rational Unified Process developed by Rational Software Corporation [1]. It is a software development methodology tailored to the European Commission that uses an iterative and incremental approach.			This methodology proposes a list of artefacts to ensure the documentation and information of the development process.				
SDLC Phase Where It Is Used									
Analysis	Design		Development		Testing		Deployment		
X	X		X		X		X		
Roles (i.e. PM, developer, etc.)			Project manager, business analyst, architect, developer, tester, release manager						
Related Methodologies, Best Practices and Tools			RUP						
Related Technologies									
Project Using This Methodology	Project 1			Project 5			Project 9		
	Project 2	X		Project 6			Project 10		
	Project 3			Project 7			Project 11		
	Project 4			Project 8			Project 12		

3.2. Best Practices Used by the Analysed Projects During the Software Development Lifecycle

BP1. Best Practice Name:	**Security in Design Phase**	
Use	**Objectives**	**Benefits**
Security experts are involved in application design to prevent possible design flaws, or provide awareness of possible security risks.	Detect possible security flaws. Detect possible risks, according to the application architecture.	Apply mitigations by design of possible risks. Apply countermeasures of possible security vulnerabilities.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
X	X	N/A	N/A	N/A	N/A

Roles (i.e. Analyst, developer, tester)	(Security) Analyst, (Security) Architect
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1		Project 5	X	Project 9	
	Project 2		Project 6		Project 10	
	Project 3	X	Project 7	X	Project 11	
	Project 4	X	Project 8	X	Project 12	

BP2. Best Practice Name:	**Explicit Security Requirements**	
Use	**Objectives**	**Benefits**
Add explicit security requirements in the requirements gathering phase Enforce the design of security functionalities in the application.	Provide security mechanisms to enhance application security.	Apply countermeasures against possible application risks. Improve application security

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
X	X	N/A	N/A	N/A	N/A

Roles (i.e. Analyst, developer, tester)	(security) Analyst, (security) architect
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1		Project 5	X	Project 9	
	Project 2		Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	
	Project 4	X	Project 8	X	Project 12	

BP3. Best Practice Name:		**Role-based Authorisation**
Use	**Objectives**	**Benefits**
Authorisation based on roles that provides an easy way to grant privileges to users according to different user profiles. This allows centralized management of user privileges.	Centralization of user privileges according to user profiles.	Access to application resources is granted in a secure way. Users only have access to the resources defined in the user profile.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
X	X	X	N/A	N/A	N/A

Roles (i.e. Analyst, developer, tester)	Analyst, architect, developer
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1	X	Project 5		Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3		Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	

BP4. Best Practice Name:	**Standard Authentication Module**	
Use	**Objectives**	**Benefits**
Use a robust and tested authentication module that is based on common authentication protocols, instead of creating a custom one.	Secure user authentication. Avoid typical authentication attacks.	The authentication mechanism is secure, and only authorized users are able to access the application.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	X	X	N/A	N/A	N/A

Roles (i.e. Analyst, developer, tester)	Analyst, developer, architect
Related Methodologies, Best Practices and Tools	Alfresco security, Spring security, ECAS, CAS, Site Minder, Crowd
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6		Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP5. Best Practice Name:		**Standard Authorisation Module**
Use	**Objectives**	**Benefits**
Use a robust and tested Authorisation module, instead of creating a custom one.	Secure access control. Avoid typical privilege escalation.	The Authorisation mechanism is secure enough to protect application resources, enabling access only for users with corresponding privileges.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	X	X	N/A	N/A	N/A

Roles (i.e. Analyst, developer, tester)	Analyst, developer, architect
Related Methodologies, Best Practices and Tools	Alfresco security, Spring security
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1		Project 5		Project 9	X
	Project 2	X	Project 6		Project 10	X
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	X

BP6. Best Practice Name:			**Standard Cryptographic Module**		
Use		**Objectives**		**Benefits**	
Use a robust and tested cryptographic module, instead of creating a custom one.		Robust encryption algorithm implementation.		Secure application communications. Tested library for critical information management.	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	X	X	N/A	N/A	N/A
Roles (i.e. Analyst, developer, tester)			Analyst, Developer, Architect		
Related Methodologies, Best Practices and Tools			OpenSSL		
Related Technologies (i.e. Java)					

Projects Using This Best Practice	Project 1		Project 5		Project 9	
	Project 2		Project 6	X	Project 10	
	Project 3		Project 7		Project 11	X
	Project 4		Project 8		Project 12	

BP7. Best Practice Name:	**Well-tested Base Technology**	
Use	**Objectives**	**Benefits**
Use a robust and mature underlying technology for application development that has been tested in many production environments, and is widely use.	Minimize Zero-day vulnerability risks. Active development of the base technology gives place to new security functionalities and patches as needed.	Take advantage of the previous experiences regarding security of the underlying technology. Reducing the number of possible security flaws in the base technology.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	X	X	N/A	N/A	N/A

Roles (i.e. Analyst, developer, tester)	Analyst, developer, architect
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	Java , PHP

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP8. Best Practice Name:	**Security Awareness From External Sources**

Use	**Objectives**	**Benefits**
Many software vendors or FOSS communities provide information about security issues of the software they develop. Also public vulnerability repositories share information of security flaws. The project team should check this information to know of possible security risks for the application.	Be updated about security issues of software used by the application (components, application infrastructure).	Knowledge of possible security risks which allow the required preventive actions to be taken.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	X	X	N/A	N/A	X

Roles (i.e. Analyst, developer, tester)	Project Manager, system administrator
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1		Project 5		Project 9	X
	Project 2	X	Project 6		Project 10	X
	Project 3	X	Project 7	X	Project 11	
	Project 4	X	Project 8	X	Project 12	X

-BP9. Best Practice Name:	Continuous Testing	
Use	**Objectives**	**Benefits**
Release management	Execute automated tests as part of the software delivery pipeline.	The development team can prevent problems from progressing to the next stage of SDLC. Reduce the time and effort needed to fixing defects.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	X	X	N/A	N/A

Roles (i.e. Analyst, developer, tester)	Tester, operations
Related Methodologies, Best Practices and Tools	Continuous delivery, continuous deployment
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1	X	Project 5		Project 9	X
	Project 2	X	Project 6	X	Project 10	
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8		Project 12	X

BP10. Best Practice Name:	**Code Review**	
Use	**Objectives**	**Benefits**
Developed code is analysed in order to review possible security issues, bugs or standard non-compliant code, during development and testing phases.	Find possible application vulnerabilities, bugs or standard non-compliant code	Early bug and vulnerability finding. Improve the code quality.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	X	X	N/A	N/A

Roles (i.e. Analyst, developer, tester)	Tester, developer
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1		Project 5		Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8		Project 12	X

BP11. Best Practice Name:	**DevOps**	
Use	**Objectives**	**Benefits**
Release management	Development and operational teams are involved from the beginning. This practice ensures the alignment between these two teams to mitigate the risks when the deployment and integration are conducted	Technical benefits such as faster resolution of problems Mitigate the risks related to the release management phase

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	X	X	X	N/A

Roles (i.e. Analyst, developer, tester)	IT team
Related Methodologies, Best Practices and Tools	Continuous deployment, continuous delivery, agile
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7	X	Project 11	
	Project 4		Project 8		Project 12	

BP12. Best Practice Name:		**Use of Non-production Environments For Testing**			
Use		**Objectives**	**Benefits**		
Validation and testing		Software development or testing should not be done on the production environment.	Development and testing on test environments avoids interfering with the production and their users. It also avoids the risk if wrong or improper actions are performed, or the possibility of new bugs resulting from new functionalities		
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	X	X	X	N/A
Roles (i.e. Analyst, developer, tester)		Tester, operations			
Related Methodologies, Best Practices and Tools					
Related Technologies (i.e. Java)					

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP13. Best Practice Name:			**Initial Assessment Of Release Components**		
Use		**Objectives**		**Benefits**	
Validation and testing		Conduct an initial quality assessment of the release components.		This process ensures that the components meet the defined quality criteria to enter the testing phase	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	X	X	N/A	N/A
Roles (i.e. Analyst, developer, tester)			Developer, tester		
Related Methodologies, Best Practices and Tools			Validation and testing, unit testing		
Related Technologies (i.e. Java)					

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP14. Best Practice Name:	**Release Testing**	
Use	**Objectives**	**Benefits**
Validation and testing	Submitting the release components to intensive tests	This process ensures that only components which meet the quality criteria can be deployed to production

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	X	X	N/A

Roles (i.e. Analyst, developer, tester)	Developer, tester
Related Methodologies, Best Practices and Tools	Validation and testing, unit testing, integration testing, functional testing, non-functional testing
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP15. Best Practice Name:	**User Acceptance Testing**	
Use	**Objectives**	**Benefits**
Validation and testing	Submitting the release of tests performed by users or business stakeholders	Business stakeholders approve the quality and functionalities implemented in the release.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	X	X	N/A

Roles (i.e. Analyst, developer, tester)	Developer, tester
Related Methodologies, Best Practices and Tools	Validation and testing, acceptance testing
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP16. Best Practice Name:		**Automation Testing**			
Use		**Objectives**		**Benefits**	
Validation and testing		Execute automatically the written tests without manual intervention.		Increase effectiveness, efficiency and coverage of software testing	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	X	X	N/A
Roles (i.e. Analyst, developer, tester)			Developer, tester		
Related Methodologies, Best Practices and Tools			Validation and testing, unit testing, integration testing, functional testing, non-functional testing, regression testing		
Related Technologies (i.e. Java)					

Projects Using This Best Practice						
	Project 1		Project 5		Project 9	
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP17. Best Practice Name:	**Security Testing**	
Use	**Objectives**	**Benefits**
Validation and testing	Check that the release is free of known vulnerabilities.	Verification of these 6 principles: Confidentiality, Integrity, Authentication, Authorisation, Availability, Non-repudiation.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	X	X	N/A

Roles (i.e. Analyst, developer, tester)	Developer, analyst, tester
Related Methodologies, Best Practices and Tools	Penetration testing, vulnerability scan, black/white testing
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1	X	Project 5		Project 9	X
	Project 2		Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	
	Project 4		Project 8		Project 12	X

BP18. Best Practice Name:	**Third-party Testing**	
Use	**Objectives**	**Benefits**
Application testing conducted by an external team of security experts	Find possible security flaws and misconfigurations.	Detect security vulnerabilities in the application. Detect configuration error in the application and application environment.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	X	X	X

Roles (i.e. Analyst, developer, tester)	Security Auditor, system administrator.
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1	X	Project 5		Project 9	X
	Project 2		Project 6		Project 10	X
	Project 3	X	Project 7	X	Project 11	
	Project 4		Project 8	X	Project 12	X

BP19. Best Practice Name:	**Release Planning**	
Use	**Objectives**	**Benefits**
Release management	Plan and schedules releases and define their scope	The release is planned in advance.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	X	N/A

Roles (i.e. Analyst, developer, tester)	Project manager, operations
Related Methodologies, Best Practices and Tools	Release management
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	
	Project 2	X	Project 6		Project 10	X
	Project 3	X	Project 7	X	Project 11	
	Project 4	X	Project 8	X	Project 12	X

BP20. Best Practice Name:	**Security Incident Management.**	
Use	**Objectives**	**Benefits**
Security incident management is part of the security plan, where the instructions about how to respond to incidents are explained.	Have clearly defined response actions in case of security incidents, as well as a contact list of key staff.	This provides an effective mechanism so as to mitigate the possible impacts in case of a security incident.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	X	X

Roles (i.e. Analyst, developer, tester)	System administrator, project manager, CISO or LISO.
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1		Project 5	X	Project 9	
	Project 2	X	Project 6		Project 10	X
	Project 3		Project 7	X	Project 11	
	Project 4	X	Project 8	X	Project 12	

BP21. Best Practice Name:		**Proactive Problem Identification**			
Use		**Objectives**		**Benefits**	
Problem management		Improve the quality of the application, by identifying bugs and vulnerabilities in advance		It is possible to address these issues in advance, providing fixings or workarounds.	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X
Roles (i.e. Analyst, developer, tester)			IT Team		
Related Methodologies, Best Practices and Tools			ITIL [2], Problem management		
Related Technologies (i.e. Java)					
Related Technologies (i.e. Java)					

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP22. Best Practice Name:		**Problem Categorisation and Prioritisation**
Use	**Objectives**	**Benefits**
Problem management	Record and prioritise the problem in order to facilitate an effective resolution	Categorize the criticality of the incident to facilitate an effective resolution.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X

Roles (i.e. Analyst, developer, tester)	IT Team
Related Methodologies, Best Practices and Tools	ITIL [2], Problem management
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP23. Best Practice Name:			**Incident Management Support**		
Use		**Objectives**		**Benefits**	
Incident management		Provide and maintain the tools, channels, skills and rules for incident management		Effective and efficient handling of incidents	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X
Roles (i.e. Analyst, developer, tester)			IT Team		
Related Methodologies, Best Practices and Tools			ITIL [2], Incident management		
Related Technologies (i.e. Java)					

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP24. Best Practice Name:			**Incident Logging and Categorisation**		
Use		**Objectives**		**Benefits**	
Incident management		Record and prioritise the incident in order to facilitate the resolution		Categorize the critical of the incident to facilitate an effective resolution.	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X
Roles (i.e. Analyst, developer, tester)			Helpdesk, first level technicians		
Related Methodologies, Best Practices and Tools			ITIL [2], Incident management		
Related Technologies (i.e. Java)					

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP25. Best Practice Name:		**Immediate Incident Resolution By 1st Level Support**
Use	**Objectives**	**Benefits**
Incident management	First level to solve issues which are not related to bugs or vulnerabilities.	Most of users' operational issues are solved at this level

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X

Roles (i.e. Analyst, developer, tester)	Helpdesk, first level technicians
Related Methodologies, Best Practices and Tools	ITIL [2], Incident management
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP26. Best Practice Name:		**Incident Resolution By 2nd Level Support**
Use	**Objectives**	**Benefits**
Incident management	Solve issues that have been escalated from the first level support	Most configuration and technical issues not related to bugs or vulnerabilities are solved at this level

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X

Roles (i.e. Analyst, developer, tester)	Specialized technicians
Related Methodologies, Best Practices and Tools	ITIL [2], Incident management
Related Technologies (i.e. Java)	

Projects Using This Best Practice						
	Project 1		Project 5		Project 9	X
	Project 2	X	Project 6		Project 10	
	Project 3	X	Project 7		Project 11	X
	Project 4	X	Project 8		Project 12	

BP27. Best Practice Name:			**Handling of Major Incidents**		
Use		**Objectives**		**Benefits**	
Incident management		Address critical issues which cause serious interruptions		Address the issue and try to recover the service as soon as possible.	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	X
Roles (i.e. Analyst, developer, tester)			IT Team		
Related Methodologies, Best Practices and Tools			ITIL [2], Incident management		
Related Technologies (i.e. Java)					

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

BP28. **Best Practice Name:**		**Security Incident Notification**
Use	**Objectives**	**Benefits**
Security incidents are communicated to users to inform them about the impact and suggest some actions for containment or resolution (i.e. password change)	Effective communication with users on security issues Provide some security awareness information to users.	Users trust that security is managed correctly. Users can perform some mitigation actions in case of a security incident that affects them.

SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	N/A	x

Roles (i.e. Analyst, developer, tester)	Project Manager
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Best Practice	Project 1	X	Project 5	X	Project 9	X
	Project 2		Project 6	X	Project 10	
	Project 3	X	Project 7		Project 11	
	Project 4		Project 8		Project 12	

BP29. Best Practice Name:		**Pro-Active User Information**			
Use		**Objectives**		**Benefits**	
Incident management, Release management		Inform end-users about service interruptions		User can adjust themselves towards these interruptions	
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	X	X
Roles (i.e. Analyst, developer, tester)			Operations		
Related Methodologies, Best Practices and Tools			ITIL [2], Incident management, Release management		
Related Technologies (i.e. Java)					

Projects Using This Best Practice						
	Project 1	X	Project 5	X	Project 9	
	Project 2	X	Project 6		Project 10	
	Project 3		Project 7	X	Project 11	X
	Project 4		Project 8	X	Project 12	X

BP30. Best Practice Name:	**Continuous Delivery**				
Use	**Objectives**	**Benefits**			
Release management	Release software faster and more frequently by allowing more incremental updates to the application in production	Ensure reliable releases which can be deployed at any time. Reduces the Time To Market			
SDLC Phase Where It Is Used					
Analysis	Design	Development	Testing	Deployment	Maintenance
N/A	N/A	N/A	N/A	X	X
Roles (i.e. Analyst, developer, tester)	Operations				
Related Methodologies, Best Practices and Tools	Continuous deployment				
Related Technologies (i.e. Java)					

Projects Using This Best Practice						
	Project 1	X	Project 5		Project 9	X
	Project 2	X	Project 6	X	Project 10	
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8		Project 12	X

3.3. Tools Used by the Analysed Projects During the Software Development Lifecycle

T1	**Tool Name:**	**EMC Documentum xCP**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Development platform	A flexible development platform for automating complex, information-intensive processes to drive better business decisions	It improves productivity, intelligence and agility.	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	X	X	N/A	N/A

Roles (i.e. Analyst, developer, tester)	IT Team
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	Documentum

Projects Using This Tool						
	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8	X	Project 12	

T2	**Tool Name:**		**Jira**	
Use	**Objectives**	**Benefits**	**Licensing Type**	
Bug tracking system Project management software	It provides bug tracking, issue tracking, and project management functions.	Connection to all the developer tools that it uses, making it the single source of truth for every step in their projects.	Atlassian	
SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X
Roles (i.e. IT Team, user...)		IT Team		
Related Methodologies, Best Practices and Tools		PM2 and development methodology		
Related Technologies (i.e. Java)		JAVA		

Projects Using This Tool	Project 1	X	Project 5	X	Project 9	X
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8	X	Project 12	X

T3	**Tool Name:**	**Yammer**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Private communications within organizations	It is a freemium enterprise social networking service used for private communication within organizations.	It helps employees collaborate across departments, locations, and business apps.	Microsoft

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	N/A	N/A

Roles (i.e. IT Team, user...)	All stakeholders
Related Methodologies, Best Practices and Tools	Deployment communication
Related Technologies (i.e. Java)	

Projects Using This Tool						
	Project 1	X	Project 5		Project 9	X
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8	X	Project 12	X

T4	**Tool Name:**	**Documentum**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Enterprise content management platform	It provides management function capabilities for all types of content	Controls access to the repository and improves compliance through comprehensive authentication, authorisation and auditing.	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	IT Team
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Tool	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8	X	Project 12	

T5	**Tool Name:**	**Crowd**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Identity Management tool.	It provides single sign on and user identity capabilities	Can be integrated with several user repositories	Atlassian

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	N/A	X

Roles (i.e. IT Team, user...)	End-User, IT Team
Related Methodologies, Best Practices and Tools	Single Sign On
Related Technologies (i.e. Java)	

Projects Using This Tool						
	Project 1	X	Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

T6	**Tool Name:**	**GitHub**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Web-based Git repository hosting service	Distributed revision control and source code management functionality of Git.	It provide a central repository where all developers can push and pull their changes to and from that repository	Several Licenses

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	X	X	X

Roles (i.e. IT Team, user...)	
Related Methodologies, Best Practices and Tools	N/A
Related Technologies (i.e. Java)	RUBY

Projects Using This Tool	Project 1	X	Project 5		Project 9	
	Project 2		Project 6		Project 10	X
	Project 3		Project 7	X	Project 11	
	Project 4		Project 8	X	Project 12	

T7	**Tool Name:**	**Confluence**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Team collaboration software	Organize work, create documents, and discussion board in one place	It has been adapted to work with Jira and other Atlassian Software such as Bamboo.	Atlassian

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	
Related Methodologies, Best Practices and Tools	JIRA, BAMBOO
Related Technologies (i.e. Java)	JAVA

Projects Using This Tool	Project 1	X	Project 5	X	Project 9	
	Project 2	X	Project 6		Project 10	X
	Project 3		Project 7	X	Project 11	
	Project 4		Project 8		Project 12	X

T8	**Tool Name:**	**SVN**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Code repository	Mainly used to manage versions and branches of the source code	It provides atomic commits, fast and flexible update/commits, and it ease of integration.	Apache 2.0

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	X	X

Roles (i.e. IT Team, user...)	Developer, Tester
Related Methodologies, Best Practices and Tools	Source code versioning
Related Technologies (i.e. Java)	

Projects Using This Tool	Project 1	X	Project 5	X	Project 9	
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7		Project 11	
	Project 4	X	Project 8		Project 12	X

T9	**Tool Name:**	**Nexus**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Libraries repository	Manages libraries used within development projects and monitors their usage, inspects security and license issues affecting the components	It provides an essential infrastructure for component-based software development and continuous delivery	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	X	X	X

Roles (i.e. IT Team, user...)	Developer, architect, operations
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Tool	Project 1	X	Project 5		Project 9	
	Project 2	X	Project 6	X	Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

T10	Tool Name:	CruiseControl	
Use	**Objectives**	**Benefits**	**Licensing Type**
Continuous integration	Automates builds, tests, and releases	Deployment automation Test results reporting	BSD-style license

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	X	X

Roles (i.e. IT Team, user...)	Developer, tester, IT operations
Related Methodologies, Best Practices and Tools	Continuous integration, Jenkins, Bamboo, Nexus, SVN, GitHub
Related Technologies (i.e. Java)	JAVA

Projects Using This Tool						
	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3	X	Project 7		Project 11	
	Project 4		Project 8		Project 12	

T11	**Tool Name:**	**Jenkins**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Continuous integration	Automates builds, tests, and releases	Deployment automation Test results reporting	Mit and Creative Commons

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	X	X

Roles (i.e. IT Team, user...)	Developer, tester, IT operations
Related Methodologies, Best Practices and Tools	Continuous integration, Cruise control, Bamboo, Nexus, SVN, GitHub
Related Technologies (i.e. Java)	JAVA

Projects Using This Tool	Project 1		Project 5		Project 9	
	Project 2	X	Project 6	X	Project 10	
	Project 3	X	Project 7		Project 11	
	Project 4	X	Project 8		Project 12	X

T12	**Tool Name:**	**Bamboo**					
Use	**Objectives**	**Benefits**	**Licensing Type**				
Continuous integration server and delivery tool.	Automated builds, tests, and releases in a single workflow.	Supports builds in any programming language.	Atlassian				
SDLC Phase Where It Is Used							
Analysis	Design	Development	Testing	Deployment			
N/A	N/A	N/A	X	X			
Roles (i.e. IT Team, user...)	Developer, tester, IT operations						
Related Methodologies, Best Practices and Tools	Continuous integration, Jenkins, Cruise control, Nexus, SVN, GitHub						
Related Technologies (i.e. Java)	Java.						
Projects Using This Tool	Project 1	X	Project 5		Project 9		
	Project 2		Project 6		Project 10		
	Project 3		Project 7	X	Project 11		
	Project 4		Project 8		Project 12		

T13	**Tool Name:**	**IBM Rational**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Application Lifecycle Management Solution	Software development management	Provides solutions to deliver requirements management, quality management, change and configuration management and project planning and tracking capabilities on a single platform	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	IT Team
Related Methodologies, Best Practices and Tools	Application lifecycle management, software development lifecycle, requirements management, quality management, change and configuration management and project planning and tracking
Related Technologies (i.e. Java)	

Projects Using This Tool	Project 1		Project 5		Project 9	
	Project 2	X	Project 6		Project 10	
	Project 3		Project 7		Project 11	X
	Project 4		Project 8		Project 12	

T14	**Tool Name:**	**Selenium**				
Use	**Objectives**	**Benefits**	**Licensing Type**			
Software testing framework for web applications	Functional, integration and regression testing	Automated testing	Apache 2.0			
SDLC Phase Where It Is Used						
Analysis	Design	Development	Testing	Deployment		
N/A	N/A	N/A	X	N/A		
Roles (i.e. IT Team, user...)	Developer, tester					
Related Methodologies, Best Practices and Tools	Functional testing, regression test, test automation					
Related Technologies (i.e. Java)	Web Technologies					
Projects Using This Tool	Project 1		Project 5		Project 9	
	Project 2	X	Project 6		Project 10	
	Project 3		Project 7		Project 11	X
	Project 4		Project 8	X	Project 12	X

T15	Tool Name:	Crucible		
Use	**Objectives**	**Benefits**	**Licensing Type**	
Peer code review	It provides a particularly tailored to distribution teams and facilitates asynchronous review and commenting on code	Peer code review increments the quality of the code since it is revised by other developers or by the QA team	Proprietary	
SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	X	X	X
Roles (i.e. IT Team, user...)	Developer, QA			
Related Methodologies, Best Practices and Tools	Code review			
Related Technologies (i.e. Java)				

Projects Using This Tool	Project 1		Project 5		Project 9	
	Project 2	X	Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

T16	**Tool Name:**	**CA Clarity**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Project management	Manage project related aspects like costs, planning and general project management.	Projects deliver desired results in line with market needs and business strategies. Manage all financial aspects of your portfolio with accountability	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	Project manager
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Tool						
	Project 1		Project 5	X	Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

T17	Tool Name:	SharePoint	
Use	**Objectives**	**Benefits**	**Licensing Type**
CMS (Content Management Systems)	SharePoint combines various functions which are traditionally separate applications: intranet, extranet, content management, document management	Share and publish documents, writings and publications easily	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	End-user, IT Team
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	

Projects Using This Tool	Project 1		Project 5	X	Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

T18	**Tool Name:**	**Piwik**

Use	**Objectives**	**Benefits**	**Licensing Type**
Web analytics	It tracks online visits to one or more websites and displays reports on these visits for analysis		GNU GPL v3

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	End-user, IT Team
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	PHP, MySQL

Projects Using This Tool						
	Project 1		Project 5		Project 9	X
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

3.4. Libraries and Building Blocks Used by the Analysed Projects During the Software Development Lifecycle

LB&B1	**Library Name:**	**Alfresco Security**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Authentication and Authorisation	It comprises a combination of authentication and Authorisation.	An internal, password-based, authentication implementation Support to integrate with many external authentication environments The option to write your own authentication integration and to use several of these options simultaneously	Lesser GNU Public License V3 (LGPLv3)

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. Analyst, developer, tester)	
Related Methodologies, Best Practices and Tools	
Related Technologies (i.e. Java)	LDAP, JAVA

Projects Using This Library	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	X

LB&B2	**Tool Name:**	**CAS**	
Use	**Objectives**	**Benefits**	**Licensing Type**
Multiprotocol Web single sign-on	Single Sign On tool that can be integrated with a number of supported authentication mechanisms including LDAP/Active Directory, Kerberos, and RDBMS	Provides Single Sign On capabilities	Apache 2.0

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
X	X	X	X	X

Roles (i.e. IT Team, user...)	End-user, IT team
Related Methodologies, Best Practices and Tools	Single Sign On
Related Technologies (i.e. Java)	ECAS

Projects Using This Library	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7	X	Project 11	
	Project 4		Project 8		Project 12	

LB&B31	Tool Name:	ECAS	
Use	**Objectives**	**Benefits**	**Licensing Type**
European Institutions Authorisation system based on CAS	Single Sign on tool that can be integrated to a number of supported authentication mechanisms including LDAP/Active Directory, Kerberos, and RDBMS	Provides Single Sign On capabilities	EUPL

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	N/A	X

Roles (i.e. IT Team, user...)	End-user, IT Team
Related Methodologies, Best Practices and Tools	Single Sign On
Related Technologies (i.e. Java)	CAS, Java

Projects Using This Library	Project 1		Project 5	X	Project 9	X
	Project 2	X	Project 6		Project 10	X
	Project 3	X	Project 7	X	Project 11	X
	Project 4	X	Project 8		Project 12	X

LB&B4	Tool Name:	E-Signature	
Use	**Objectives**	**Benefits**	**Licensing Type**
Electronic Authentication	Demonstrate the authenticity of a digital message or document	Provides a level of assurance that the message or document was created by a known person	N/A

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	N/A	N/A

Roles (i.e. IT Team, user...)	End-User
Related Methodologies, Best Practices and Tools	N/A
Related Technologies (i.e. Java)	N/A

Projects Using This Library	Project 1		Project 5	X	Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8	X	Project 12	

LB&B5	Library Name:	OpenSSL	
Use	**Objectives**	**Benefits**	**Licensing Type**
Security Library	Software library to be used in applications that need to secure communications against eavesdropping or need to ascertain the identity of the party at the other end	It implements basic cryptographic functions	Apache license 1.0 and four-clause BSD License

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	N/A	X

Roles (i.e. Analyst, developer, tester)	
Related Methodologies, Best Practices and Tools	Encryption
Related Technologies (i.e. Java)	

Projects Using This Library						
	Project 1		Project 5		Project 9	
	Project 2		Project 6	X	Project 10	
	Project 3		Project 7		Project 11	X
	Project 4		Project 8		Project 12	

LB&B6	**Tool Name:**	**Site Minder**	
Use	**Objectives**	**Benefits**	**Licensing Type**
User Authentication and single sign-on.	It is usually connected to a LDAP so this integration is also able to import users from LDAP.	Authentication with Screen Name	Commercial

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	N/A	N/A	N/A	X

Roles (i.e. IT Team, user...)	End-User, IT Team
Related Methodologies, Best Practices and Tools	Single Sign On
Related Technologies (i.e. Java)	

Projects Using This Library						
	Project 1		Project 5		Project 9	
	Project 2		Project 6		Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8	X	Project 12	

LB&B7	**Library Name:**	**Spring Security**

Use	**Objectives**	**Benefits**	**Licensing Type**
Authentication and access control framework	It is a framework that focuses on providing both authentication and authorisation to Java applications.	Provides authentication, authorisation and other security features for enterprise applications	Apache 2.0

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	X	X	X	X

Roles (i.e. Analyst, developer, tester)	End-user, IT team
Related Methodologies, Best Practices and Tools	Authorisation, authentication
Related Technologies (i.e. Java)	JAVA/JAVA EE

Projects Using This Library						
	Project 1		Project 5		Project 9	
	Project 2	X	Project 6	X	Project 10	
	Project 3		Project 7		Project 11	
	Project 4		Project 8		Project 12	

3.5. Programming languages used by the analysed projects for software development

LG1	**Language Name:**	**Java**

Use	**Objectives**	**Benefits**	**Licensing Type**
It is a general-purpose software programming language.	Provides concurrent, class-based, object oriented, and is specifically designed to have a few implementation dependencies as possible.	Java code can run on all platforms that support Java without the need for recompilation	Oracle America INC.

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	X	X	X	X

Roles (i.e. Analyst, developer, tester)	Developer, tester, analyst, architect
Related Methodologies, Best Practices and Tools	Object Oriented
Related Technologies (i.e. Java)	

Projects Using This Programming Language	Project 1	X	Project 5	X	Project 9	
	Project 2	X	Project 6	X	Project 10	X
	Project 3	X	Project 7		Project 11	X
	Project 4		Project 8	X	Project 12	X

LG2	**Language Name:**	**PHP**	
Use	**Objectives**	**Benefits**	**Licensing Type**
General-purpose software programming language.	Programming language originally designed to create web sites	Flexible and powerful specially to create web sites	PHP

SDLC Phase Where It Is Used				
Analysis	Design	Development	Testing	Deployment
N/A	X	X	X	X

Roles (i.e. Analyst, developer, tester)	Developer, tester, analyst, architect
Related Methodologies, Best Practices and Tools	Object Oriented
Related Technologies (i.e. Java)	

Projects Using This Programming Language	Project 1		Project 5		Project 9	X
	Project 2		Project 6		Project 10	
	Project 3		Project 7	X	Project 11	
	Project 4	X	Project 8		Project 12	

4. Analysis of identified software development methodologies used in the European Institutions

Deliverable 1 provided a list of 15 projects to be analysed. Out of the 15 projects, 11 were from the European Commission and the remaining 4 from the European Parliament. The information gathering was conducted in interviews and a total of 14 out of 15 projects were covered during the interview rounds; thus, the analysis is based on the responses from the interviewees of these projects.

4.1. Project Management

4.1.1. Methodologies

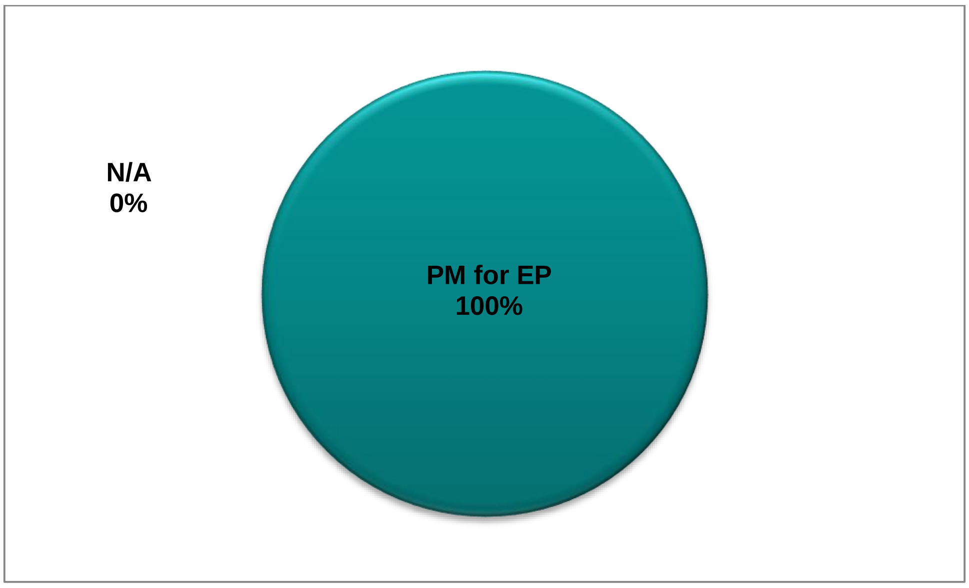
This point shows the different methodologies used for project management (based on the 14 analysed projects).

Table 4-1 Analysis of identified software development methodologies used in the European Institutions – Project management methodology

Project Management methodology	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
PM2	10	71%
PM for EP	3	21%
N/A	2	14%

The table shows that most of the projects analysed use a specific project management methodology, either PM2 or PM for EP. The relationship between the project management methodologies and the European Institutions (EC, EP) is depicted in the following figures.

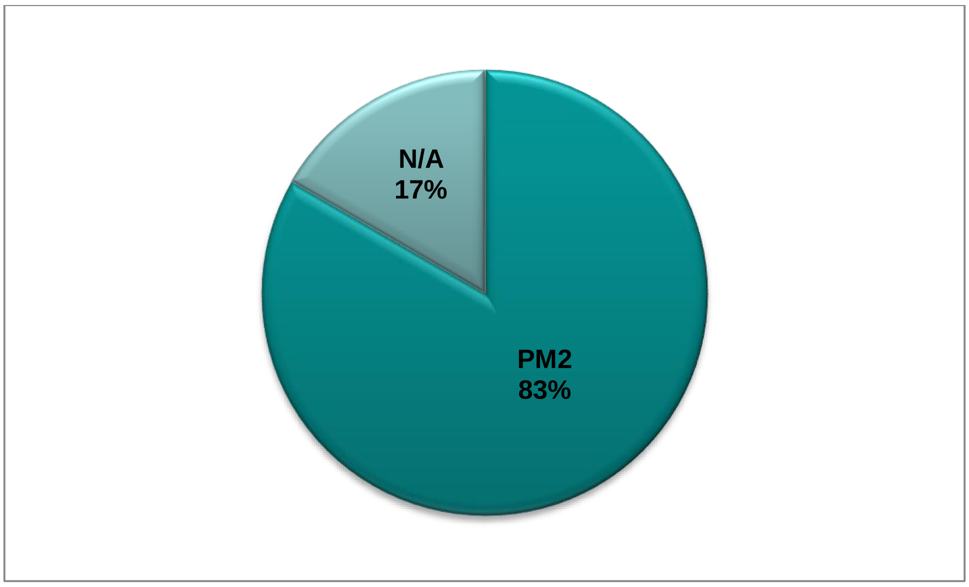
Figure 2. Analysis of identified software development methodologies used in the European Institutions – Usage of PM for EP methodology in the European Parliament



Project Management methodology	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
PM for EP	3	100%
N/A	0	0%

This chart shows that all the projects analysed from the European Parliament use PM for EP as a project management methodology.

Figure 3. Analysis of identified software development methodologies used in the European Institutions – Usage of PM2 methodology in the European Commission



Project Management methodology	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
PM2	10	83%
N/A	2	17%

This chart shows that the majority of the European Commission projects analysed follows the PM2 methodology for project management.

4.1.2.Tools

- Jira
- Clarity

4.1.3.Conclusion

Most of the analysed European Institutions take advantage of the project management methodologies developed by them (PM2 and PM for EP). Since both methodologies are PMBOK based, they are similar in nature, and as such, they ensure that project management is well covered in the projects analysed

4.2.Software Development Lifecycle

4.2.1.Software Development Lifecycle Methodologies

4.2.1.1.Methodologies

In this point we will analyse the software development methodologies used within the analysed projects. The use of these methodologies helps the development team to follow structured processes to manage and produce high quality software products.

The following table shows the methodologies used in the projects analysed.

Table 4-2 Analysis of identified software development methodologies used in the European Institutions – Software development methodologies (Detail)

Software Development Methodology	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Scrum	7	50%
Custom Scrum	5	36%
Kanban	3	21%
Agile@EC	2	14%
Waterfall	1	7%
RUP@EC	1	7%

As it is showed in the table, the majority of the projects use agile methodologies, only one project uses Waterfall and one project uses RUP@EC. Among the agile methodologies, the most widely used is Scrum, in 50% of the projects, followed by custom methodologies based on Scrum. Three projects complement Scrum with the Kanban methodology.

The previous table identifies the high level software development methodologies used by the projects of this study. In order to manage this information, they have been grouped, as shown in Figure 4.

Figure 4. Analysis of identified software development methodologies used in the European Institutions – Methodologies

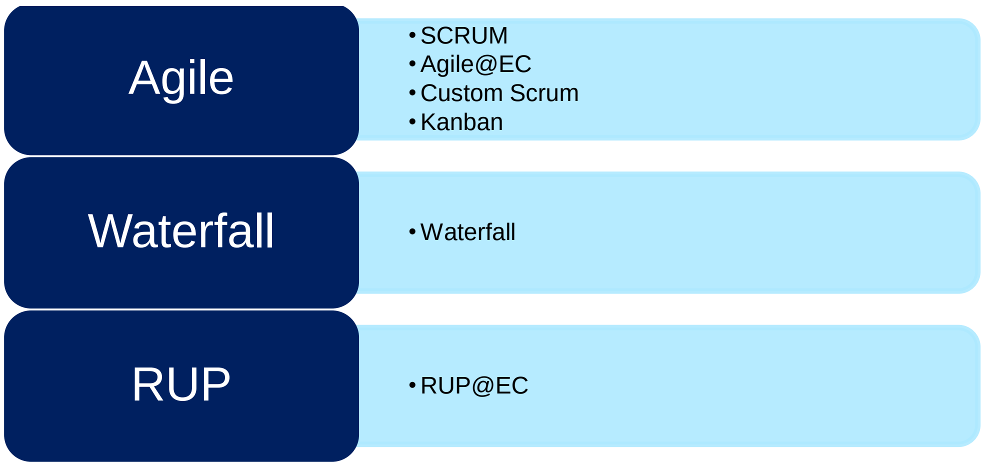


Table 4-3 Analysis of identified software development methodologies used in the European Institutions – Software development methodologies

Software Development Methodology	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Agile	13	93%
Waterfall	1	7%
RUP	1	7%

As it can be seen in the table, most of the projects analysed follow the agile approach, whereas only one project uses waterfall and another one uses RUP.

The following table shows the approach followed with regard to the development lifecycle according to the identified methodologies.

Table 4-4 Analysis of identified software development methodologies used in the European Institutions – Software development methodologies (cycle)

Software development cycle	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Iterative	13	93%
Waterfall	1	7%

Considering that RUP and Agile are iterative cycle methodologies, only one of the projects analysed uses a waterfall approach.

4.2.1.2. Tools

- Atlassian Jira
- Atlassian Confluence
- IBM Rational

4.2.1.3. Conclusion

As it has been shown in all the charts and tables analysed, the majority of the projects use an agile methodology for software development. Since agile is a methodology that delivers new functionalities in an iterative and incremental way, we can conclude that most of the projects analysed are software delivery oriented. This approach simplifies steps, formalities and artefacts, in comparison with the more procedural approaches of methodologies like Waterfall or RUP.

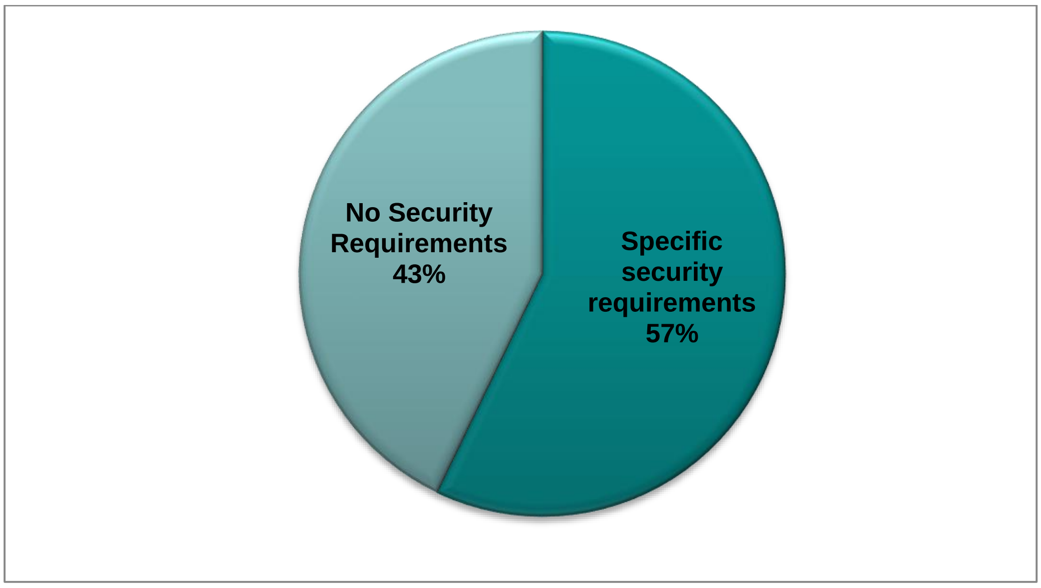
The analysis had to consider that some approaches are agile but not supported by applying a complete methodology like Scrum or Kanban, 36% of the projects analysed use a reduced part of these methodologies, thus, some parts of the software development lifecycle could not be well covered for these projects.

4.2.2. Security Requirements

4.2.2.1. Security Requirements Definition

Frequently software development is focused in the implementation of the functionalities which are requested by the business owners. However, from a security perspective, an attacker would be more interested on what the application will allow him to do (not necessarily related to the functionalities) and thus use it for his own benefit. Following OWASP [2] best practices, it is recommended to conduct an analysis of the security requirements in the early stages of the software development lifecycle, mainly during the design and analysis phases. This point analyses how these requirements are fulfilled in the projects analysed.

Figure 5. Analysis of identified software development methodologies used in the European Institutions – Security requirements



Security requirements	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Specific security requirements	8	57%
No security requirements	6	43%

A bit more of the half of the analysed projects use specific security requirements gathered from a security definition phase at the beginning of the project. For the remaining projects, security requirements are defined within business requirements, so they are not explicitly addressed.

The study also shows that 43% of the projects have a security plan.

4.2.2.2. Conclusion

This section shows that only half of the projects analysed take into consideration the requirements from the security point of view. Some IT teams from analysed projects consider that the security requirements could be more relaxed if the application is not exposed to the internet, at this point it is necessary to consider that an application could be attacked not only from the internet, but also from within the intranet.

4.2.3. Testing and Validation

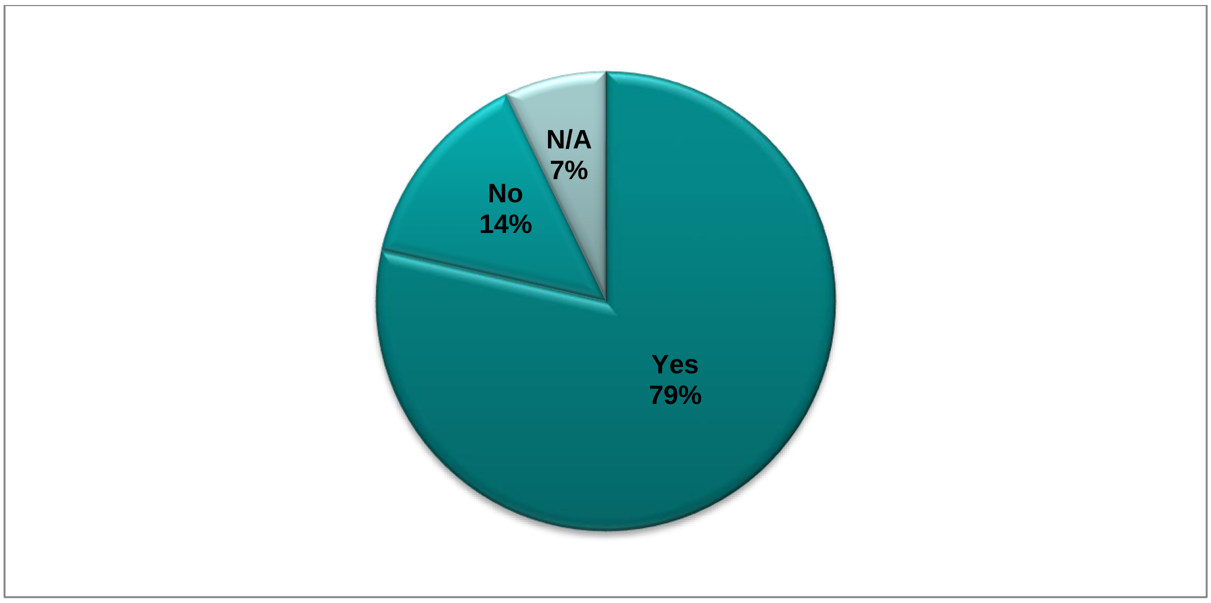
Testing is performed by DIGIT test centre in order to find possible bugs or vulnerabilities before the application is deployed in the production environment, and also to ensure the quality of the product and the implementation of the expected requirements. One main step of this phase is to receive the validation from the customer.

All the projects analysed conduct tests, but it is important to analyse how they are being conducted

4.2.3.1. Automatic Testing

Automatic testing allows the execution of written test, mostly in silent mode, without the manual intervention of the development or QA teams. Frequently this approach is based on suites or groups of individual tests which are logically-related, incremental, and repeatable. This analysis considers functional, non-functional, unit and regression tests.

Figure 6. Analysis of identified software development methodologies used in the European Institutions – Automatic tests



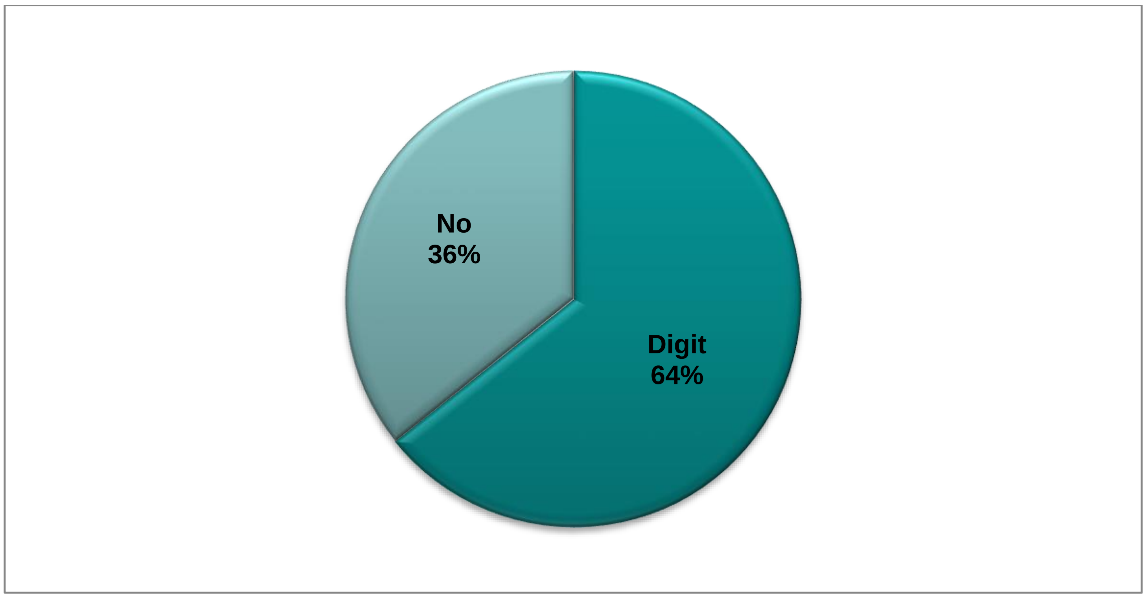
Perform automatic tests	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Yes	11	79%
No	2	14%
N/A	1	7%

The chart shows that most of the analysed projects use automatic testing for unit, functional, non-functional and regression test.

4.2.3.2. *Security testing*

This point analyses if projects execute tests to identify security bugs or vulnerabilities. These tests use techniques like penetration test, vulnerability scan and black and/or white box testing.

Figure 7. Analysis of identified software development methodologies used in the European Institutions – Automatic security tests



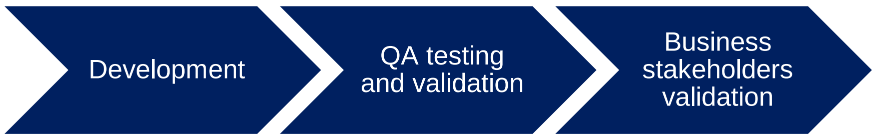
Perform automatic security tests	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Digit	9	64%
No	5	36%

As it is represented by the chart, more than half of the projects analysed use the security vulnerability testing service provided by Digit, the rest do not perform vulnerability testing for their applications.

4.2.3.3. *Validation Testing*

Most of the projects analysed perform validation tests that are conducted by the business stakeholders in order to validate the application's functionalities. These tests are performed once the QA team validates the tests conducted by the IT team. The sequence is as follows.

Figure 8. Analysis of identified software development methodologies used in the European Institutions – Validating testing flow



The analysed projects use a special environment to perform these validation tests. These environments are analysed in section 4.2.4.1

4.2.3.4. *Tools and methods*

- Selenium
- JUnit
- Continuous deployment
- Security tools used by the Digit cyber-security team[1]

4.2.3.5. *Conclusion*

This point shows that most of the analysed projects perform automatic tests. Although some tests cannot be automated, it is a good practice to perform complete and repeatable tests automatically, following the ITIL [2] "Service Validation and Testing stack" as a best practice:

- **Release Component Acquisition**: It is covered when the developers execute initial tests after new code is developed.
- **Release Test**: It is covered by regression, functional, non-functional, integration and unit tests which are executed within the test environment (this environment is covered in point 4.2.4.1.)

[1] Tools used by this department are not within the scope of this analysis.

- **Service Acceptance Testing**: It is covered by the business stakeholders, who execute acceptance tests within the acceptance environment (this environment is covered in point 4.2.4.1).

Although most of the projects analysed conduct testing, not all the projects do security testing, which could help mitigate the risk of vulnerabilities

It is important to highlight that some IT teams believe that it is not necessary to perform security test when the application is deployed within the intranet, not taking into consideration the fact that attacks can be triggered by insiders.

4.2.4.Release Management.

It is the process of managing a software build through different stages and environments, including the testing (4.2.3) and deployment (4.2.4.1).

One of the main objectives of the projects that use Agile is to increase the frequency of releases to reduce the Time to Market. At this point, the concept continuous delivery influences how the transitions from development to productions are.

4.2.4.1.Deployment

Deployment is the phase where the binaries of the software are installed in the servers in order to expose the functionalities of the application.

Regarding the deployment practices preferred in the European Institutions, we should differentiate two types of environments:

- Test and Acceptance / Pre-production: Where the application is deployed to be tested and accepted by the team involved in the project (IT team and business stakeholders), before the deployment in the production environment.
- Production: Where the application is deployed to be used by end users.

In our analysis most of the projects and developments split the environments into test, acceptance/pre-production and production following the pipeline shown in Figure 9.

Figure 9. Analysis of identified software development methodologies used in the European Institutions – Delivery pipeline

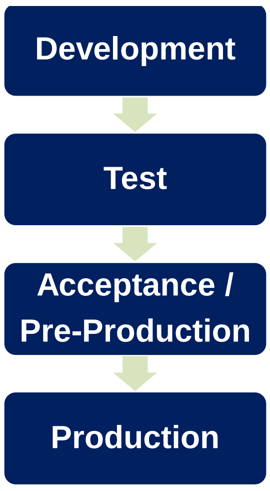


Table 4-5 Analysis of identified software development methodologies used in the European Institutions – Delivery pipeline

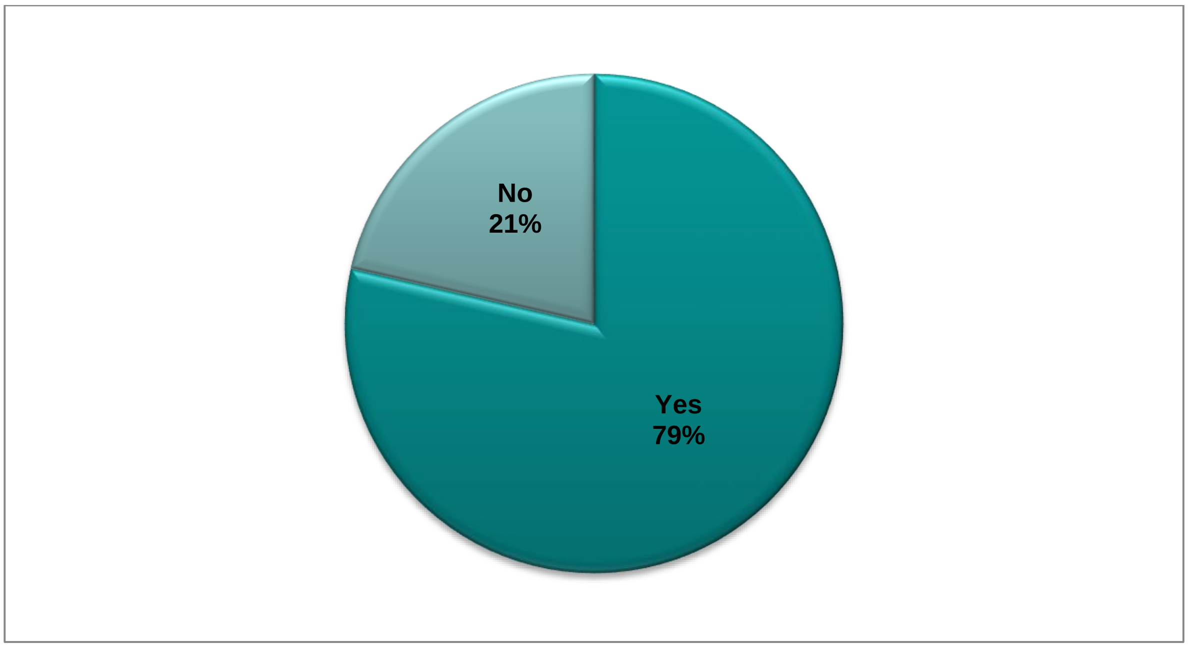
Environment	Description
Development	Developers use this environment to work and perform initial development tests of the application.
Test	Test the application to ensure the quality of the product before the deployment in the Acceptance environment. In some projects this acts as integration environment.
Acceptance / Pre - production	It is the environment before production. Once tests in the Test environment are successful, the version is promoted to this environment, where business stakeholders test the functionalities according to the original requirements. This environment should be as similar as possible to the production environment to mitigate the risks associated with promoting to the Production environment.
Production	When the acceptance/pre-production version is approved, it is promoted to this environment in order to offer the functionalities of the application to end users. This is the environment that manages the production data or “real” of the application. For the majority of the projects the promotion to this environment is performed by an operational team, commonly an external team in charge of the systems layer.

To deploy and promote versions between environments, continuous deployment is a good practice and it is widely used within the projects analysed. This practice allows more frequent deployments which in turn helps to find bugs or vulnerabilities faster.

Most of the projects use continuous deployment as a trigger to execute automatic regression, integration and unit tests. Additionally, some projects use tools for automatic code review to ensure the quality of the code.

The following chart depicts the usage of continuous deployment.

Figure 10. Analysis of identified software development methodologies used in the European Institutions – Continuous deployment



Continuous deployment	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Yes	11	79%
No	3	21%

4.2.4.2. Continuous testing and validation

Continuous testing is the process of executing automated tests as part of the delivery pipeline. The focus is on receiving continuous feedback on the business risks related to a software release candidate and determining if the software is ready to be promoted through the delivery pipeline. During this process, functional and non-functional tests (4.2.3.1), static code analysis (4.2.5.1) and security testing (4.2.3.2) can be involved.

The combination of figures 6 and 10 shows that all the analysed projects which implement automatic testing also use continuous deployment, which means that they take advantage of continuous testing.

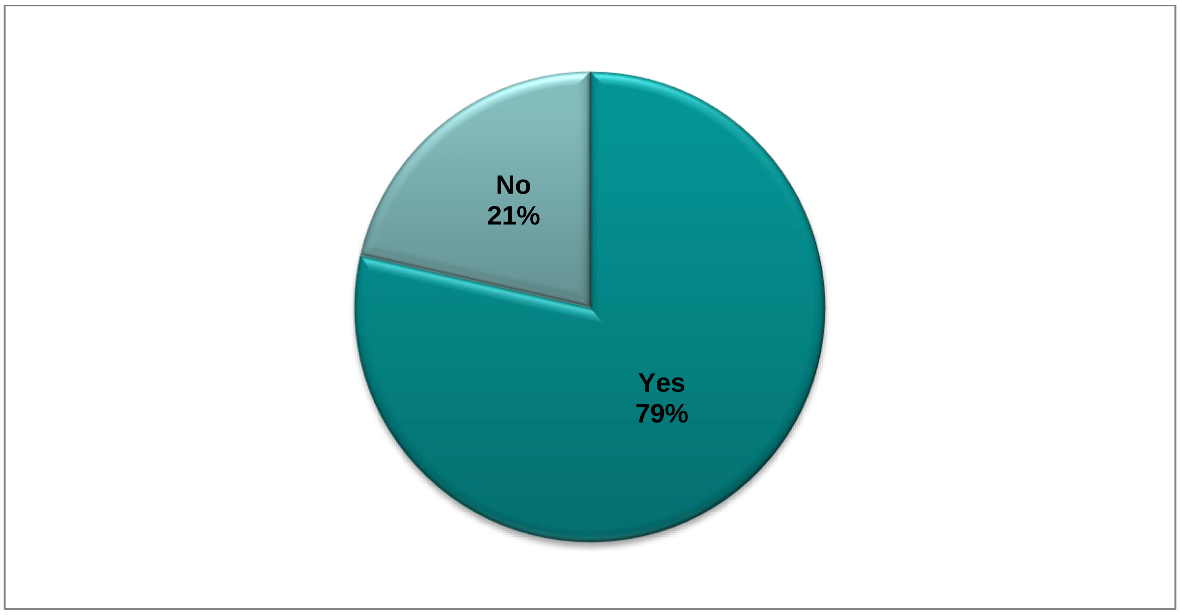
4.2.4.3. DevOps

This practice is based on agile methodologies in which the development and operational teams are involved from the beginning. This practice ensures the alignment between these two teams to mitigate the risks when the deployment is performed. Only one of the projects analysed uses this methodology since their development and operational teams are internal and work closely.

4.2.4.4. Release Planning

This point analyses if the projects use a roadmap for planning new releases. Additionally, we will analyse if the roadmap includes security aspects.

Figure 11. Analysis of identified software development methodologies used in the European Institutions – Roadmap definition



Roadmap definition	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Yes	11	79%
No	3	21%

As it is depicted in the chart, 79% of the analysed projects use a roadmap to plan future features and updates. Taking into account these projects, 27% include security aspects.

4.2.4.5. Channels and tools used

All the tools used for continuous deployment are open source. The usage of each one is showed in the table below.

Table 4-6 Analysis of identified software development methodologies used in the European Institutions – Continuous delivery tools

Continuous delivery tool	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Jenkins	5	36%
Bamboo	4	29%
Cruise Control	2	14%

The majority of the projects use Jenkins as continuous deployment tool, followed by Bamboo and only two projects use Cruise Control.

In order to communicate service interruptions derived from the deployment in production, some analysed projects communicate these interruptions to end-users using the tools listed in section 4.3.1.5.

4.2.4.6. Conclusion

This point shows that most of the analysed projects have a well-defined deployment pipeline to promote releases between environments. Additionally, most of the projects analysed take advantage of the continuous delivery best practice, which ensures the rapid identification of bugs and vulnerabilities.

There are two projects that apply the DevOps methodology. This methodology mitigates the problems between the development and operations teams when the release has to be deployed into the servers. These problems are solved by both teams working closely in all phases of the project following an agile approach.

Taking into consideration ITIL [2] Release Management as a best practice, the following processes are covered:

- **Release Planning:** It is covered by the roadmap for the planning of the releases.
- **Release Build:** It is covered by the automatic deployment performed by continuous integration servers.

- **Release Deployment:** Most of the deployments in production are conducted by an external operations team within Digit that is in charge of the infrastructure administration.

4.2.5. Inspection and code review

4.2.5.1. Code review

This point introduces the teams in charge of performing code reviews for the analysed projects.

Table 4-7 Analysis of identified software development methodologies used in the European Institutions – Code review teams

Code review team	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Development	6	43%
Internal but different from development	4	28%
No	3	21%
External	2	14%
FOSS Communities	1	7%

The majority of the projects rely on the development team for the code review, as they inspect the code produced by another developer (peer-to-peer review) or by using automatic tools like SonarQube. These are followed by projects which use different teams to perform code review. Two projects use external resources to do the code review, while another one uses the code that has been reviewed by the FOSS communities. It is worth noticing that three of the analysed projects do not conduct any code reviews.

4.2.5.1.1. Tools

- SonarQube
- Crucible
- SVN
- GitHub

4.2.5.2. Projects reviewed by security experts

This point shows who reviews the security of the projects developed.

Table 4-8 Analysis of identified software development methodologies used in the European Institutions – Security experts review

Security experts team	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Digit	8	57%
External	4	29%
No	3	21%
Internal	2	14%

Most of the projects developed by the European Commission are submitted to a service offered by Digit to review and execute automatic tests to ensure the security of the projects. 29% of the projects use specialised external services in order to check the security of the project. 14% use an internal expert and 21% do not get an expert to perform a security review.

4.2.5.3.Phase where the project is reviewed by security experts

This point shows in what phase the project is reviewed by security experts. The following table shows when the security is reviewed, in the case of those projects that perform security reviews.

Table 4-9 Analysis of identified software development methodologies used in the European Institutions – Security review phase

Security review phase	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
At the end (test, deployment)	6	54%
In all phases	2	18%
At the beginning (analysis, design)	2	18%
During development	1	9%

A good approach is to take security into consideration in all phases, with more emphasis at the beginning of the project. It is worth mentioning that 45% of the projects take security into consideration at the beginning of the project. However, the majority conduct the security review after the development phase.

4.2.5.4. *Conclusion*

Code review ensures the quality of the code and helps to find errors which could turn into bugs or vulnerabilities in the future. A good approach is to assign the review to a team of reviewers that are not involved in the development of such code.

On the other hand, although the projects are reviewed by security experts, they concentrate on conducting only security tests. Following the recommendations from OWASP [2], a good approach is to perform a code review that also focuses on security, and to conduct security reviews during all phases of the development lifecycle, focusing on the early phases.

4.2.6. *Application authentication and authorisation*

This section analyses which methods of authorisation and authentication are implemented in the projects analysed.

4.2.6.1. *Authentication*

This point analyses the methods and tools used to conduct the authentication in the projects.

Table 4-10 Analysis of identified software development methodologies used in the European Institutions – Authentication methods

Authentication method	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
ECAS	11	79%
CAS	2	14%
Biometrics	1	7%
CA Site Minder	1	7%
Crowd	1	7%
Others	1	7%

Most of the analysed projects use a CAS solution (since ECAS is based on CAS), one project uses advanced security systems like biometric authentication, another one uses CA Site Minder and another one use Crowd. This analysis shows that most of the applications use Single Sign On Authentication, since ECAS, CAS CA Site Minder and Crowd implement this technology.

4.2.6.2.Authorisation

This point analyses how applications manage authorisations.

Table 4-11 Analysis of identified software development methodologies used in the European Institutions – Authorisation

Authorisation	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Based on Roles	10	71%
Based application/tool	3	21%
Based on Groups	2	14%
N/A	2	14%

As it is showed in the chart most of the applications use an authorisation based on roles, few applications use groups and the rest uses the authorisation method offered by the base application or tool (i.e. Drupal, Alfresco).

4.3.Project Maintenance

In order to ensure the maintenance of the project and its sustainability, it is important to perform a good maintenance to fix bugs and vulnerabilities fast and in a preventive rather than reactive way.

4.3.1.1.Incident management support and categorisation

In this point we will analyse the tools, channels and procedures used to report bugs or vulnerabilities.

The following table analyses the entry points to report incidents.

Table 4-12 Analysis of identified software development methodologies used in the European Institutions – Bug and vulnerability reporting

Entry Point	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Helpdesk	11	79%
Jira	6	43%
Mailbox	1	7%

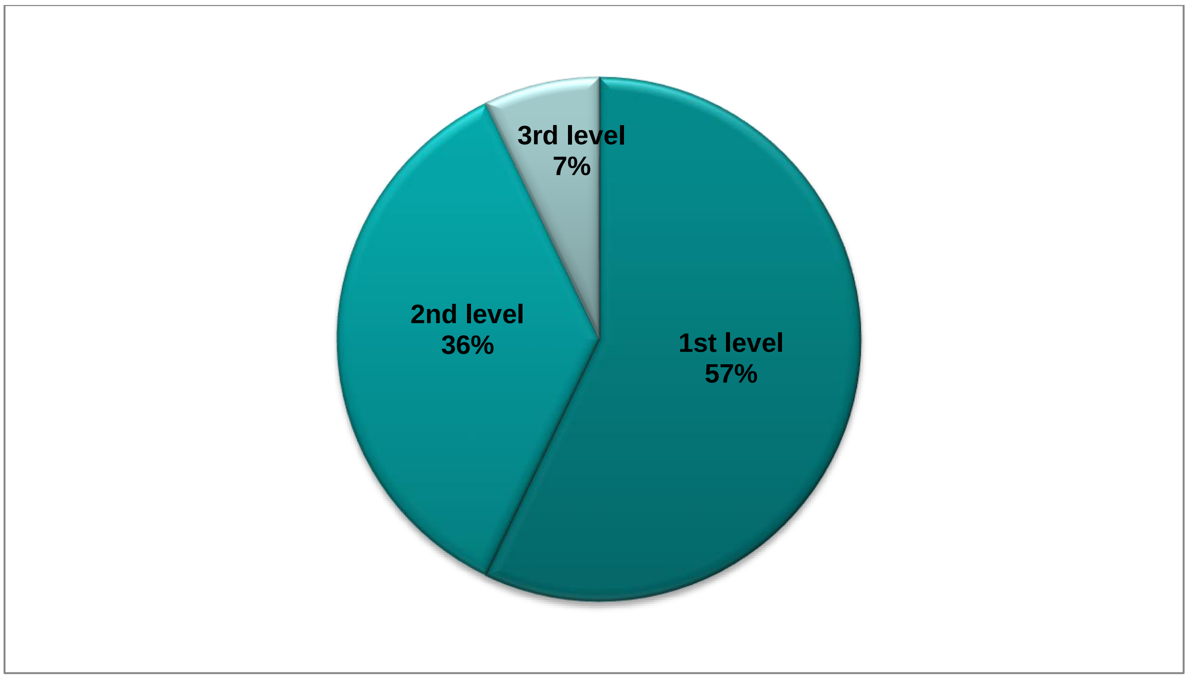
As the table shows most of the analysed projects use the helpdesk as an entry point for end-users to report bugs or vulnerabilities, 43% use Jira and 7% use their mailbox. The majority of the analysed projects that implements helpdesk as an entry point use SMT as a management tool.

Helpdesk is responsible for prioritising and categorising reported bugs or vulnerabilities from the projects that have use it as an entry point. In the rest of the cases, it is the development team who prioritises and categorises.

4.3.1.2. Incident resolution

For the analysed projects that have Helpdesk, this acts as a 1st level support to solve incidents related to user operations. Some of the projects analysed use a 2nd level support which is mostly composed of technicians from the IT team; if the problem cannot be solved, it is escalated to the 3rd level, which can be the problem management team within the project or the product provider.

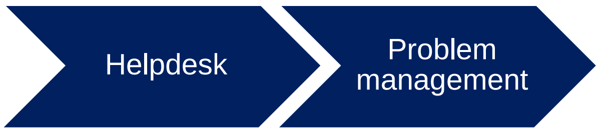
Figure 12. Analysis of identified software development methodologies used in the European Institutions – Incident supporting levels



Incident supporting level	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
1st level	8	57%
2nd level	5	36%
3rd level	1	7%

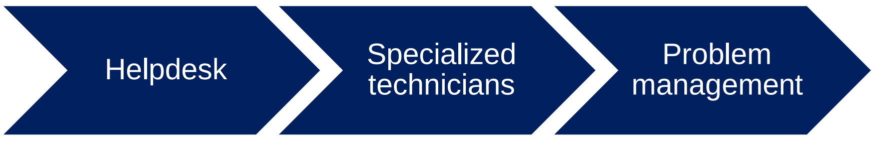
- **Projects with 1st level support**: This support level is performed by the helpdesk to resolve issues raised by the end-users. If the issue is not solved, it is escalated to problem management 4.3.2.2.

Figure 13. Analysis of identified software development methodologies used in the European Institutions – 1st level support flow



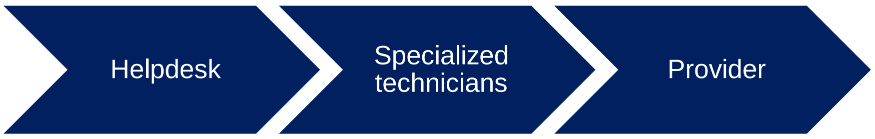
- **Projects with 2nd level support**: if the issue cannot be solved at the 1st level support, it is escalated to the 2nd level, and handled by specialized technicians from the IT team. If the issue is not solved, it is escalated to problem management 4.3.2.2.

Figure 14. Analysis of identified software development methodologies used in the European Institutions – 2nd level support flow



- **Projects with 3rd level support**: if the issue cannot be solved at the 2nd level support, it is escalated to the 3rd level and handled directly by the provider of the product or the problem management team.

Figure 15. Analysis of identified software development methodologies used in the European Institutions – 3rd level support flow



4.3.1.3.Handling of major incidents

In this point we will analyse if a special plan or procedure, different from the standard, is implemented for critical issues.

Table 4-13 Analysis of identified software development methodologies used in the European Institutions – Major incidents resolution

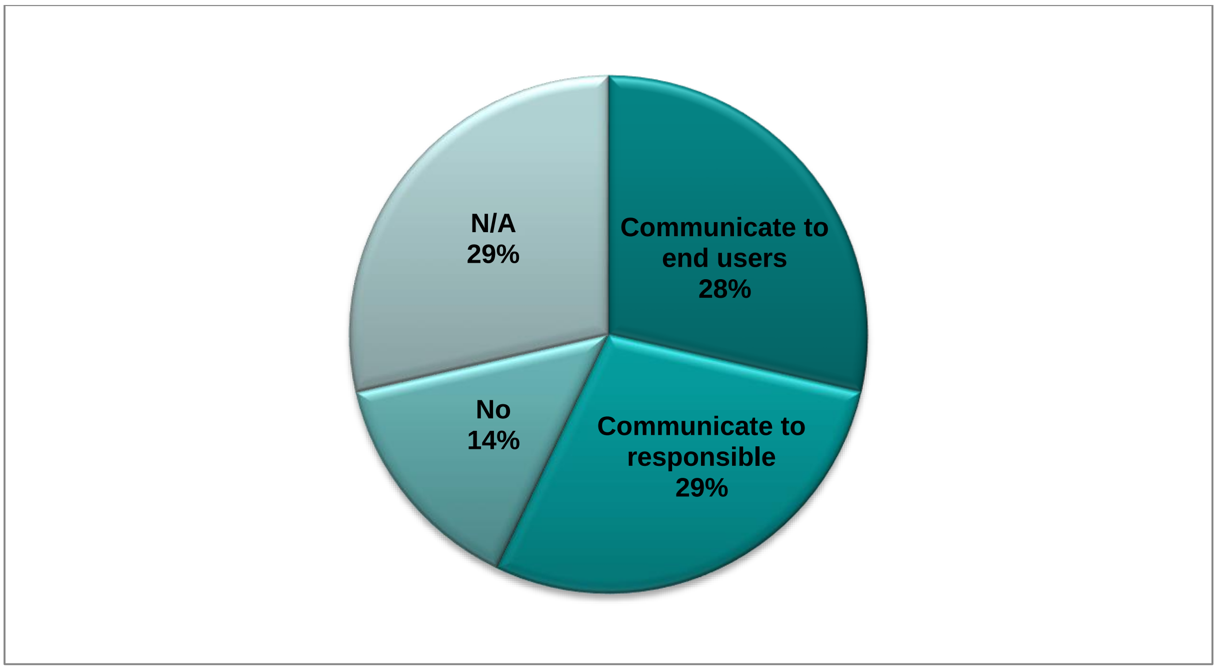
Special Procedure	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Raise the priority	13	93%
Skip environments from non-critical procedure	8	57%
Shutdown System	4	29%

All the projects have a special plan in order to address critical bugs or vulnerabilities, as it is shown in the previous table. 93% raise the priority to the highest in order to fix the issues as soon as possible, 29% shutdown the system in order to avoid the propagation of the vulnerability or to prevent errors caused by the bugs, and 57% of the projects skip some test environments to promote the fix to production as soon as possible and solve the incident.

4.3.1.4.User notification

This point analyses the approach used to notify the users when a new bug or vulnerability is identified or when it is necessary to conduct a new deployment that could cause service interruptions.

Figure 16. Analysis of identified software development methodologies used in the European Institutions – User notification



User notification procedure	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Communicate to end users	4	29%
Communicate to responsible	4	29%
N/A	4	29%
No	2	14%

29% of the analysed projects notify end users about the service interruptions, 29% to the person responsible for the application on the business side, two projects use the helpdesk to disseminate the communication and 14% conduct the deployment or patch installation without notification. For the remaining projects, this variable does not apply because the projects are not yet deployed in production.

4.3.1.5. Channels and tools used

The analysed projects that report the service interruptions use the following tools:

Table 4-14 Analysis of identified software development methodologies used in the European Institutions – Channels and tools for communication

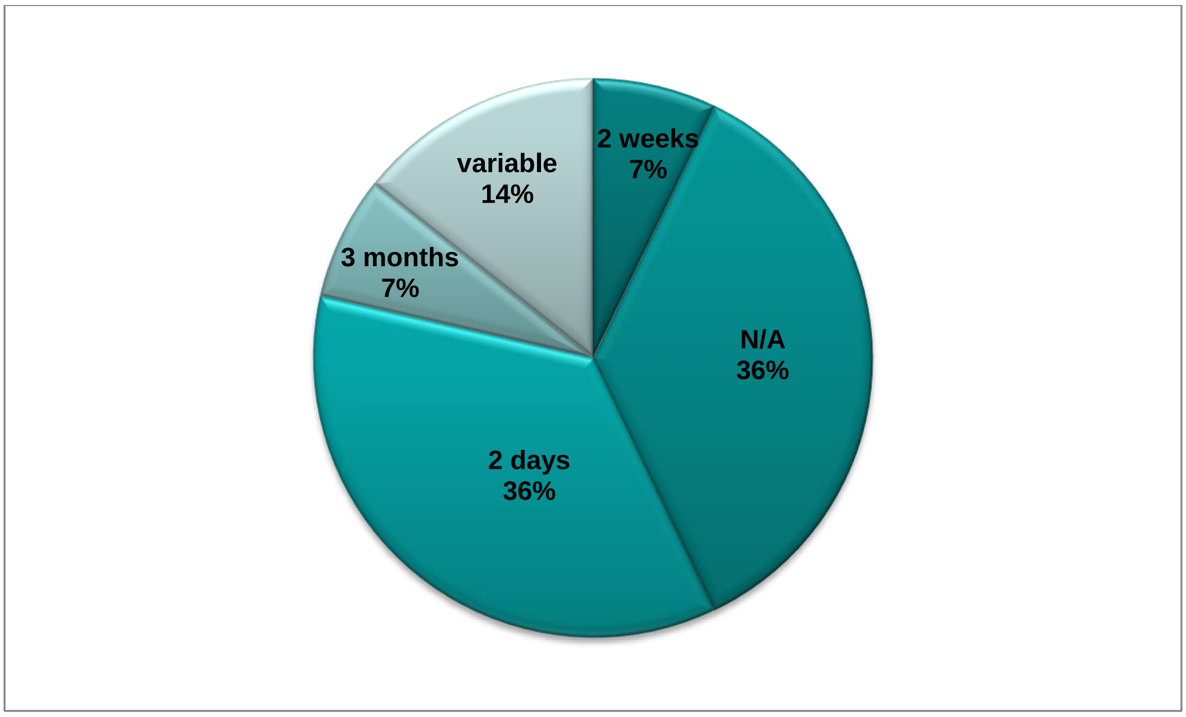
Channel and tool	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
e-mail	5	50%
Yammer	4	40%
Static Page	4	40%
Intranet	3	30%
Helpdesk	2	20%
Confluence	1	10%
IRM	1	10%

50% of the analysed projects use e-mail to notify, 40% use Yammer, 40% use a static page which shows a predefined message, 30% use the intranet, 20% use the helpdesk, 10% use Confluence and another 10% use IRM channel.

4.3.1.5.1.Time to resolve a bug or vulnerability

This is a metric to measure how much time the development team spends fixing the bugs and vulnerabilities to ensure the quality and security of the application.

Figure 17. Analysis of identified software development methodologies used in the European Institutions – time to fix bug / vulnerability



Period	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
N/A	5	36%
2 days	5	36%
Variable	2	14%
3 months	1	7%
2 weeks	1	7%

This chart shows the average time to resolve a bug, for five projects is 2 days, for one project it takes 2 weeks, for another one 3 months, two projects mention that is difficult to measure because depends of the bug, and another five do not have this information.

4.3.1.6. Conclusion

Taking into consideration ITIL [2] Incident Management as a best practice, the following processes are covered by the analysed projects.

- **Incident management support:** It is covered by the entry points and tools used for this purpose.
- **Incident logging and categorisation:** It is covered in the entry point and problem management phases.
- **Immediate incident resolution by 1st level support:** All the analysed projects have a least 1st level support.
- **Incident resolution by 2nd level support:** Some analysed projects have 2nd level support with specialized technicians.
- **Handling of Major Incidents:** Some analysed projects address major incidents with their own process.
- **Pro-Active User Information:** Not all end users of the analysed projects receive notification when the service is disrupted.

4.3.2. Problem Management

4.3.2.1. Identification of security updates or bugs

This point analyses how the project team identifies potential security updates or bugs in advance, as a preventive process. This point it is important since newly discovered vulnerabilities or bugs handled on time will reduce the probability of service disruptions, and thus will ensure the availability of the system and the quality of the service provided to end users.

Table 4-15 Analysis of identified software development methodologies used in the European Institutions – Identification of security updates or bugs

Procedure	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Regression Test	13	93%
Security information updated from providers	5	36%
Security information updated from external resources	5	36%

Most of the analysed projects identify deployed vulnerabilities and bugs using regression tests, but this procedure doesn't ensure that other bugs or vulnerabilities different from written tests are addressed. 36% of the projects rely on the recommendations from the provider of the baseline application (Piwik, Drupal, EMC, Alfresco) and the remaining 36% of the projects update the security information from external resources specialised in security.

4.3.2.2. Problem resolution plan

In this point we analyse how the project team responds to a bug or vulnerability that has been reported using the channels analysed in section 4.3.1.5

Table 4-16 Analysis of identified software development methodologies used in the European Institutions – Problem resolution plan

Procedure	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
Standard Release	11	79%
Special Process	5	36%

Most of the projects respond to problem resolution with a standard release procedure. The bug or vulnerability is logged as a task; the development team fixes the issue and uses the normal procedure, through the defined deployment pipeline, in order to promote the version between environments. However, three projects have a special plan to fix bugs or vulnerabilities and two projects use a combination of these two, standard release and special process, depending on the criticality of the issue.

4.3.2.3. Tools and resources used

- Selenium
- JUnit.
- OWASP.
- European Parliament Standards and Methods recommendations.
- Digit recommendations
- Provider recommendations (Drupal, Alfresco, Piwik)

4.3.2.4. Conclusion

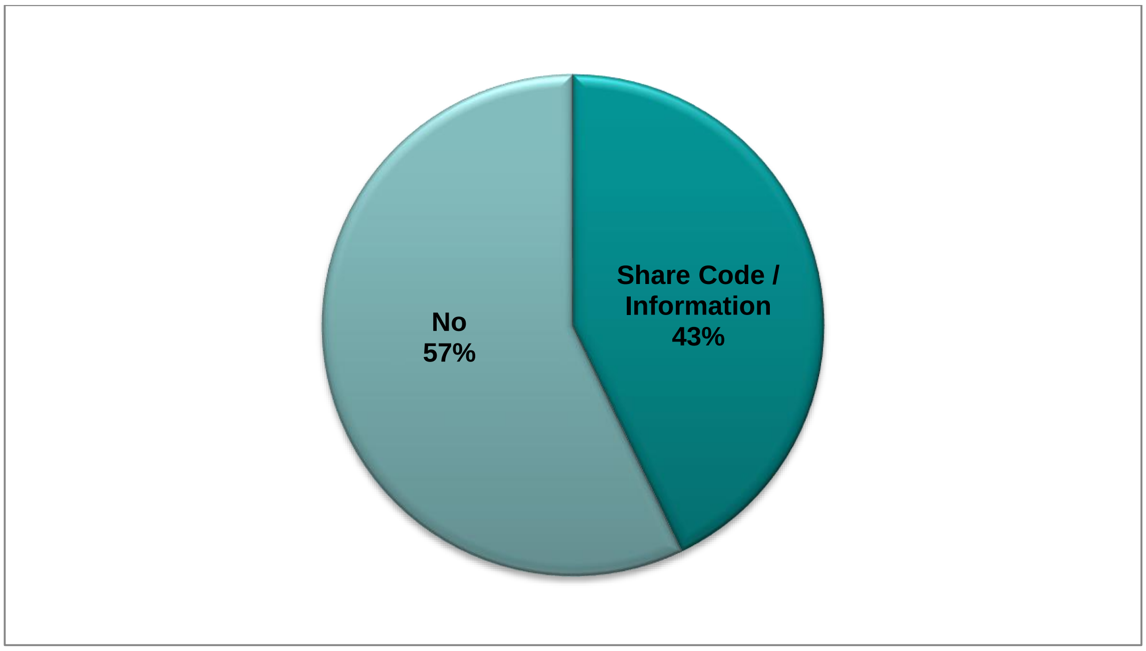
Taking into consideration ITIL [2] Problem Management as a best practice, the following processes are covered by the analysed projects:

- **Proactive Problem Identification:** Some projects update the security information from external resources to solve them as soon as possible. Regression tests are also used as they can provide bug identification in advance.
- **Problem Categorisation and Prioritisation:** It is covered within the entry point and development phases
- **Problem Diagnosis and Resolution:** It is covered by the IT team.

4.4. How European Institutions contribute to FOSS Communities

In this point we will consider how the analysed projects contribute to the FOSS Communities in terms of sharing code or information.

Figure 18. Analysis of identified software development methodologies used in the European Institutions – FOSS communities contribution



Foss communities contribution	Out of the 14 analysed projects	
	Number of projects	Percentage of the projects
No	8	57%
Share code / information	6	43%

The chart shows that 43% of the analysed projects share code or information with the FOSS Communities. Among these contributions, the most important ones are:

- **Code sharing**: For new or updated functionalities which have been developed within the European Institutions projects, they share the code with the corresponding FOSS communities.

- **Information Sharing**: Detailed information about procedures, methodologies and architectures is provided to the FOSS communities.

- **JoinUp**: 29% of the projects are published in JoinUp [4],

This analysis shows that the majority of the contributions to FOSS communities are on a personal level as the developers do not associate themselves with the European Institutions. Two projects contribute with the FOSS communities sharing the entire application to the community licensed as EUPL. The role used in this contribution is institutionally based and it is sponsored by the corresponding DG.

One project is willing to become a sponsor for the FOSS community that develops the product that they use, PIWIK.

4.5.Relevant opinions and advices from interviewees

During the interview process some opened questions were addressed to the interviewers. The most relevant opinions are:

1. PM^2 doesn’t define properly the security role.
2. Security and PM^2 should be linked
3. Security is coming at the end of the SDLC and it should be at the beginning.
4. Improve the security in the European Commission.
5. System security experts should report directly to the development team so security is considered from the early phases.
6. The European Commission should open up to Free and open source.
7. Create a list of supported Free and open source products.
8. The European Institutions should support European innovation using software companies or European FOSS communities.
9. The European Commission experts should provide a Free and open source replacement list for private applications.

5. Bibliographic references

[1] IBM, “IBM Rational,” [Online]. Available: http://www.ibm.com/software/rational.

[2] OWASP, “OWASP,” Free and open software security community, [Online]. Available: https://www.owasp.org.

[3] ITIL, “Itil Books,” [Online]. Available: http://www.itil.org.uk/.

[4] E. Comission, “joinup,” [Online]. Available: https://joinup.ec.europa.eu/.

6. Annexes

6.1. Questionnaire for the interview

Questionnaire

6.2. Executive summary

Executive summary