FP7-SME-2008-2 – 243768

OPEN-SME
"Open-Source Software Reuse Service for SMEs"

Final Report

Deliverable Type:	**PU***
Nature of the Deliverable:	**R****
Date:	**August 30th, 2012**
Distribution:	**WP1**
Code:	**OPEN-SME/EMHPEE/WP1/FR**
Editor:	**EMHPEE**
Contributors:	**Consortium**

* ***Deliverable Type:*** *PU= Public, RE= Restricted to a group specified by the Consortium, PP= Restricted to other program participants (including the Commission services), CO= Confidential, only for members of the Consortium (including the Commission services)*

** ***Nature of the Deliverable****: P= Prototype, R= Report, S= Specification, T= Tool, O= Other*

Abstract: This is the Final Report of the project, according to the specified format

The OPEN-SME Consortium consists of:

Enosi Mihanikon Pliroforikis & Epikinonion Ellados	Project Coordinator	Greece
Drustvo za informacione sisteme I racunarske mreze-Informaciono drustvo Srbije	Partner	Serbia
Epistimoniko Techniko Epimelitirio Kyprou (Technical Chamber of Cyprus)	Partner	Cyprus
Teknikbyn Science Park Vasteras AB	Partner	Sweden
SOLINET GmbH Telecommunications	Partner	Germany
GNOMON Informatics SA	Partner	Greece
Maelardalens Hoegskola	Partner	Sweden
Teletel S.A. - Telecommunications and Information Technology	Partner	Greece
Aristotelio Panepistimio Thessalonikis	Partner	Greece
Universiteit Maastricht	Partner	Netherlands
BITGEAR Wireless Design Services DOO	Partner	Serbia

PROJECT FINAL REPORT

Grant Agreement number: FP7-SME-2008-2-243768

Project acronym: OPEN-SME

Project title: Open Source Software Reuse Service for SMEs

Funding Scheme: Research for SME associations/groupings

Period covered: **from** 01/07/2010 **to** 30/06/2012

Name, title and organisation of the scientific representative of the project's coordinator:

Prof. Michalis Loupis, Greek Association of Computer Engineers

Greek Association of Computer Engineers, Karageorgi Servias 7, Athens 10563, Greece

Tel: +30-210 3325230

Fax: +30-210 7560324

E-mail: open-sme@computer-engineers.gr

Project website address: http://opensme.eu

Contents

1 Final Publishable Summary

1.1 Executive Summary

The OPEN-SME main idea is to introduce a reuse service that will be operated by SME Association Groups (AGs) on behalf of their SME software development members. This service will be operated by software experts of the SME AGs who will produce components from OSS projects, test them, generate documentation, resolve licensing etc. **asynchronously** to application development by SMEs and **independently** from the SMEs. The components will be related to domains that are relevant to the SMEs. Therefore when the SMEs will want to reuse them, the components will already be there. The OPEN-SME project collectively provides two processes and three tools, namely:

1. The Reuse-Oriented Domain Engineering (RODE) process.
2. The Application Engineering process.
3. The OCEAN tool. OCEAN is a tool for searching OSS code search engines. Essentially a meta-search engine.
4. The COPE tool. COPE is a tool for extracting, testing, documenting and packaging software components originating from OSS projects.
5. The COMPARE tool. COMPARE is a repository for storing the extracted component packages and delivering them to SMEs.

The Open-Source Search Engine (OCEAN) is a meta-search engine that provides unified access to existing Open Source Software (OSS) search engines. This allows the reuse-engineer to find open source software assets (i.e projects, packages, files etc.) satisfying certain criteria, such as software that is written in a specific programming language, containing certain keywords, having a specific license etc. Moreover, it allows the re-user to detect a software asset that is of some value and place an order to adapt that specific asset to the reuse-engineer. OCEAN is a web portal (*http://ocean.gnomon.com.gr/web/guest/home , root/test)* that allows mainly locating and browsing open-source files and projects that are available on popular open-source search engines. The Component Adaptation Environment (COPE) is a tool-chain that provides an environment for the enactment of the domain engineering process of OPEN-SME, thus allowing the reuse-engineer produce reusable components for the domain(s) of interest (*http://opensme.eu/deliverables/86-deliverable-d32b , trialsuser/opensmeuser)*. COPE is a desktop application to perform the following tasks in order to achieve the aforementioned result:

- Identify and model primary concepts of the domain
- Analyse the different aspects
- Comprehend the project and detect candidate components
- Generate components and validate them
- Classify the produced component
- Upload the Component to COMPARE component repository and search engine.

The Component Repository and Search Engine (COMPARE) is a web portal (*http://www.teletel-projects.net/compare , demo/1234),* that allows SME software re-users to search and discover software artefacts, technical documents, test suites related to open-source software. In addition, it allows the stakeholders of the Domain Engineering Process (reuse engineers, domain experts, component testers and certifiers) to manage and maintain the assets stored in the repository. The end-users can be endowed by using the advanced filtering capabilities as well as by accessing information about the verification and certification attributes of a component. Finally, it provides a communication mechanism between the re-users and the reuse-engineers

1.2 Summary description of project context and objectives

1.2.1 Overview

Open Source Software (OSS) reuse has the potential to improve software quality, shorten time-to-market and bring competitive advantages to Software Development SMEs. However, currently OSS reuse is restricted to:

- Whole OSS projects (e.g. Apache web server, MySQL Database)
- Opportunistic reuse of isolated classes (i.e. copy-paste-adapt reuse).
- Well-known selected infrastructure components (e.g. Apache Commons)

The OPEN-SME proposal is to extend the landscape of OSS reuse to domain-specific components extracted by arbitrary OSS projects. Achieving this goal however involves a number of challenges:

- Valuable OSS components exist in every OSS project. However it is difficult to recognize them, extract them, test them, document them etc.
- During software development, usually there is no time for the aforementioned activities. Developers often prefer to develop new code from scratch although this code has been written before many times by many others.
- Even when developers recognize the opportunity to reuse OSS code there are several uncertainties related to the provided functionality and quality.
    - What the component does exactly?
    - How well it does it?

The OPEN-SME main idea is to introduce a reuse service that will be operated by SME Association Groups (AGs) on behalf of their SME software development members. This service will be operated by software experts of the SME AGs who will produce components from OSS projects, test them, generate documentation, resolve licensing etc. **asynchronously** to application development by SMEs and **independently** from the SMEs. The components will be related to domains that are relevant to the SMEs. Therefore when the SMEs will want to reuse them, the components will already be there. The OPEN-SME project collectively provides two processes and three tools, namely:

1. The Reuse-Oriented Domain Engineering (RODE) process.
2. The Application Engineering process.
3. The OCEAN tool. OCEAN is a tool for searching OSS code search engines. Essentially a meta-search engine.
4. The COPE tool. COPE is a tool for extracting, testing, documenting and packaging software components originating from OSS projects.
5. The COMPARE tool. COMPARE is a repository for storing the extracted component packages and delivering them to SMEs.

1.2.2 OCEAN

The Open-Source Search Engine (OCEAN) is a meta-search engine that provides unified access to existing Open Source Software (OSS) search engines. This allows the reuse-engineer to find open source software assets (i.e projects, packages, files etc.) satisfying certain criteria, such as

software that is written in a specific programming language, containing certain keywords, having a specific license etc. Moreover, it allows the re-user to detect a software asset that is of some value and place an order to adapt that specific asset to the reuse-engineer. OCEAN is a web portal (*http://ocean.gnomon.com.gr/web/guest/home , root/test)* that allows mainly locating and browsing open-source files and projects that are available on popular open-source search engines. OCEAN is extensible to incorporate any open-source search engine available regardless of the integration strategy. What this means is that the integration of an arbitrary search engine can be performed in any way possible (i.e. use of provided api, web-scrapping, etc).

1.2.3 COPE

The Component Adaptation Environment (COPE) is a tool-chain that provides an environment for the enactment of the domain engineering process of OPEN-SME, thus allowing the reuse-engineer produce reusable components for the domain(s) of interest (*http://opensme.eu/deliverables/86-deliverable-d32b , trialsuser/opensmeuser)*. COPE is a desktop application to perform the following tasks in order to achieve the aforementioned result:

- Identify and model primary concepts of the domain (using: Knowledge Manager)
- Analyse the different aspects (using: Static Analysis, Design-pattern Analysis, etc.) of an Open-Source project
- Comprehend the project (using: the outcome of the Analysis, Documentation Generation, in-project Search)
- Detect candidate components (using: the outcomes of the project Analysis (ii) and project Comprehension (iii))
- Generate components (using: the various Component Makers)
- Validate them (using: Dynamic Analysis)
- Classify the produced component (using: Knowledge Manager)
- Upload the Component to COMPARE component repository and search engine.

As far the physical architecture is concerned, only a couple of the aforementioned tasks initiate an interaction with one of the OPEN-SME servers. In the following subsection we describe each scenario of use and the associated servers as they are instantiated for the OPEN-SME trials.

1.2.4 COMPARE

The Component Repository and Search Engine (COMPARE) is a web portal (*http://www.teletel-projects.net/compare , demo/1234),* that allows SME software re-users to search and discover software artefacts, technical documents, test suites related to open-source software. In addition, it allows the stakeholders of the Domain Engineering Process (reuse engineers, domain experts, component testers and certifiers) to manage and maintain the assets stored in the repository. The end-users can be endowed by using the advanced filtering capabilities as well as by accessing information about the verification and certification attributes of a component. Finally, it provides a communication mechanism between the re-users and the reuse-engineers

1.2.5 System Architecture as a whole

In Figure 1, a consolidated view of the system architecture is provided, depicting all the OPESME servers and the roles for both the SME-AGs and the SMEs.

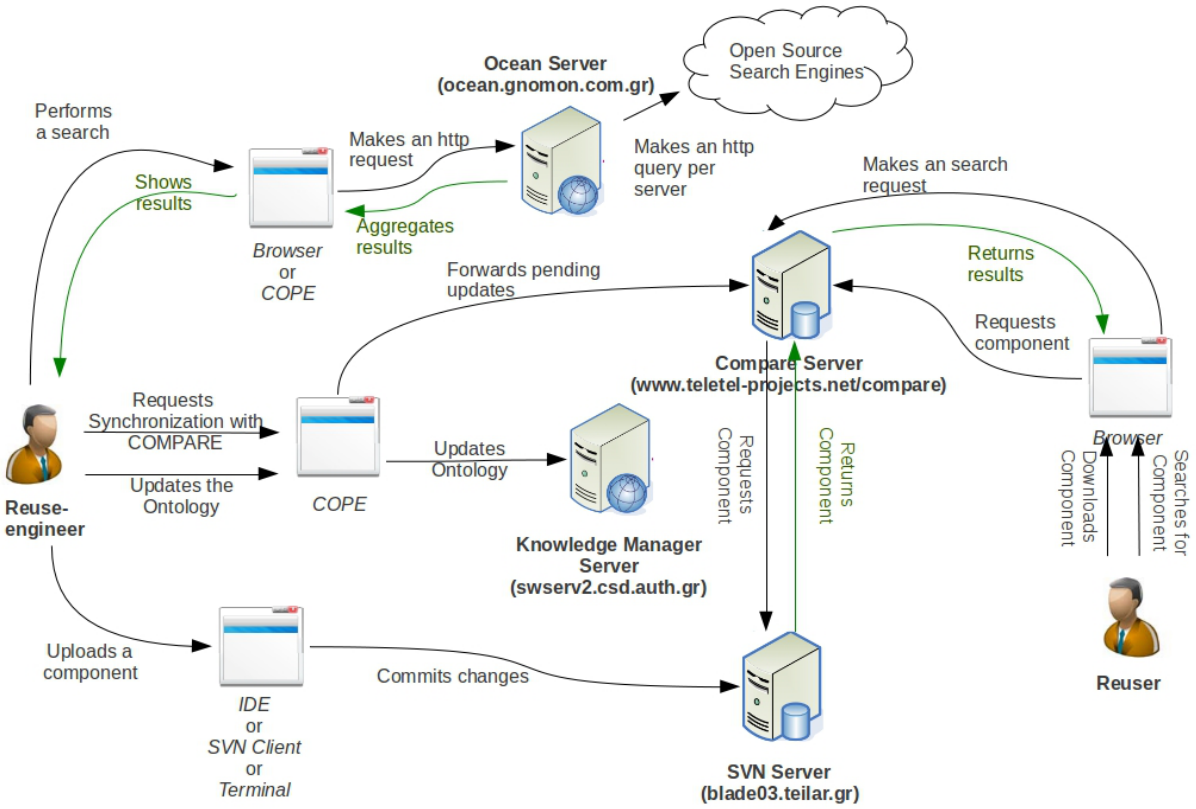


Figure 1: An overview of the whole physical architecture[*].

The reason this decentralized topology was selected is that the end-users perceive it as a robust, fault-tolerant system. So, if one of the servers malfunctions, the rest of the functionalities provided by the system do not cease to exist, but on the contrary the associated users can still perform their tasks without being affected by a malfunction that is irrelevant with what they have to perform. Moreover, this architecture makes the evolution of the services independent from each other which is both desirable and necessary. It is desirable, not only for purposes of robustness and fault-tolerance but also for tracking and maintaining reasons. It is also necessary because at any moment during the trials the end-users may require additions or enhancements in order to successfully use the services, so the services should be easily maintainable thus independent from each other. Nevertheless these services can be hosted on a single physical server and thus do not impose additional costs to the SME-AGs.

*Some return messages are omitted.

1.3 Description of the main S&T results/foregrounds

The OPEN-SME main idea is to introduce a reuse service that will be operated by SME Association Groups (AGs) (e.g. Greek Association of Computer Engineers, Vasteras Science Park etc.) on behalf of their SME software development members. This service will be operated by software experts of the SME AGs who will produce components from OSS projects, test them, generate documentation, resolve licensing etc. **asynchronously** to application development by SMEs and **independently** from the SMEs. The components will be related to domains that are relevant to the SMEs. Therefore when the SMEs will want to reuse them, the components will already be there. In Figure 2 we can see an overview of the OPEN-SME project which collectively provides two processes and three tools that we will describe in some detail in the following sections:

- Section 1.3.1 will provide a description of the Reuse-Oriented Domain Engineering (RODE) process that was developed in the context of the OPEN-SME project. This process can also be accessed online in http://opensme.eu/rode/. The online process description contains all the knowledge that is necessary to enable an SME AG to apply the process.

- Section 1.3.2 will provide a description of the Application Engineering process that was developed in the context of the OPEN-SME project. This process is used by the SMEs to develop their software projects and contains specific activities cantered on the reuse of the software components extracted by the RODE process. This process can also be accessed online in http://opensme.eu/aep/ which contains all the knowledge necessary to the SMEs for applying the process.

- Section 1.3.3 will provide a description of the OCEAN tools that was developed in the context of the OPEN-SME project. OCEAN is a tool for searching OSS code search engines. Essentially a meta-search engine.

- Section 1.3.4 will provide a description of the COPE tool that was developed in the context of the OPEN-SME project. COPE is a tool for extracting, testing, documenting and packaging software components originating from OSS projects.

- Section 1.3.5 will provide a description of the COMPARE tool that was developed in the context of the OPEN-SME project. COMPARE is a repository for storing the extracted component packages and delivering them to SMEs.

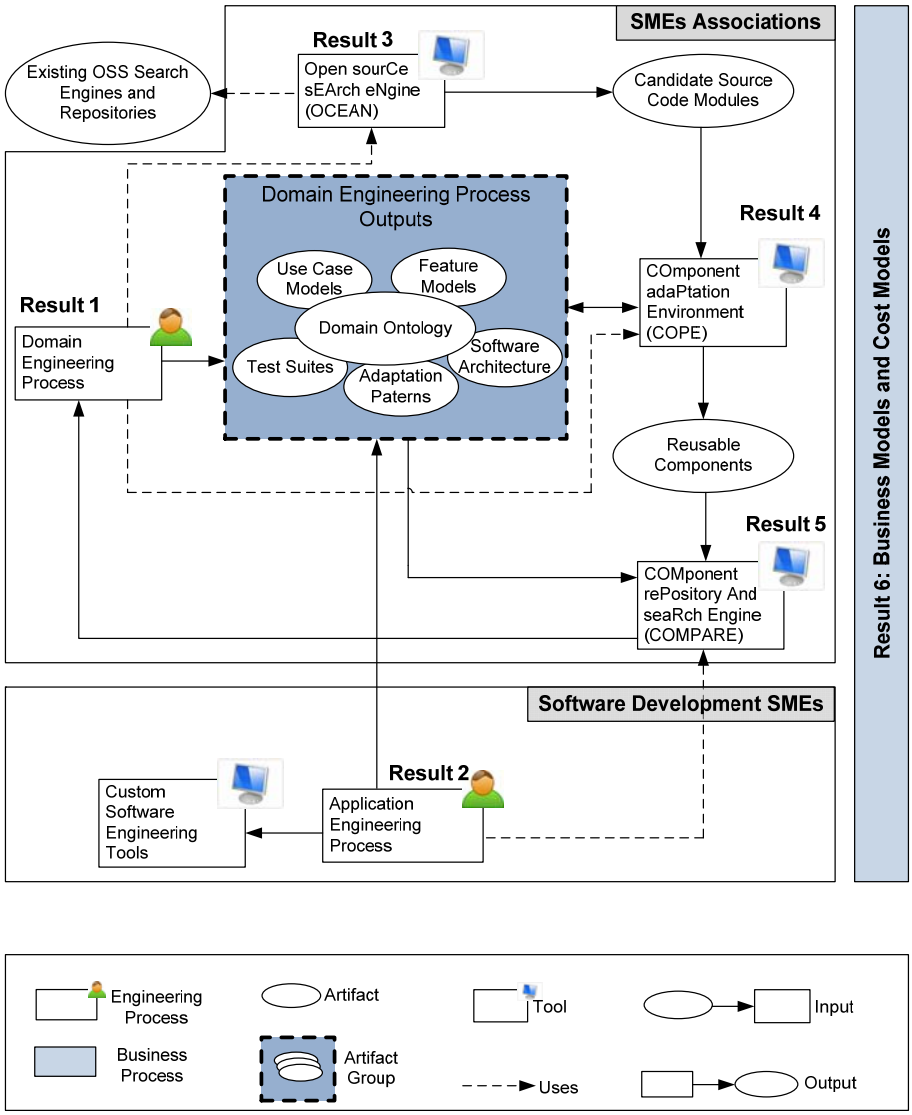


Figure 2: OPEN-SME project overview

1.3.1 Domain Engineering Process (RODE)

In this section we will discuss a domain engineering process for the creation of domain models based on existing OSS projects. We believe OSS projects provide not only a quality alternative to commercial software but also a knowledge resource that we can exploit in developing the necessary domain knowledge for the domain engineering. Domain engineering methods invariably propose the use of so-called exemplar projects that are existing projects used during domain analysis and design. We propose a domain engineering process that uses OSS projects as exemplars during all phases of domain engineering, including the domain implementation phase in which existing OSS components are reused for the partial implementation of the domain artifacts. The process is suitable for Small and Medium Enterprises (SMEs) that experience limitation of resources and characterized by a limited portfolio of owned projects having difficulties in applying systematic reuse methods based on domain engineering approaches.

1.3.1.1 Introduction

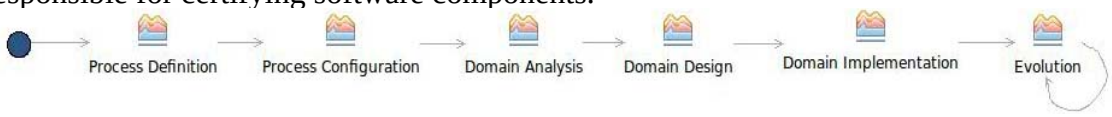
Systematic software reuse is divided into a) activities and/or processes related to building reusable assets, referred as *domain engineering* processes or methodologies, and b) activities and/or processes related to reusing these assets in the context of a software application

development, referred as *application engineering* processes. The authors in [1] define domain engineering as "*the set of activities involved in developing reusable assets across an entire application domain, or family of applications*". In domain engineering a number of applications, belonging to a specific domain, are identified and their similarities and variabilities are analysed in order to produce a domain model. Thereafter the model is designed, implemented, and concrete artefacts of the implemented model are produced to be reused in a number of applications.
The domain engineering process is used to create a specific reusable software platform in which future applications will be based upon. It encompasses phases for requirements analysis, design, implementation and testing of this platform. After this reusable platform has been implemented applications can be developed more efficiently with the reuse of the platform.
Domain engineering is a necessary step towards the establishment of systematic reuse within a software development organization. However there have been known limitations such as the difficulties in analysing a domain thoroughly [2] and therefore tactical reuse should be allowed to prove its value before the domain analysis is completed, to extend a domain model beyond its initial scope [3] and in developing reusable modules, gaps among analysis, design and implementation in reuse processes and achieving development with reuse in conjunction with development for reuse [4]. Domain engineering therefore constitutes an active research area independently or, more recently, in conjunction with product line approaches to reuse [5].
In our research work in the context of the OPEN-SME EU[1] funded project, we look at methods and tools for enabling Small and Medium Enterprises (SMEs) to effectively reuse Open Source Software (OSS) components in their software development processes. In order to establish a systematic link between OSS available components and their domains so that their reuse is more efficient we formulated a domain engineering process for SME Association Groups (SME-AGs) that uses open source software projects as exemplar applications used for domain analysis, as well as a source of reusable components during domain "implementation".

1.3.1.2 RODE: A domain engineering process based on OSS projects

Figure 3 depicts the RODE process comprising of distinct phases which will be analyzed in detail. Each one of the phases, is performed only once with the exception of the Evolution Phase. In the RODE process, we try to build all the necessary tools, reusable assets, artefacts, documents, models, etc. until they reach a certain level of maturity thus allowing SME-AGs to provide the services of OPEN-SME, and perform a continuous, on-going, evolution of the assets. The stakeholders of the Domain Engineering Process are:

- Reuser: Software re-user (in particular SMEs) is the key beneficiary role since they apply their application engineering process using reusable assets produced by RODE process.
- Reuse Engineer: Reuse engineers are professionals (hired by SME-AGs) who are responsible for discovering and adapting software components in order to produce reusable assets that will be stored in the Reuse Repository.
- Domain Expert: Domain Experts are professionals that specialize in a specific domain and are engaged in assisting the reuse engineer by providing their knowledge on the domain.
- Tester: Testers are responsible for the core activities of the test effort. Their main responsibility is to test software components.
- Certifier: Certifiers are software engineers with experience in Software Verification and responsible for certifying software components.



[1] http://opensme.eu/

Figure 3: *RODE Process overview*

The tools that are provided by the Open-SME in order to fulfil its goals are:

- OCEAN: The Open Source Search Engine (OCEAN) is meta-search engine that allows the initial discovery of OSS projects and/or components by providing unified access to existing open source software search engines and forges.
- COPE: The Component Adaptation Environment (COPE) is a tool-chain that assists the reuse engineer to the enactment of the domain engineering process. It also allows the Tester and Certifier to test, verify and certify software components by providing testing and model checking tools.
- COMPARE: The Component Repository and Search Engine (COMPARE) will serve as the Reuse Repository that will allow reusers to search and discover reusable assets produced by the Domain Engineering Process and the COPE tool-chain.

In the website[2] of the process we provide further information regarding the roles, methods and tools included in the RODE process and generally more detailed information about the process itself.

1.3.1.2.1 Process Definition Phase

This phase aims at organizing the usage of resources and the way the process as a whole will be carried out. In this phase the reuse Engineer should create, document and execute a domain engineering plan including standards, methods, activities, assignments, and responsibilities for performing domain engineering including the candidate stakeholders. S/he will also select any additional representation forms to be used for the domain models.

1.3.1.2.2 Process Configuration Phase

The purpose of this phase is to configure (if necessary) the process itself to address the specificities of the domain of interest by performing the following activities (as shown in *Figure 4*):

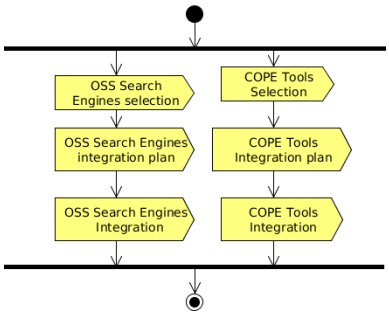


Figure 4: Overview of the Process Configuration phase.

1. *OSS Search Engine selection*: Refers to the selection of the most suitable Open Source Software Search Engines for the domain of interest. Selected engines will be the only ones used in order to discover OSS Projects.
2. *OSS Search Engine Integration plan*: In this (optional) activity the reuse engineer decides whether any OSS search engine, identified in the "OSS Search Engines selection" activity, should be integrated into OCEAN tool or used "as-is". The reuse engineer should design the integration of the OSS search engine into OCEAN, or design how the results of the OSS Search Engines can be exploited by COPE, respectively.
3. *OSS Search Engine Integration*: In this (optional) activity the reuse engineer implements either the integration of the selected OSS search engines into OCEAN or the process and tool, if required, to exploit the search engine externally.

[2] http://opensme.eu/rode

4. *Tool selection*: The reuse engineer selects any additional tools that might be necessary for the implementation of the Domain Engineering Process and/or for instantiation of COPE. For the selection of the most appropriate tools, the reuse engineer can use a decision analysis method.
5. *Tool Integration plan*: In this (optional) activity the reuse engineer decides whether any additional tools should be used independently or integrated into COPE. The reuse engineer should design how the results of the additional tools can be exploited by COPE or design the integration of the additional tools into COPE, respectively.
6. *Tool Integration*: In this (optional) activity the reuse engineer implements the integration of any additional tools with COPE (resulting in a new instance of COPE) or the process and tool, if required, to exploit the assets produced by the specific tool externally.

1.3.1.2.3 Domain Analysis Phase

In this phase the reuse Engineer has to analyze the domain(s) of interest by performing the following activities (as shown in Figure 5):

1. Domain Boundary Definition: The reuse engineer, assisted by the domain expert, should define the boundaries of the domain.
2. Primary Concept Identification and Modelling: In this iterative activity, the reuse engineer while analyzing the domain of interest identifies primary concepts of the domain and models them in the Ontology provided by the Knowledge Manager of COPE.

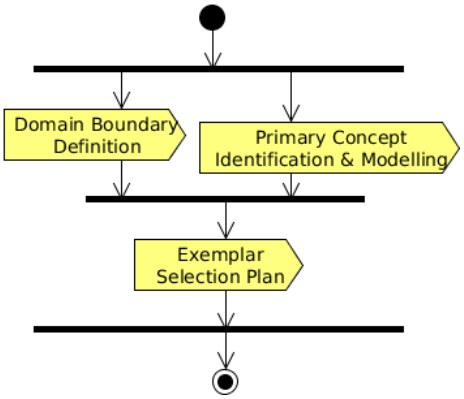


Figure 5: Overview of the Domain Analysis phase.

3. Exemplar Selection Plan: In this activity the reuse Engineer should create and document in which way the exemplars will be selected. S/he should identify and document the criteria, as well as their relative importance by which an exemplar is more suitable to be selected for reuse. These should include functional, technical, business criteria. Finally, s/he should estimate the number of exemplars required to cover the primary concepts.

1.3.1.2.4 Domain Design Phase

In this phase the reuse Engineer selects exemplar projects for the domain of interest and validates whether they are within the domain scope by performing the following activities (as shown in Figure 6):

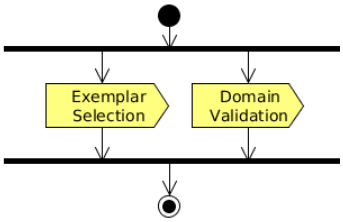


Figure 6: Overview of the Domain Design phase.

1. Exemplar Selection: In this activity the reuse engineer executes the Exemplar Selection Plan and discovers, selects and retrieves the most representative OSS projects to be used as exemplars. Based on the criteria defined in the exemplar execution plan s/he evaluates them using a decision analysis method and selects the most appropriate.
2. Domain Validation: While the reuse engineer searches exemplars, the domain expert should validate whether the exemplars are out of the domain boundaries or the domain boundaries are too strict. In that case, the reuse engineer can either exclude the exemplar or modify the domain boundaries at his/hers discretion.

1.3.1.2.5 Domain Implementation Phase

In this phase the reuse Engineer has to implement all the assets assimilating the exemplars integration and incorporate the selected exemplars. This phase is broken down to the following activities (as shown in *Figure 6*Figure 7:

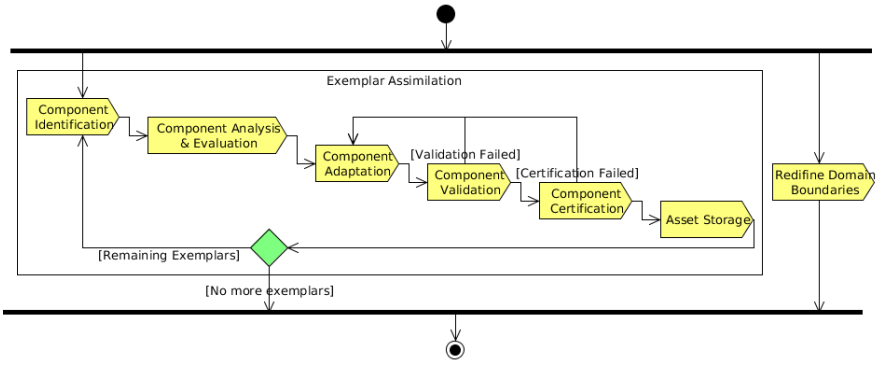


Figure 7: Overview of the Domain Implementation phase.

1. *Exemplar Assimilation*: This iteration, performed mainly by the reuse engineer, aims at the assimilation of each exemplar by following the activities *2 - 7*.
2. *Component identification*: Using reverse engineering tools, static and dynamic analysis tools provided by the instantiation of COPE, the reuse engineer identifies reusable components within the project.
3. *Component Analysis & Evaluation*: Afterwards, the reuse engineer analyzes each component, identifies concepts of the components related to the domain of interest, and evaluates its suitability using decision analysis methods.
4. *Component Adaptation*: Using model driven development tools and the adaptation pattern library of COPE, the reuse engineer adapts the component and documents the resulting asset.
5. *Component Validation*: In this task, the tester validates the component making use of the validation tools provided by COPE.
6. *Component Certification*: In this (optional) task the Certifier using advanced certification techniques, such as model-checking, certifies that a specific component has a set of desired properties.
7. *Asset Storage*: Upon successful completion of the previous activities, the reuse engineer models into the Ontology the concepts that are related to the component and gathers all the produced artifacts. S/he then stores the component into COMPARE along with its metadata or other assets (Metrics, Use cases, UML Diagrams, Test Cases, etc.)
8. *Redefine Domain Boundaries*: While the reuse engineer executes the “Exemplar Assimilation” s/he may have to redefine the domain boundaries.

1.3.1.2.6 Evolution Phase

In this perpetual and iterative phase the reuse engineer assimilates new projects into the Reuse Repository while maintaining the already embodied assets and thus evolves the domain engineering process as a whole. This is performed by following the activities described below (as shown in *Figure 8*).

1. *New OSS Search Engine Discovery and Integration*: In this (optional) activity the reuse engineer performs the corresponding activities described in the "Domain Configuration" phase.
2. New Tool Discovery and Integration: In this (optional) activity the reuse engineer performs the corresponding activities described in the "Domain Configuration" phase.
3. Exemplar Selection: The reuse engineer performs corresponding activities described in the "Domain Design" phase.
4. Project Assimilation: In this iteration, performs the corresponding activities described in the "Domain Implementation" phase.
5. Component Certification: In this task the Certifier using advanced certification techniques, such as model-checking, certifies that a specific component has a set of desired properties.

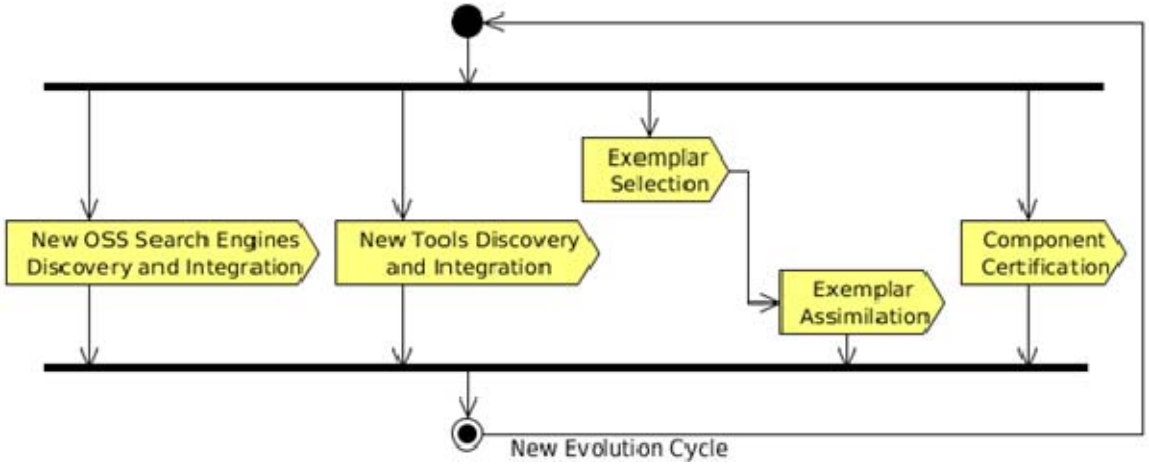


Figure 8: Overview of the Evolution phase.

1.3.1.2.7 Related Work In Domain Engineering Processes

In this section we will review some of the more established approaches to domain engineering as well as some more recent proposals. *Feature Oriented Reuse Method (FORM)* [6], an extension of Feature Oriented Domain Analysis (FODA) [7], is a method for both domain engineering and application engineering using the reusable assets which are the outcomes of the domain engineering process. It creates a domain dictionary which includes capabilities, operating environments, domain technologies and implementation techniques along with composition rules and techniques for application development. The FORM domain engineering approach comprises three phases: (a) *Context analysis*: Scoping and intended use of the domain applications as well as identification of external conditions and interactions with the external world. (b) *Domain (or feature) modeling*: Identification of domain features and their interactions, creation of the feature model, and (c) *Architecture (and component) modeling*: Construction of the reusable components and their configurations and hierarchical decompositions. According to the authors of [6] good sources of features according to their type include (a) user manuals for capabilities, (b) requirement and design documents for operating environment and domain technology features, and (c) design documents and source code for implementation features.

In [8,9], *Organization Domain Modeling (ODM)* was proposed as a domain modeling approach that addresses difficulties observed with previous domain engineering approaches including scoping, contextualizing, separating descriptive from prescriptive modeling approaches and formalizing models of variability. The intent behind ODM was to produce a configurable Domain Analysis process model that could be used by different types of organizations developing systems in diverse domains, using a variety of implementation technologies. To achieve this neutrality in relation to organizations and implementation technologies, ODM provides a *core domain modeling lifecycle* that can be integrated with a variety of *supporting methods* that diverse organizations may want to use. ODM was designed to be applied to domains that are mature,

relatively stable and economically viable. ODM is organized in phases which are: *Domain planning (*set the objectives and scope of the domain), *Domain Modeling (*produces a domain model for the domain) and *Asset Base Engineering (*architects and implements an asset base that addresses the market needs).

Reuse-Driven Software Engineering Business (RSEB) [10] is a software engineering approach based in (UML) models for developing families of applications with reuse. RSEB has several processes which analyze existing identified applications in order to define a layered architecture and decompose into component systems, analyze sets of requirements to produce reusable components and constructs applications with the selection of components their customization to the application needs. Features, as in FODA [7] or FORM [6], are not explicitly defined but (some of them) are instead part of use case definitions. This presents some problems (absence of technical/implementation details, difficulties in decision making regarding design/technical/implementation issues, etc.) since use cases are user-oriented whereas features are reuser-oriented [11]. To overcome these problems FeatuRSEB [11] uses the feature model as a central unifying diagram for expressing commonality and variability in the domain allowing the reuser to understand what can be combined, selected or customized and possible constraints among features. In FeatuRSEB the inputs of the process include exemplar systems, domain expertise, requirement documents and early domain models. The outputs are the feature model, the use case model, the context model and the domain scope.

To overcome shortcomings when applying existing processes into an industrial environment, *Product Line Software Engineering (PuLSE)* [12] was introduced. These shortcomings include: (a) Existing domain analysis methods had a domain focus rather than a product focus which is important for the enterprises, (b) are vague and inflexible, and (c) are overstressing the organizational issues neglecting the technological issues. PuLSE has four *deployment phases* which are: (a) *Initialization*: During the initialization the product line is customized to the specific organizational context. (b) *Infrastructure Construction*: The infrastructure construction is responsible for scoping, modeling and architecting the product line infrastructure. Specifically the scoping of the infrastructure takes under consideration the specific products that exist, are in development or anticipated for the particular organization. (c) *Usage:* Usage of the product line involves the instantiation and validation of *one member of the product line*. During usage of this member change requests may arise. These are not handled in the usage phase of the process but rather are passed as *change requests* in the evolution and management phase. (d) *Evolution and Management:* This phase handles change requests by consolidating, evaluating and determining their possible ramifications for the product line.

The *KobrA approach* to product line engineering [13, 14] attempts to integrate the component-based approach for software development (reuse in-the-small) with the product line approach (reuse in-the-large). KobrA can be viewed as a customization of the PuLSE method suitable also for immature organizations. PuLSE may be proved problematic for this particular setting, since it assumes that the activities of PuLSE will be customized for the existing software development methodology. On the other hand KobrA aims at being concrete and prescriptive. The infrastructure construction phase of PuLSE corresponds to KobrA *framework engineering*, the usage phase of PuLSE corresponds to KobrA *application engineering* and the evolution and management phase of PuLSE corresponds to KobrA *maintenance* of the framework and the applications. The use of UML in KobrA, makes it suitable for the majority of software engineers as well as neutral to programming languages and component technologies.

Since RODE process is focused on SMEs and SME-AGs, it differentiates from a “typical” domain engineering process for a couple of reasons. In RODE, we diverge from creating reference architecture as this would undermine the competitiveness between SMEs that share the same domains as well as it would be impossible to create such an architecture that would incorporate all the different technologies already adopted or implemented by the SMEs. Also, to

identify concepts in domain analysis and design we use exemplar open source projects, thus providing real-life examples as well as short-term exploitable results. Finally, in the domain implementation phase we tend to create all sorts of reusable assets from existing resources found in the OSS rather than building them from scratch.

1.3.2 Application Engineering Process

Component based software engineering had received significant focus from the research community during the last decade and several interesting models have been proposed. At the same time, Open source software development also had become popular, thanks to the dedicated efforts of the developer community. Both communities have a lot to learn from each other and a proper blending of their processes and methods could provide the software developers with greater opportunities and well as cost efficiency.
In this Section, we present the specification of an application engineering process envisaged for the reuse-oriented software development approach that can be beneficial for small and medium enterprises (SMEs). This application engineering process is described through various phases and activities included therein. This process is to help the SME engineers to have a clear picture and comprehension of the issues involved. We also present a set of requirements and challenges identified for the realization of such a process.

1.3.2.1 Introduction

In spite of the large research efforts on the component based software engineering (CBSE) as well as the growing development efforts of the open source software community, we are yet to see any strong efforts in bringing a synergy between these two communities. We believe that understanding the models and processes proposed by CBSE and blending them carefully to their processes and models could provide the open-source community with much greater re-use capability and hence cost-efficiency.
One of the high level objectives of OPEN-SME is to define and systematically document the OPEN-SME generic and customisable Application Engineering Process. In the context of the OPEN-SME business cases, 'Application Engineering Activities' are typically performed by the Software Development SMEs. These activities need to be organised in the context of a component-based and reuse-oriented Software Development Methodology that is capable of exploiting the outcomes of the Domain Engineering Process (Section 1.3.1). OPEN-SME will also develop the OPEN-SME Component Repository and Search Engine (COMPARE, Section 1.3.5) having the following main features:

- Allow software resuers to effectively search, browse, and retrieve the assets produced by the Domain Engineering Process. These assets include software artefacts, technical documents, test suites, metamodels, etc.
- Provide to resuers a clear view of the software component attributes relating to software qualification and certification.
- Provide a communication channel supporting the effective exchange and processing of structured information flows between the software reuse stakeholders (placement of orders for software components, bug reports, event notifications, etc.)

There exist many models for software development processes and lifecycles. Most of them are specified considering some specific, often non-technical goals, such as quality, predictability, dependability, or flexibility, and are often independent of technology. Examples of such models are different sequential models such as Waterfall or V model, or iterative models such as spiral model, or different agile methods, or de-facto standards such as ISO 9000, or CMMI. Component-based software engineering (CBSE), as a young discipline is still focused on

technology issues: modelling, system specifications and design, and implementation. There is no established component-based development process. Yet many principles of component-based development (CBD) have significant influence on the development and maintenance process and require considerable modifications of standard development processes.
The main idea of the component-based approach is building systems from already existing components. This assumption has several consequences for the system lifecycle [15]:

- Separation of development processes. The development processes of component-based applications are separated from development processes of the components. Majority of the components should already have been developed and possibly have been used in other products when the application engineering process starts.

- A new process: A new, possibly separate process dedicated to finding and evaluating the components appears. Discovery and evaluation can be a part of the main process, but many advantages are gained if the process is performed separately. The result of the process is a repository of components that includes components' specifications, descriptions, documented tests, and the executable components themselves.

- Changes in the activities in the application engineering/development processes. The activities in the component-based development processes are different from the activities in non-component-based approach. For the application engineering process, the emphasis will be on finding the proper components and verifying them. For the component-level process, design for reuse will be the main concern.

Current technology limitations being addressed by the OPEN-SME project are:

1. Absence of component-based Application Engineering Process specifications that consider a cross- organisation software reuse environment: In the context of the OPENSME use cases, a component- based application engineering process should be cantered on the exploitation of the outcomes (use case models, feature models, software artefacts, architecture metamodels, etc) of an external domain engineering process. Indeed stateof-the-art approaches take into account activities such as component searching, discovery, and assessment as activities to be exercised in parallel to software product development processes. However, the market presently lacks a concrete Application Engineering Process specification that is dedicated to component-based software development (i.e. with clear partition and defined interfaces with the domain engineering activities) and considers the exploitation by reuse of external domain-specific components.

2. Limitations of existing software reuse repository solutions: The Reuse repositories are an essential factor for the success of any component-based and reuse-oriented application engineering effort, since they allow searching and retrieval of reusable software artefacts. The number of reuse specific tools is limited. More specifically, software dealing with component asset management is difficult to find, quite expensive and allows the sharing of only intra-company rather than inter-company components.

OPEN-SME use cases consider a cross-organisation environment that requires the effective exchange of information flows between the software reuse stakeholders (reuse engineers and resuers). Such communications may relate to the placement of software component orders, the provision of reuse feedback (e.g. bug reports), notifications on the publishing of new components, etc. In the context of large communities (as the ones considered by OPEN-SME) the exchanged information flows should be systematically structured towards facilitating and partly automating their organisation and processing by the software providers. Existing reuse repository systems do not tackle the requirement described above. As a result such cross-organisational

communications presently take place in an informal manner (typically using e-mails or forums), which significantly impacts the efficiency of the software reuse processes.

1.3.2.2 Overview

This includes description of domain engineering as well from the point of view of the application engineering process. The OPEN-SME Application Engineering Process will form a generic software development and lifecycle methodology that will be component-based, reuse-oriented and applicable (customizable) across different Application Domains. The purpose of this specification will be as follows:

- It will define in detail and in a concrete manner a set of software-lifecycle activities and associated work products
- It will be used as a functional and technical specification of the OPEN-SME Component Reuse Repository and Search Engine (COMPARE).
- It will provide guidelines on the use of the OPEN-SME Component Reuse Repository and Search Engine (COMPARE).
- It will also constitute one of the main topics of the project training activities and furthermore it will form a key project result to be disseminated to third parties.

The Application Engineering Process will comprise two streams of activities that will be exercised in parallel and in close synergy with each other as shown in Figure 9:

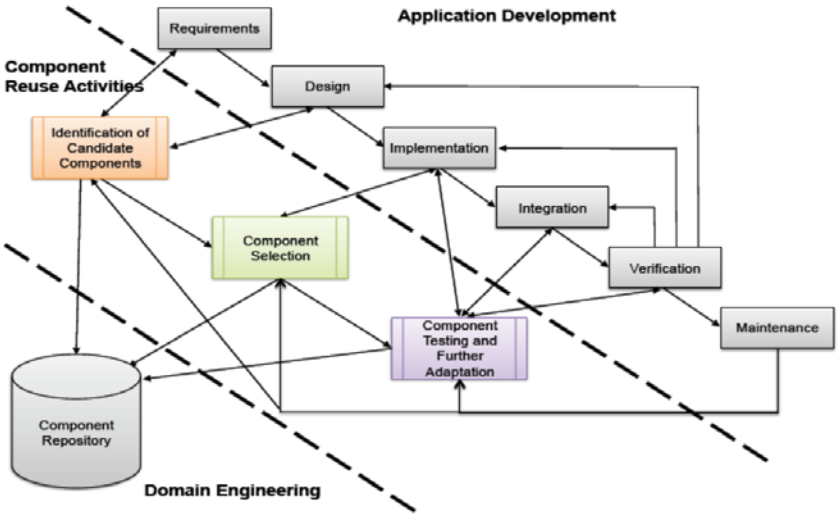


Figure 9: Overview of the reuse-oriented OPEN-SME Application Engineering Process

1. **Application Development**: This will comprise of the pertinent lifecycle phases during the application development. Starting with the simplest waterfall model as a base for this process, which can be extended to more iterative development processes (e.g. agile development processes).

2. **Component Reuse**: It will define a set of processes that concern the exploitation of the resources (software artefacts, software metadata, test suites, technical documentation) that the re-users will be able to access at the Reuse Repository.

The component re-use activities are typically done either at the Domain Engineering or as part of the Application Engineering, based on several factors such as business considerations, timing aspects, domains specific issues etc. As depicted in Figure 9, the component reuse activities make

use of the Component repository through the COMPARE tool as well as interact with the Domain Engineering.
Such a generic application engineering process however needs to be further detailed in to multiple phases, with clear distinctions between the phases in order to provide appropriate guidelines and tools to the SME developer for assisting them in achieving their goals of cost-efficient development of high quality software systems. However please note that there are considerable links between system development and component development phases. Also based on the domain of the application being developed (for example, enterprise or embedded), an activity at the lower level could be more appropriate to be included either as part of the Domain Engineering phases or as part of the Application Engineering phases. Further, a generic application process needs to be adapted based on the prevalent life cycle model (such as traditional V model, Agile, RUP etc.) being followed in a given SME organization.

1.3.2.3 PROPOSED APPLICATION ENGINEERING PROCESS

In the following subsections we define the OPEN-SME Application Engineering process in detail. Based on the type of application domain under consideration (whether embedded system or Enterprise application), the process will include a specific set of phases and activities from those described.

1.3.2.3.1 Inputs to the application engineering process

On a higher level, the inputs to the application engineering Process are a) the application requirements and b) available components produced by the domain engineering and stored in the reuse repositories. The application requirements either come as a specification in an order for product development or could evolve through discussions with domain experts and the system developer. Since components are the major inputs to the applications engineering process (as the assets stored in COMPARE), we provide more details on what a component contains and try to exemplify. Specifically, for each component, COMPARE will contain:

1. Classification of the components in relation to the domain concepts (see Section 6 “A Domain Ontology for Domain Representation” of D2.2).

2. Component Information:

    a. The source code of the component.

    b. Definition of one or more provided interfaces, which list the services the component type provides and definition of zero or more required interfaces, which list the functional services the component requires in order to operate correctly. An interface is a set of one or more operations, with a defined operation signature determined by an operation name and an ordered set of parameters, each one with a direction chosen between in, in out, out and a parameter type chosen between the defined types (Figure 10).

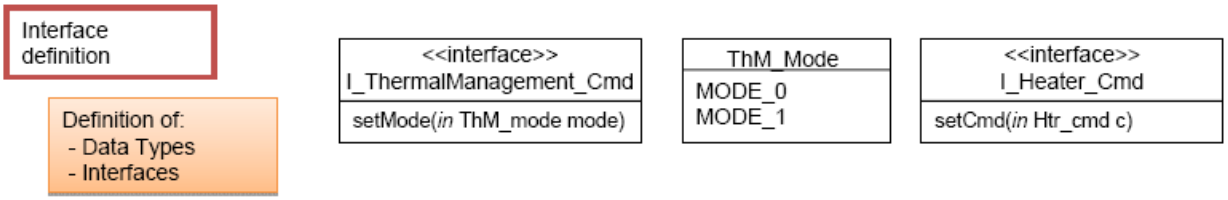


Figure 10: Example definition of data types and interfaces to be referenced in component implementations.

c. Definition of component attributes. Each attribute is typed with an already defined data type and has a set of modifiers defined at type level (read-only, read-write). An example is given in Figure 11. From the list of attributes and their modifiers we can automatically generate a set of operations (possibly in a dedicated provided interface) which operate as getter and setters for the attribute. In particular: (i) for read-write attributes we generate a getter and a setter operation; (ii) for read-only attributes we generate a getter operation.

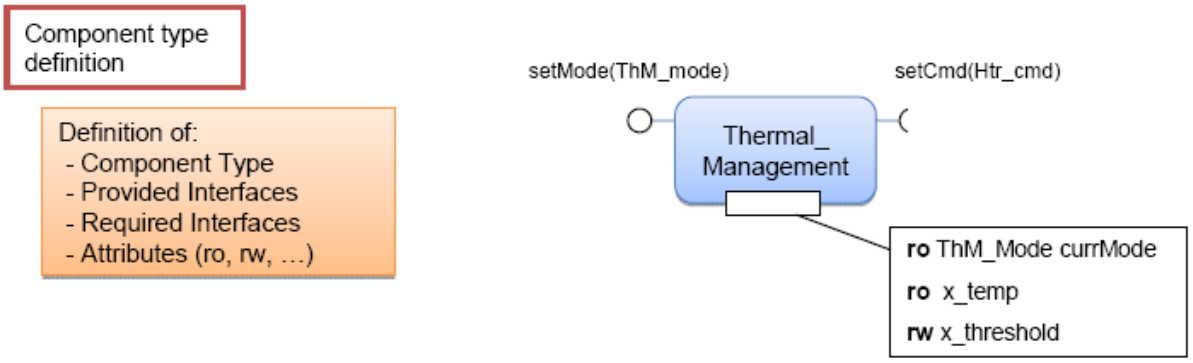


Figure 11: Example definition of a component's provided interfaces, required interfaces and attributes

3. Definition of platform constraints (like assumption on the processing unit or execution platform)
4. Packaging information (name of source code files, information on the generated object files, etc.)
5. Definition of non-functional constraints (some implementations may place some constraints on the correctness of their behaviour. For example a control law in an embedded system may work correctly only if executed within an interval of frequency, say 5Hz to 10Hz).
6. Additional information for operations (e.g. “threadsafe” or not, i.e. there is a need or no need to protect interfaces with mutual exclusion at instantiation level).

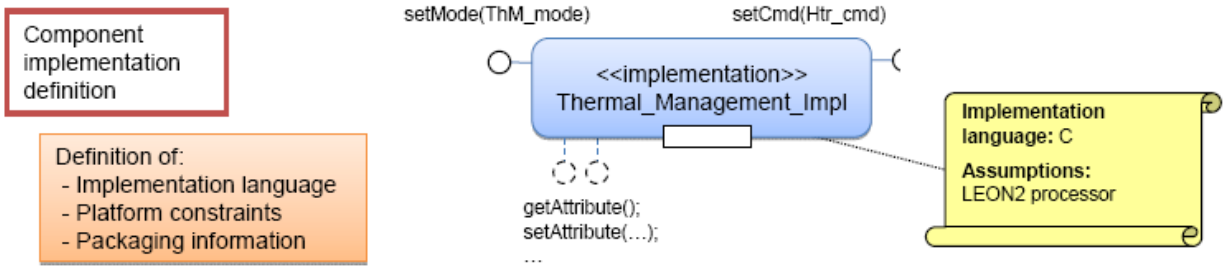


Figure 12: Example definition of a component implementation

1.3.2.3.2 Application engineering phases

The main phases of the Application Engineering/Development in comparison with “classical” software development and lifecycle phases, and in relation to the outcomes of the domain engineering activities (as described in Section B) are as follows:

P1. Application requirements phase
P2. Physical architecture definition phase

P3. Application Design Phase
P4. Implementation- Component Realization
P5. System integration phase
P6. System testing phase
P7. Release Phase
P8. Maintenance Phase

The above phases are described in detail in the following subsections along with the main activities, inputs, roles and outputs in each one of them.

Phase#1: Application Requirements

In a non-component-based approach the requirements specification is the main input for development of the system. In a component-based approach the requirements specification will also consider the availability of existing components. Within OPEN-SME, the requirements should correlate to the assortment of the components, i.e. the requirements specification will not only be input to further development, but also a result of the activities that took place during both the Domain Analysis and Domain Implementation phases. For example, certain requirements are not essential for a project and/or can be slightly modified in order to reuse as-is an existing component that is too difficult or too expensive to implement from scratch. However this search is more focused on internal component repository as well as the goal is to identify a set of candidate (potential) components by looking at the compatibility in a macro level.

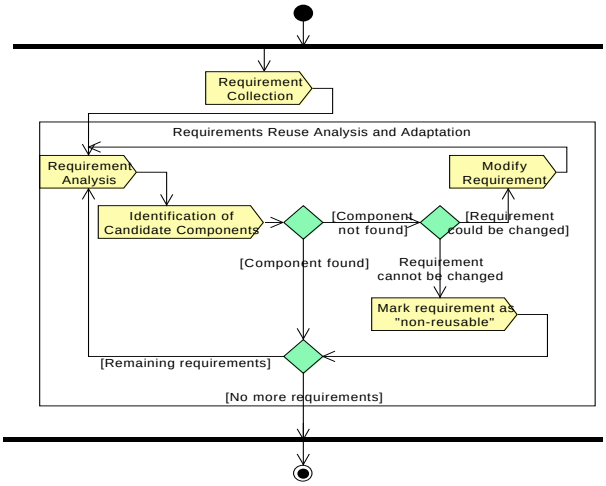


Figure 13: Overview of the Application Requirements phase

In this phase the reuser performs the following activities, depicted in Figure 13 (with some possible support from domain experts):

1. **Requirements Collection**: In this activity, the reuser collects and specifies the requirements for the application to be developed.

2. **Requirements Reuse Analysis and Adaptation**: This constitutes of the following sub activities:

   2.1. **Requirements Reuse Analysis**: In this activity, the reuser looks for potential candidates of components satisfying the requirement.

   2.2. **Requirements Modification**: If for a given requirement, no reusable components could be found, then the reuser together with the optional support of the domain experts decides

whether the requirement could be modified.

2.3. **Mark Requirement as 'non-reusable'**: If for a given requirement, no reusable components could be found as well as it could not be modified then the requirement is marked as 'non reusable'. This could ultimately result in a request to either domain engineering or to in-house development.

2.4. **Identification of candidate components (Iteration 1)**: In this phase the first iteration of the component candidates will be done. Later (in the design and implementation phase) the same activity will be repeated with slightly different goals. In this phase the goal is to find the candidates that might meet the component requirements found in the requirements analysis. The concrete support of this activity will be as follows. The Domain Feature Models given by the Domain Engineering Process will provide the first hints on what functionality is supported by the existing components. The Search Engine of the COMPARE tool will then be used for searching components on the basis of a multitude of criteria ranging from desired features (functionality) to programming languages, execution frameworks (e.g. J2EE, .NET), etc. In the first iteration, the specification of the components do not need to be on a detailed level; for example the interface functionality (i.e. operations) can be specified, but not necessary all parameters of the operations, (i.e. the operation arguments). The result of this activity will be a set of components that might meet the requirements. In the case that a set of component that fully matches the requirement of the reuser cannot be found, then the resuer will be able to place a relevant order on the Reuse Repository. If no component was found, the information will be forwarded to the Domain Engineering process with a possible order for such components. This can prove particularly effective for the cases whereby the desired features are supported by some existing components, however the desired execution framework or programming language is not supported and therefore some type of component packaging or further adaptation is required.

Inputs:	High level description of the requirements of the application to be developed Or this gets evolved during this phase inconsultation with Domain experts.
Iterative:	Yes
Roles:	Reuser, Domain Expert (optional)
Tools:	COMPARE
Assistive Tools:	Requirements Engineering tools
Product Outcomes:	1. List of requirements (original & re-defined) 2. A set of candidate components satisfying them 3. List of requirements that do not have any reusable components

Table 1: Summary of the Application Requirements Phase

Phase #2 –Physical architecture definition

The role of the Physical Architecture is to provide a model-level description of the relevant hardware of the system. A physical architecture specification can assist in decision making during component search and selection. Also this can later on get refined based on the software reference architecture. In the physical architecture the following elements are described:

1. Processing units are units that have a general-purpose processing capability.
2. Equipments / Instruments / Remote terminals
3. The interconnection between the elements above, in terms of buses or point-to-point links, etc.

All the elements above should be decorated with a set of attributes that are relevant for analysis or code generation purpose. In the case of buses, point-to-point links and equipment, the elements should be decorated also with attributes to be able to drive an automatic generation of communication code.

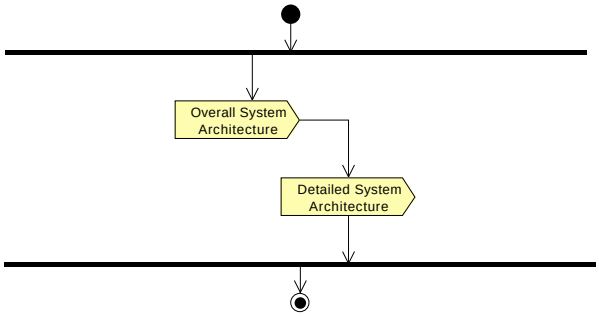


Figure 14: Overview of the Physical Architecture Definition phase

In this phase the reuser performs the following activities (with some support from domain experts):

1. **Overall system architecture**: This is based on the requirements on CPUs/nodes, memory, busses,
2. **Detailed system architecture**: This evolves through refinement of the overall design by including details, dependencies, constraints on the types of devices, platform, etc. This comes from the requirements and also based on the information from Domain Engineers/Domain Experts. The set of candidate components found during the requirement analysis phase has a valuable role to play during detailed system architecture modelling. The domain engineer can provide identification of the potential target platforms. For instance, if majority of the candidate components run on a specific platform, then this could as well be a deciding factor from a business perspective.

Inputs:	High level description of the requirements of the application to be developed
Iterative:	No
Roles:	Reuser, Domain Expert (optional)
Tools:	COMPARE
Assistive Tools:	-
Product Outcomes:	1. Platform definition 2. System architecture specification 3. System model decorated with attributes for analysis

Table 2: Summary of the Physical Architecture Definition Phase

Phase#3: Application Design

The OPEN-SME application design phase will follow the same pattern as a design phase of software in general; it will start with a system analysis and a conceptual design providing the system overall architecture and continue with the detailed design. However, a major deviation from traditional approaches will be taken as the system architecture will need to adhere to the Domain Software Architecture and incorporate assemblies of the existing components stored in the Reuse Repository. As in the requirements process, a trade-off between desired design and a possible design using the existing components must be analysed. In addition to this, there will be many assumptions that must be taken into consideration: For example, it must be decided which component model(s) will be used, which will have impact on the system architecture as well as on certain system quality properties.

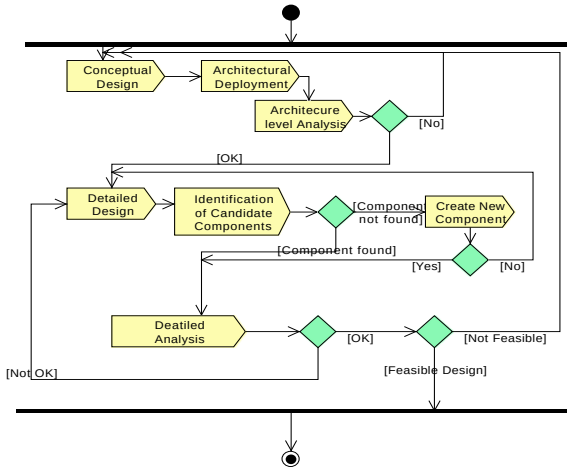


Figure 15: Overview of the Application Design phase

In this phase the reuse Engineer performs the application design and analysis through the following activities:

1. **Conceptual Design**: In this activity the reuser identifies the overall software architecture. Identification of the subsystems or subsystems built from architectural components will be the focus in this activity. By architectural components we refer to the units of some main service of the application.

2. **Architectural Deployment**: This activity will decide on a high level which component will run on which nodes/platform etc. Here the main considerations are specific requirements and constraints arising from the candidate components and the platform architecture definitions.

3. **Architectural level Analysis**: Aspects, which have wider system level implications, are addressed and analysed in this activity. For instance the fault tolerance requirements (dual vs multiple redundancies), separation of concerns, physical isolation requirements etc., are typically analysed during architectural level analysis. The results from the architectural analysis are used in detailed design activity.

4. **Detailed Design**: This activity will include design of subsystems, breakdown to architectural components, identification of components etc. This will also include specification of behaviours, sequence diagrams and state diagrams. Specification of components includes specification of interfaces. This will be an iterative activity. Either selecting the existing interfaces of components, or the specifications of component to be developed will be used in the detailed design.

5. **Detailed Analysis**: Based on the system requirements most of the model level analysis with

respect to extra functional properties will be performed at this stage to analyse the design soundness. Resource analysis, timing analysis, reliability modelling etc., are some of the typical analysis needed and there exists a large set of tools and techniques for performing these analyses. The exact choice of the tool for a specific type of analysis is not the focus of the OPENSME project. The results of the detailed analysis will be checked against the specifications. If they are not satisfactory, the possibility of generation of new components will be explored with the support of the domain engineer or as in-house development. If the resulting detailed design turns out to be an infeasible one, the one has to re-start from the conceptual design activity.

Inputs:	Application requirements (from phase 1)
Iterative:	Yes
Roles:	Reuser
Tools:	COMPARE
Assistive Tools:	Design and modelling and analysis tools
Product Outcomes:	1. System design (architecture) 2. Architectural components 3. Refined set of candidate/implementable components 4. Analysis results

Table 3: Summary of the Application Design Phase

Phase#4: Implementation- Component Realization

The component realization activities will only partially consist of coding – actually the more pure a component-based and reuse-oriented approach is achieved, the less coding will be needed. The main emphasis is put on component selection and its adaptation into the system. This process can require additional efforts. First the selection process should ensure that appropriate components have been selected with respect to their functional and extra-functional properties. This requires verification of the component specification, or testing of some of the component's properties that are important but possibly not documented in the Reuse Repository.
Provided that the system architecture adheres to the Domain Architecture the effort required for the adaptation of components (from the resuer perspective) will be very small or ideally zero. In any case, using the already tested and documented components from the COMPARE reuse repository will significantly reduce the burden on the reusers.

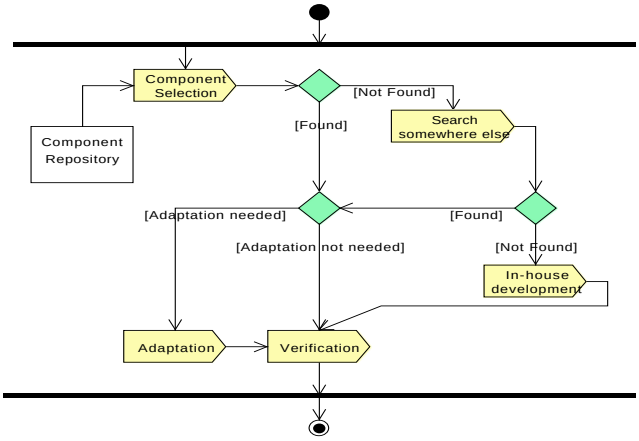


Figure 16: Overview of the Component Realisation phase

In this phase the resuer will perform the following activities:

1. **Component Selection** – the reuser selects the most appropriate components between the component candidates from the domain component repository. The existing components that are closest to the component specification from the design phase will be selected. Note that this specification considers both functional and non-functional properties. Note also, that the selection process does not only consider the component candidates, but also different component versions and variants. The candidate components found, will be compared and ranked. Appropriate COMPARE user interfaces will assist this procedure. A component that is most suitable for the given requirements and constraints will be selected. The ranking of components will be maintained throughout system development such that alternatives for a function can quickly be found. The selection of the components may result with the following cases for each component: a) the selected component fits well to the specification; b) the selected component fits partially to the specification, but there are some mismatches, functional or non-functional, - in that case an adaptation of the component is needed; c) There is no component that matches the specification from the design phase. In that case this components should be searched for outside the domain, or internally developed.

2. **Component adaptation** – When a particular component has been selected it may happen that it does not comply with the specification (either functional or non-functional properties). These components should be adapted to meet the specifications. A simplest form of adaptation is to creation of adapters. Adapters are mediators between components with a goal to make the components compatible. A typical adapter will change type of the interface but not the interface itself. A next level of the adaptation is s.c. *wrapper* – a new "component" that adjust the interface of the selected component with the component specification from the design phase. Wrappers can be used to add or remove some parts of the interface, or to change its behaviour, so this may require some programming efforts. Note however that in both adapter and wrapper cases the selected component has not been changed. The most drastic type of adaptation is the change of the component. The resuer modifies the component for the specific needs of the application. In this case a new version of the component will be created. In some cases the resuer can send a requirement to the Domain Engineer to perform the adaptation, if the new adapted component version is suitable for resuablity. In some cases the resuer will do the changes himself, but the result will be forwarded to the domain engineering.

3. **In-house development** – in some cases no components for a specific service will be found in the domain repository. In some cases the company developing the application encapsulates its business advantage and do not want to share this knowledge with the domain or other competitors. This implies that the application engineer (the reuser) will develop specific components – using the application development tool. In other cases the resuer will develop the component, but will also share it in the domain. In that case the resuer will send the component (specification and implementation) to the domain engineer who will probably improve the component with respect to its reusability.

4. **Component verification** – when a component is selected and adapted according to the requirements from the design phase, or when developed, it must be verified. This verification corresponds to a unit test, so it includes the verification of the functional properties. In addition some of the non-functional properties can also be verified (for example memory size, response time, and other component attributes). Despite the fact that components will have been verified by the reuse engineers during the Domain Implementation Phase, it is very probable that the reusers will also need to test components themselves towards/after integrating them in their systems under development. The Test Suite Implementations and/or

Abstract Test Suite Specifications that will be available at the Reuse Repository will be exploited for assisting this procedure. The first level of verification will include testing functional and certain extra-functional properties of a component in isolation (unit testing). A second level of verification includes testing the component in combination with other components integrated in an assembly (integration testing). The reusers will be able to provide structured feedback (e.g. bug reports, and orders for its fixes) on the implemented components. This type of orders and feedback is very important, since it will allow the repository growth over time with many variants of existing components, suitable for different environments or with slightly different functional and/or quality properties.

Inputs:	System conceptual and detailed design Physical architecture specification Components candidates
Iterative:	Yes
Roles:	Reuser, Domain Expert (optional)
Tools:	COMPARE
Assistive Tools:	Application development tools, test tools, error reporting system
Product Outcomes:	1. Components ready to be integrated into the system 2. Adapters and wrappers needed for the components adaptations 3. New components – candidates for the Domain Engineering 4. New component versions – candidates for the Domain Engineering

Table 4: Summary of the Implementation-Component realization Phase.

Phase #5 – System integration

This phase includes activities that support integration of the selected, or the newly created components into the application. In the component-based approach this phase, although consists of many complex activities, most of them are integrated parts of many component technologies and are done automatically or semi-automatically. Further, we refer to two types of integrations: a) integration of a set of components into assemblies that constitute a service or a subsystem or an architectural unit, and b) the entire system. Also the integration can be completed a) before the deployment of the application, but also b) after the deployment of the application – when a new component is deployed into the application during run-time (a well-known “plug-and –play” component deployment). See Figure 17.

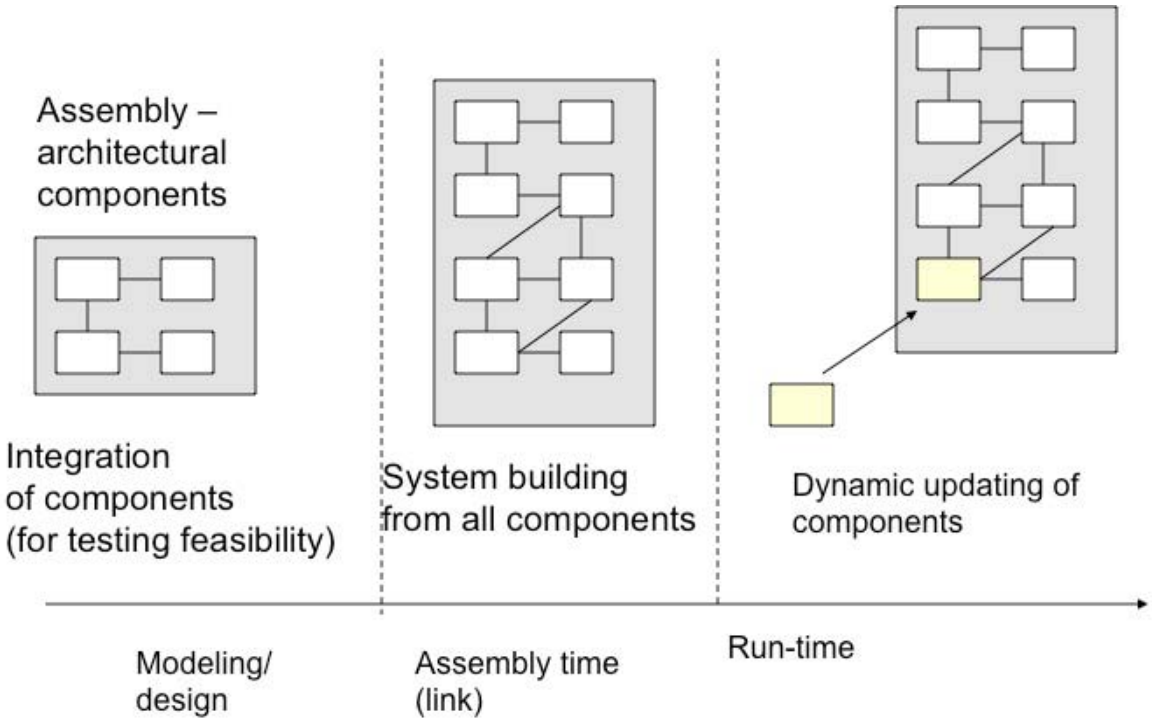


Figure 17: Integration in different phases

The list of the integration activities is depicted in Figure 18 and detailed descriptions are as follows:

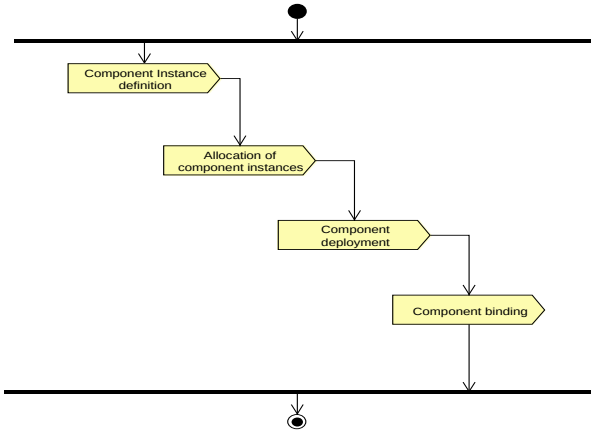


Figure 18: Overview of the Integration Phase

1. **Component Instance definition** – Component instances are defined from selected or new component implementation. The component instance is the component entity that gets concrete values of functional and non-functional properties. For example, a component can have parameterised interface, which in the instantiation process gets some concrete values. Similar is with some of the non-functional properties (for example static memory size). Instantiation with variant properties is a common use in product line engineering approaches. Figure 19 illustrates a component instantiation where the component is annotated with concrete values of the attached attributes. Note that some of the attributes depend on the physical platform the component will be deployed (which is specified in the phase 2 – physical architecture).

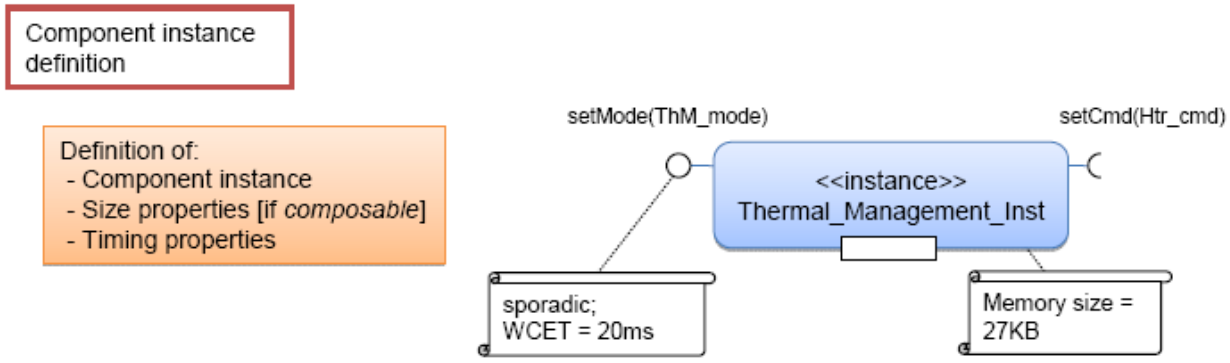


Figure 19: Declaration of component instances

2. **Allocation of component instances** – The allocation of the components is supposed to be done in the design phase. Here, according to that input, the component instances are allocated on the physical structure - by this the component instances get the concrete values of some properties. Instances of components are allocated to processing units defined in the physical architecture. The need for explicit allocation of component instances is necessary when two or more processing units are defined in the physical architecture. In the vast majority of cases, given two components allocated on distinct processing units, it is straightforward to deduce the allocation of the bindings between them. In fact typically there is only one bus or point-to-point link that connects the two processing unit. However, there can be the case were there are more connections between the units.

3. **Component deployment** – this is the activity that integrates the component into the application – i.e. it creates a connection to the underlying platform, middleware or component containers. This is usually a matter of the component technology. Containers are special type of the components/wrappers that are carriers of certain properties (for example they implement authentication mechanisms that are activated when the components from that container are being accessed. Containers enable connection to the middleware and indirectly to other containers and components. The provided and required interfaces of the container match the interfaces of the components. As carriers of certain properties the containers are often means for management of non-functional properties (aka extra-functional properties – EFP). This management is related to runtime EFPs and realised in combination of components and underlying component execution platform that can often be integrated as a part of a middleware. We distinguish four types of support (see Figure 20): (i) Exogenous Management. The EFP management is provided outside the components, (ii) Endogenous Management. The EFP management is implemented in the components, i.e. the component developers are responsible to implement it; (iii) Management per Collaboration. The EFP management is realized in direct interactions between components; (iv) System-wide Management. The EFP management is provided by the component framework, or underlying middleware. By a combination of these types we get four possible types of the EFP support:

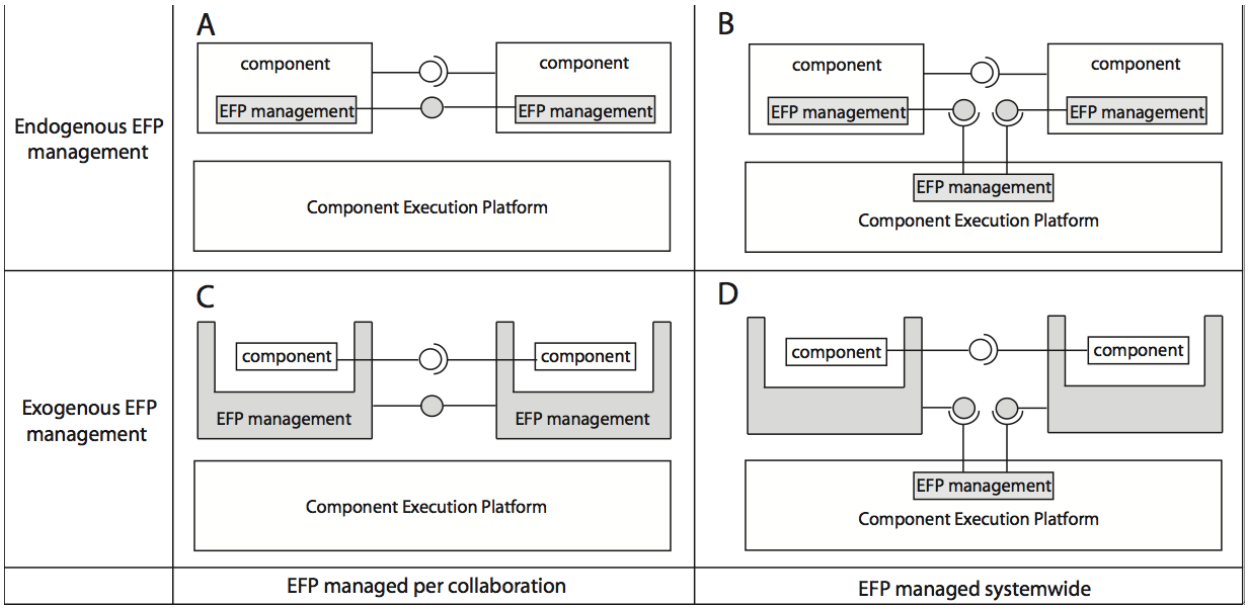


Figure 20: Support for Management of extra functional properties

- *Approach A (endogenous per collaboration).* A component model does not provide any support for EFP management, but it is expected that a component developer implements it. This approach makes it possible to include EFP management policies that are optimized towards a specific system, and also can cater for adopting multiple policies in one system. This heterogeneity may be particularly useful when COTS components need to be integrated. On the other hand, the fact that such policies are not standardized may be a source of architectural mismatch between components. A risk of using this approach is heterogeneity of policies for handling a single EFP in a system. As a result, managing and predicting emerging properties at the system level can be very difficult.

- *Approach B (endogenous system-wide).* In this approach, there is a mechanism in the component execution platform that contains policies for managing EFPs for individual components as well as for EFPs involving multiple components. The ability to negotiate the manner in which EFPs are handled requires that the components themselves have some knowledge about how the EFPs affect their functioning. This is a form of reflection applied to EFP management.

- *Approach C (exogenous per collaboration).* In this approach, components are designed such that they address only functional aspects and are oblivious to EFP. Consequently, in the execution environment, these components are surrounded by a container. This container contains the knowledge on how to manage EFPs. In this approach, containers are connected to other containers. Connected containers can manage the EFPs for the components that they encapsulate. The container approach is a way of realizing the separation of concerns in which components concentrate on functional aspects and containers concentrate on extra-functional aspects. In this way, components become more generic because no modification is required to integrate them into systems that may employ different policies for EFPs. Because these components do not address EFPs, they are simpler to implement. A disadvantage of the container approaches might be a degradation of the system performance.

- *Approach D (exogenous system-wide).* This approach is similar to approach C, except that the system can coordinate the management of an EFP from a global system-wide perspective (e.g. global load balancing). Consequently, a more complex support need to

be built into the component execution platform.

4. **Component binding** – this is the activity in which a component implements connections to other components (components binding). Component bindings are established between component instances. The binding is established between the required interface of a component instance and the provided interface of another component instance. The binding is subject to a static check to ensure that the candidate provided interfaces fulfils the functional needs of the client required interface. This can be done by asserting the compatibility of the two interfaces (as shown in Figure 21). An alternative approach instead does not rely on the signature of operations (name of operation, ordering, type and direction of parameters), but the compatibility of two interfaces is checked ignoring the names of interfaces (and operations therein) and just asserting the compatibility of the types and the direction of the parameters of the operations. If the binding is considered legal according to this binding approach, a later step requires that when the required interface is called, the call is dispatch to the correct operation in the bound provided interface. The signature of the calling operation (in the RI) and the called operation (in the PI) in fact are different. Arguably, the connector is in charge of performing this step and a tool support should help the configuration of the connector to perform this kind of binding. When bindings have been established, it is possible to complete the description of the instances with synchronization properties, queuing properties (like queuing protocols and queue sizes), non-functional properties (like Minimum Inter Arrival Time) and end-to-end timing properties.

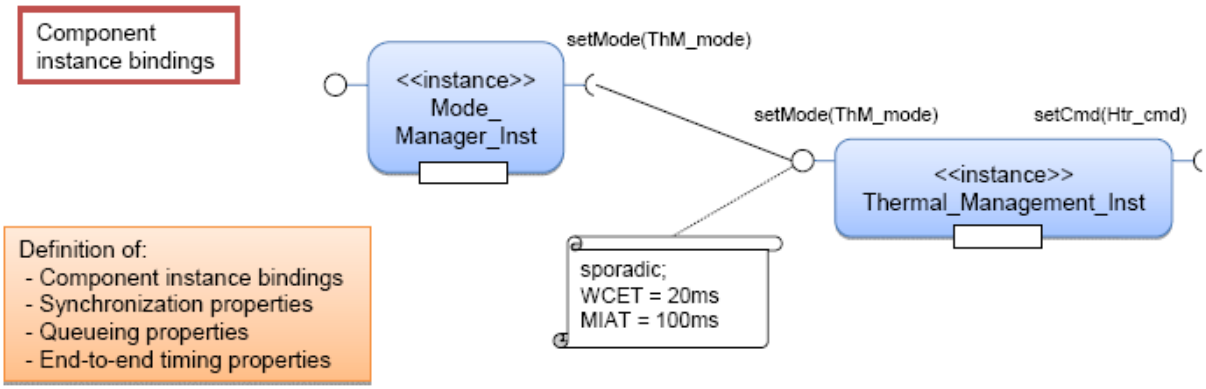


Figure 21: Definition of component bindings

In distributed applications, and in the applications with dynamic binding (plug-and-play) special types of connectors can be created – proxies that play a role of components and transparency in the application development. The proxies, as well as some adapters and containers can be automatically created by the tools, but also saved in the Domain engineering repositories if they are typical domain-specific solutions.
Table 5 shows input/output and the tools used in this phase:

Inputs:	1. Components ready to be integrated into the system 2. Adapters and wrappers needed for the components adaptations 3. Physical platform detailed specification
Iterative:	Yes
Roles:	Reuser, Domain Expert (optional)
Tools:	COMPARE
Assistive Tools:	Application development/integration tools

Product Outcomes:	1. Component assemblies ready for the verification 2. The application

Table 5: Summary of the Application Integration Phase

Phase #6 – System testing

Due to the fact that the tests that will have been performed in isolated components are usually not enough, since their behaviour can be different in the assemblies and in other environments, thorough system and subsystem tests will need to be performed. In case of embedded systems, multiple levels of verification and validation often need to be performed using simulations, hardware-in-loop, etc., before the system can be deployed in actual operational environments. In the waterfall model the test is performed after the system integrations, whereas in CBD Tests are present in all phases. Tests are performed on isolated components (unit testing), component assemblies and finally on the system
In this phase the developed system is verified against the system specification. This is also known as testing in the large and proves the systems readiness for deployment. Any system failure or abnormal behaviour will lead to debugging and bug fixing activities. The structuring of the system test suite and logging of test results should be performed in such manner to facilitate the reverse traceability of a failure to a fault (bug) in a specific component. Once a bug has been associated with a specific component, then bug fixing can be attempted either in-house or with the help of the domain engineer. In any case the rectified component is stored back in the repository as well regressing testing is performed on the modified system.

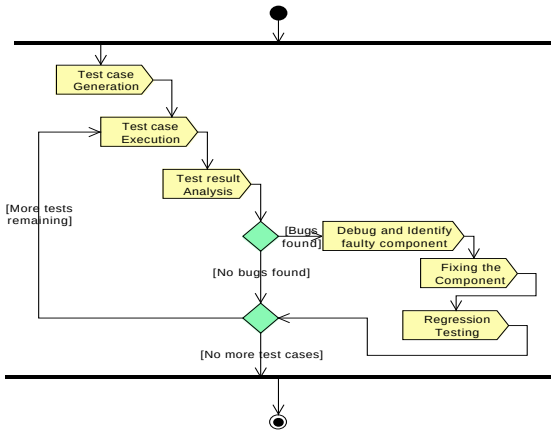


Figure 22: Overview of the System Testing phase

The major activities performed during the System Testing phase are as follows:

1. **Test case Generation**: – As a starting activity reuser is generating a set of test cases or test suite from the system requirements specification.

2. **Test case Execution**: – In this activity reuser is executing a predefined set of test cases or test suites on a complete system in a setting that is as close as possible to the real environment.

3. **Test results Analysis**: – Upon executing test cases, reuser has to perform analysis of test results to compare if the results are as expected by the system requirements. This activity will define if some specific behaviour of components should be considered as a fault of the system or not.

4. **Debug and identification of faulty component**: – For each fault in the system identified

during test results analysis, reuser has to perform debugging activity in order to locate the faulty component.

5. **Fixing the faulty component**: – Once the faulty component is identified reuser can modify the component in-house or collaborate with domain engineering in obtaining a new version of component.

6. **Regression testing**: – When a new component is obtained reuser has to perform regression testing in order to validate that the bug previously identified is addressed but also to ensure that new bugs are not introduced in the system with a new version of component.

Inputs:	Application requirements Integrated application
Iterative:	Yes
Roles:	Reuser, Tester
Tools:	COMPARE
Assistive Tools:	Testing tools, Test Management Systems
Product Outcomes:	1. Modified components – candidates for the Domain Engineering 2. Test Suites 3. Verified and deployable release of the application

Table 6: Summary of the System Testing Phase

Phase#7 - Release Phase

The release phase includes packaging of the software in forms suitable for delivery and installation. The component-based development release phase will not be significantly different from that of a "classical" software development process. The major activities performed during the Release phase are as follows:

1. **Deployment**: –The release is deployed on the specified target platform.

2. **Release Certification**: – In this activity reuser will execute a predefined set of test cases or test suites on the deployed release of the system. Once the verification activities (mainly functional ones) are completed, the application is certified for release.

Inputs:	Application Release Target Platform Release Test suites
Iterative:	No
Roles:	Reuser, Tester
Tools:	-
Assistive Tools:	Testing tools, Test Management Systems
Product Outcomes:	Certified and deployed release of the application

Table 7: Summary of the Release Phase

Phase#8- Maintenance Phase

The maintenance of a software system is a necessity mainly due to the changes of the environment that the software operates in. Even if a system functions properly, as time goes by, it has to be maintained. The approach of a component-based development process is to provide maintenance by replacing old components by new components or by adding new components into the systems. The paradigm of the maintenance process is similar to this for the development: Find a proper component, test it, adopt it if necessary, and integrate it into the system. These activities are essentially those discussed earlier as part of component realization and hence are not repeated here.

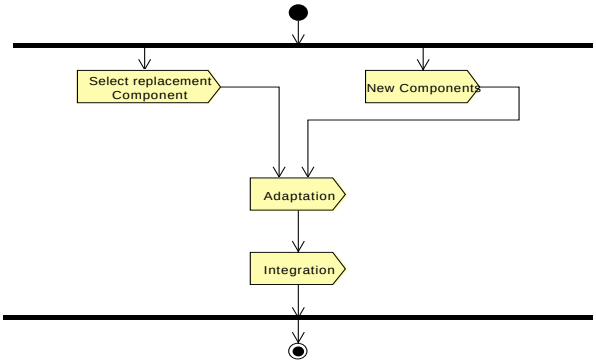


Figure 23: Overview of the Maintenance phase

The major activities performed during the maintenance phase (Figure 23) are as follows:

1. **Selection of Component to replace**: –The decision for replacement of a component could be based on several factors. Limitations in performance of the current system, dependencies to other modifications or updates in the target platform or middleware, or even the release of a new upgraded version of a component could trigger this activity.
2. **Component Adaptation:** This will be same as in previous cases.
3. **New Component development**: Same as in previous phases. This can be either ordered from the domain engineering or developed in house.
4. **Component integration:** Same as in System Integration phase

Iterative:	No
Roles:	Reuser
Tools:	COMPARE
Assistive Tools:	--
Product Outcomes:	1. New version of the system 2. New/Updated components stored in COMPARE

Table 8: Summary of the Maintenance Phase

1.3.3 OCEAN

Source code search engines assist the software development process by providing a way of searching for free source code in code repositories. Although their use is rather straightforward, there exist a few of them and the differences in the way they index and provide access to their assets require considerable time and effort from the programmer to use them. This Section describes OCEAN, a federated open source code search engine, that simultaneously asks, in real time, existing open source code search engine sites and detail the way we overcome the

integration obstacles, by combining provided APIs, browser automation and web content extraction techniques.

1.3.3.1 Introduction

The concepts of Software Reuse [**16**] and Rapid Development have been adopted by large software development companies, small and medium enterprises (SMEs), research institutes and freelancers. According to a survey conducted in [**17**], software reuse in general and Free/Libre Open Source Software (F/LOSS) reuse in particular are important for the software development SMEs for a series of reasons:

- Reuse has a positive effect on lowering the development costs (91%), shortening the development and testing time (83%), increasing the quality of the final product (76%) and shortening time to market (72%).
- In relation to the different artifacts that can be reused, source code is the most important (87%), followed by design (80%) and documentation (75%).
- Almost half of the organizations (51%) have an in-house reuse repository whereas 39% have some formal process for reusing components they develop.
- The vast majority of the respondents (80%) said that their organization supports OSS reuse.

Meanwhile, millions of lines of reusable code have become available in the different source code forges and, in many cases, lots of alternatives exists for specific functionality [**18**]. This availability (and "redundancy") uncovered the need for effective ways of discovering reusable source code.

Source code forges (SourceForge, Git, Bitbucket, etc.) provide ways for internal navigation and search (such as categories, tags and internal search engines). For the reuse engineer however, who's primary goal is to find the component that best fits the needs of the functionality he wants to implement, searching to each source code forge separately, creates a significant overhead. This is also captured in [**17**], where the most important factors preventing OSS reuse were the lack of documentation (80%), the uncertainty on the quality of OSS components (76%), and the difficulty in searching and retrieving OSS components (66%).

Web-based source code search engines follow the architecture of classic, web search engines. Despite the differentiation of the nature of their data (that is, source code files), they provide crawling, indexing, reporting and ranking mechanisms identical to those of a typical web search engine. This is probably the reason why only 12% of the responders in [**17**] said that they have used a specialized OSS code search engine. They seem to prefer general purpose web search engines instead (e.g. Google). In fact, general purpose web search engines contribute more reusable components than specialized code search engines (65% vs. 31%). More significant is the fact that this 31% comes from the aforementioned 12% [**17**]. It seems like there is quality in the specialized OSS code search engines. The above observations suggest that the current status of specialized OSS search engines leave much to be desired for the developers [**19**], since although they can be potentially an important source of reusable components, the developers do not view them as important enough to use them. Finally, one cannot overlook the diversity of important sources of reusable components: in-house and public code repositories, specialized (code) and classical search engines. This, together with the difficulty reported earlier in searching and retrieving OSS components asks for a search mechanism able to provide results collectively, from different free/open source code sources.

Focusing on website-based code search tools, Krugle [**20**] and Koders [**21**] are among the most popular. They host source code on which they provide search services and also index other forges like Sourceforge. Merobase [**22**] can be mostly described as a code meta-search engine since it does not own a code repository but rather indexes and collects metadata from other sources on which it provides search services. Moreover it defines itself as component oriented search engine, meaning that, it can return sets of source code classes that implement a specific functionality. These specialized code search engines are valuable but each one poses specific requirements to the user, like searching using a diversity of search forms with different criteria in each or interpreting differently presented results. This definitely creates cognitive overhead to the end user. Even availability is sometimes questionable and thus a source of

frustration. Finally, dealing with more than one search points is more time consuming. Eventually, the "Google solution" becomes more attractive and the findings in [**17**] get justified!

Software reuse in general and OSS reuse in particular is important for the software development SMEs. To alleviate the problems mentioned and make the use of web-based code search engines more attractive, we propose a federated code search engine that provides the user with a single point to define his criteria-based search query, propagates the question to other code search engines in real time and finally presents the aggregated results to the user ([29, 30]). The proposed architecture can cooperate with individual code search engines either through an API or by using browser automation and web content extraction techniques.

1.3.3.2 OCEAN High Level Design

The federated code search engine we propose should be flexible enough to incorporate individual existing or future code search engines. Typically, one can retrieve data from another web source either through an API or via web content extraction. Having an API is preferred because it is faster and more reliable. Merobase belongs to this case. However, in cases like Koders and Krugle, which give their answers as http pages only, web extraction is the single option. Web content extraction is the non-trivial process of collecting unstructured web data and storing them in a database or an XML file [31]. This is usually accomplished by pattern matching an html pattern (extraction rule or wrapper) with a target web page. Upon a successful match, a data record becomes available as data from the web page is unified with variables in the extraction rule. Tricky cases like record-data scattered on different html sub-trees, pages with more than one data records, data records distributed in many web pages as a result of some pagination procedure, incomplete data records that break the html pattern used, etc., make the extraction task non-trivial. Flexibility in extraction rule management by means of visual/GUI tools, deployment and orchestration (use many extraction rules in a cooperative fashion) are all required features from a web extraction solution that we want to last long. The deployment diagram of the proposed system is depicted in Figure 24. Queries submitted via the single search form provided, are forwarded as http calls to one or more query services (this is a user preference) utilized by the query engine. Each of these services forwards the query to its own code search engine, collects the results in XML format and sends them back to the main system where they are collated and presented in HTML to the user. The system aims at reducing the time and effort required by a user to search all the individual search engines, offering a transparent search solution. It does not perform any actual asset indexing or search by itself. A prototype, namely OCEAN [**25**], of the federated code search engine described in Figure 24 has been implemented, in the context of the OPEN-SME project.

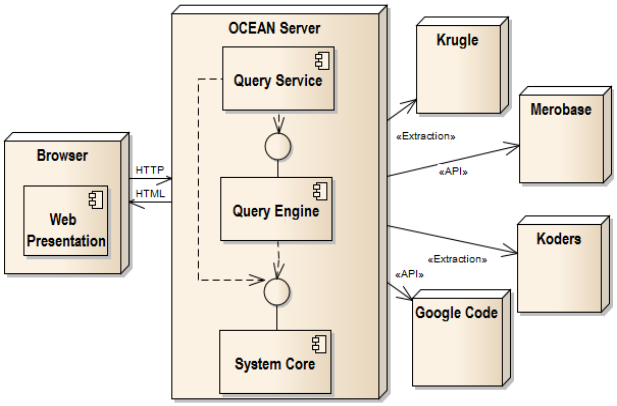


Figure 24: Deployment diagram of the federated code search engine.

1.3.3.3 Implementation Details

In this section, we give implementation details of the Query Engine subsystem (Figure 24), which actually implements the federation. It supports two types of foreign search engine integration: API-based and

Extraction-based.

1.3.3.3.1 API-based Integration

Merobase

Merobase [**22**] integration belongs to this case and was implemented by means of a JAR search client provided by the Merobase creators. A Perl web service was written utilizing this API and returning the results for a user-specified query in a suitable XML format. The Merobase API supports 2 parameters: *s* for the search keyword and *n* for the number of results requested. An upper limit of 30 results per query has been set by Merobase developers. An example http call the OCEAN sends to call this service is:

http://<*system*>/cgi-bin/merobase.pl?s=java&n=25

It is clear that, adding another API based search engine into the OCEAN's federation, is just a matter of pipelining its API with OCEAN's search form by means of a Perl script (as merobase.pl does in the example above).

Google Code Search

Google Code Search was integrated through its API. It turned out though that soon after the integration Google announced that the service will be no longer available. This is a nice example of the value of a federated search that continues to serve its users even though some sources are not available. Given the situation described, we do not give further details on this case.

1.3.3.3.2 Extraction-based Integration

When APIs are not available, web extraction does the integration. This requires the availability of an easy to use, robust and flexible web content extraction framework. DEiXTo was the tool of choice. It is briefly described right after.

DEiXTo – A web content extraction framework

DEiXTo [**27**] is a powerful web data extraction tool that is based on the W3C Document Object Model (DOM). It provides the user with an arsenal of features aiming at the construction of well-engineered extraction rules that describe what pieces of data to scrape from a website. DEiXTo consists of three separate components: a) *GUI DEiXTo*, implementing a friendly graphical user interface that is used to manage extraction rules. b) *The Command Line Executor* (CLE for short) massively applies wrapper project files built with GUI DEiXTo, on the desired web pages. CLE is actually a specialized instance of DEiXToBot. c) *DEiXToBot* is a Perl module aiming at tailor-made scraping and browser automation solutions. It facilitates the combination of multiple extraction rules as well as the post-processing of their results through custom code. Therefore, it can deal with complex cases and cover more advanced web scraping needs at the cost of the programming skills it requires.

Since there was no API access available for Koders [**21**] and Krugle [**20**], DEiXTo-based wrappers were successfully deployed in order to enable the extraction of the *N* first results returned from these search engines.

Koders

Koders [**21**] integration was smooth, in the sense that the html result pages were fully accessible by DEiXTo. The service supports 4 URL parameters: *s* for the search keyword, *li* for license type, *la* for language and *n* for the number of results requested.

Krugle

Krugle [**20**] integration on the other hand raised some difficulties mostly due to the heavy use of AJAX calls in its search results pages. Currently, DEiXTo does not support JavaScript automation. As a result we used Selenium [**28**] which actually automates a Firefox instance and were able to get Krugle's HTML

results properly, and then forwarded them to DEiXTo for the actual extraction. Again, OCEAN sees Krugle as a web service supporting 4 URL parameters: *s* for the search keyword, *pro* for the aiming project, *lic* for the desired code license and *n* for the number of results desired.

1.3.3.4 The System in Use

The main screen of the search facility of OCEAN consists of the form depicted in Figure 25. In the textbox entitled "Search" the user can specify the keyword(s) of his search, separated by spaces. The search space can then be narrowed down by using the three combo boxes labelled language, license and type, respectively. *Language* refers to the programming language of the source code the user wishes to retrieve (e.g. Java, Perl, PHP, etc.). *License* refers to the type of the license under which the source code retrieved has been initially published. Finally, *type* refers to the type of the file the user is looking for. This type can be *class*, *interface* or *enum* (enumeration type). If any criteria do not apply to some search engine, they are simply omitted.

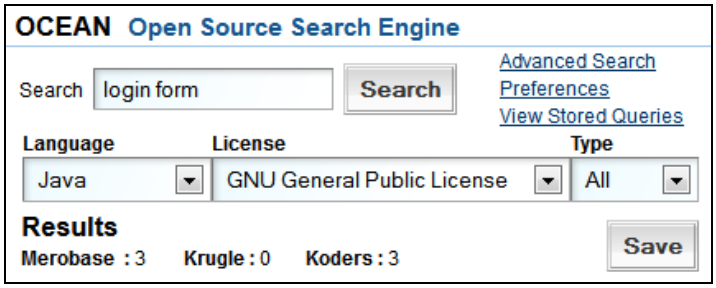


Figure 25: OCEAN's Search form in action

OCEAN provides a set of preferences allowing the user to customize the service to her own needs. In preferences (Figure 26), the user can review his account details (if he is a subscribed user) and manage his account credentials and saved queries. He can also set the number of results per search engine OCEAN is going to return as well as the individual search engines he would like to include in his query.

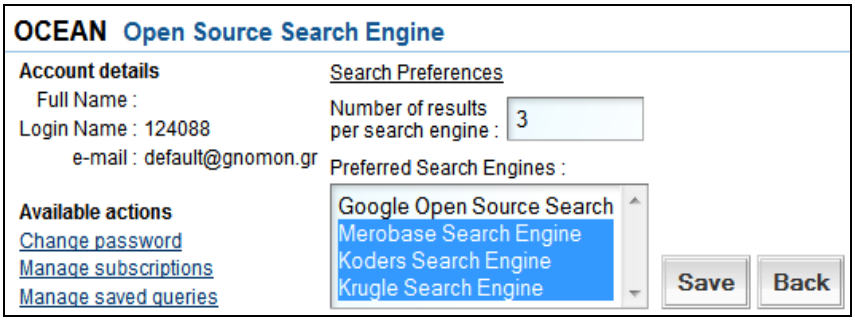


Figure 26: OCEAN's Preferences Screen

In the query of Figure 25, the system returned six results, three of which were retrieved by Merobase and three other from Koders. For this specific query, Krugle did not return any results although it was engaged. The aforementioned summary of results is reported at the bottom of Figure 25. The detailed results (Figure 27) include the following information: the source search engine (Search Entry), the title of the source file returned and the URL of the repository to which it is hosted (Result Entry), lines of code (loc) for the source code file (Metrics), any possible metadata (Metadata) and finally the type of license under which the source code file was originally published. For search engines providing more detail, OCEAN can also provide more detail since we do extract all the data available. Currently, OCEAN does not perform any global ranking on the results. They are displayed in the order returned by their search engines.

Save	AA	Search Entry	Result Entry	Metrics	Metadata	License
☐	1	Merobase Search Engine	**Title:** Login **Link:** cvs://cvs.sourceforge.net /j2eeindustry/w...	[-] Minimize loc 19	[-] Minimize	GNU Genera License
☐	2	Merobase Search Engine	**Title:** Login **Link:** cvs://cvs.sourceforge.net /j2eeindustry/w...	[-] Minimize loc 19	[-] Minimize	GNU Genera License
☐	3	Merobase Search Engine	**Title:** Login **Link:** cvs://cvs.sourceforge.net/niftyec /ec/tag...	[-] Minimize loc 23	[-] Minimize	no license
☐	4	Koders Search Engine	**Title:** FtpClient.java **Link:** http://www.koders.com/java/fi...	[-] Minimize loc 379	[-] Minimize	GPL

Figure 27: Detailed search results

Table 9 presents some performance results of a few, informal, comparative runs. The aim was to get an idea of the overhead introduced by the Query Engine of OCEAN. For the rest of the system, the overall performance depends on the performance of the slowest code search engine used (they used in parallel). OCEAN's performance in terms of speed and response time is quite satisfactory, at least to a large extent. The slowest search service is the one that queries Krugle. This is due to the fact that Selenium automates a Firefox instance and in its current implementation needs to get launched each time a query is posed to OCEAN. This introduces significant delay.

As far as Merobase and Koders are concerned, the required time for a search to complete is relatively low. Especially for Merobase, because of the API-based integration, the time needed was a bit smaller than the time the native Merobase web interface requires.

keyword: calendar **pref/nces**: "all languages" "all licenses"	
KODERS: 3.9	OCEAN w/ KODERS (10results): 9.6 OCEAN w/ KODERS (25 results): 10.0
KRUGLE: 7.0	OCEAN w/ KRUGLE (10 results): 32.6
MEROBASE: 7.8	OCEAN w/ MEROBASE (10 results): 5.7
OCEAN w/ KRUGLE+KODERS+MEROBASE (10 results): 36.6	

Table 9: Delays for Answer Recorded in Various Query Scenarios (All Times are in Seconds)

Finally, it should be noted that the Query Engine was run separately from the rest of the OCEAN. It was accessed through a large and constantly busy academic network that definitely contributes slightly in the delays recorded.

1.3.4 COPE

The Component Adaptation Environment (COPE) tool is used by reuse engineers of SME AGs to recognize, extract, test, document etc. components from OSS projects. The extracted components are then placed in the Component Repository and Search Engine (COMPARE) tool that is used by SMEs to discover the extracted components in the context of the application engineering process.

SME AGs experts, who are the operators of COPE, are called reuse engineers. After they have identified a potentially interesting OSS project for the application domain of their software development SMEs they create a reuse project for this OSS project using COPE. A “Reuse Project” combines the source code related information (of the original OSS project) with information resulted from the analysis process carried out by the reuse engineer. A Reuse Project’s lifecycle consists of four phases. First there is an Analysis phase in which the source code of the target OSS project is being analysed and the results of this analysis are being stored in the reuse project database. Then in the Component Recommendation phase the COPE tool automatically suggests class clusters that could serve as reusable components. The suggestions

can be based on different criteria. Following in the Component Making phase a set of functionalities allows the user to extract components from the reuse project by either using class clusters recommended in the Cluster Recommendation phase or by selecting a single class that along with its dependencies will form a reusable component. Finally in the Knowledge Management phase the user provides information for the generated components. Using the "Semantic Application" feature, the user can describe the functionality of each component. Moreover the reuse engineer can classify the resulting component to a specific domain and concept and finally upload the component to the COMPARE component repository.

The creation of a reuse project entails a preparatory phase in which the reuse engineer collects some project artefacts that are required by the COPE analysers and recommenders. These artefacts include: (a) The binary file of the compiled program which in the case of Java is a Java Archive (JAR file), (b) The libraries used by the project which are a collection of external JAR files that the project reuses, (c) The Version Control System URL of the project if available, and (d) The source code directory of the project which contains the source files. COPE reuses itself a number of Open Source components to perform its analysis. Some of these components require the binary JAR file.

After a reuse project has been created the first step is to perform static analysis. Static analysis is used to collect dependencies and metrics from the source code. COPE stores these facts in a relational database that relates information extracted by different types of analysis and related tools.

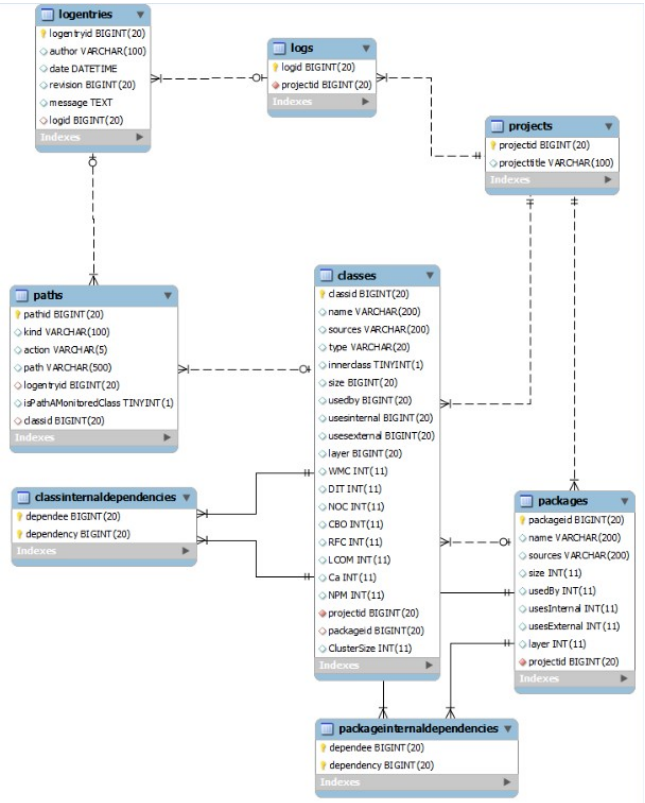


Figure 28: COPE Database Entity Relationship Diagram

Figure 28, depicts a part of the COPE database schema. Information originating from different source code analyzers is unified so that it is possible to recommend clusters of classes for componentization with algorithms that make use of the combined information. In Figure 28 we can see that there are projects which have a number of classes. Classes have dependencies with other classes and packages. Packages contain a number of classes. The dependencies are collected from the Classycle tool [32]. However for each class we also collect the Chidamber and Kemerer (CK) metrics [33] for Object-Oriented design complexity. The information for the CK

metrics is collected with the usage of the CKJM tool [34]. In COPE's DB schema this information is inserted as fields in the class table (e.g. WMC, DIT, NOC etc.).
The general approach for COPE component extraction is depicted in Figure 29. In Figure 29, we can see the different layers of COPE. At the first layer a number of analysers, analyse the OSS artifacts and insert the information in the database of OSS facts. At the second layer a number of recommenders access these facts and based on the facts recommend clusters of classes for component extraction. At the third layer these recommendations are used to create components from the selected recommendation. Although the process is tool-assisted it is not automatic. The reuse engineer decides which recommendation to accept and which component to extract. Furthermore after the component extraction has been performed the reuse engineer uses COPE to perform the testing and validation of the component and to create the test documentation for the testing and validation process. He or she also classifies the component under a domain and category and uploads it to the component repository where it becomes available to the reusers.

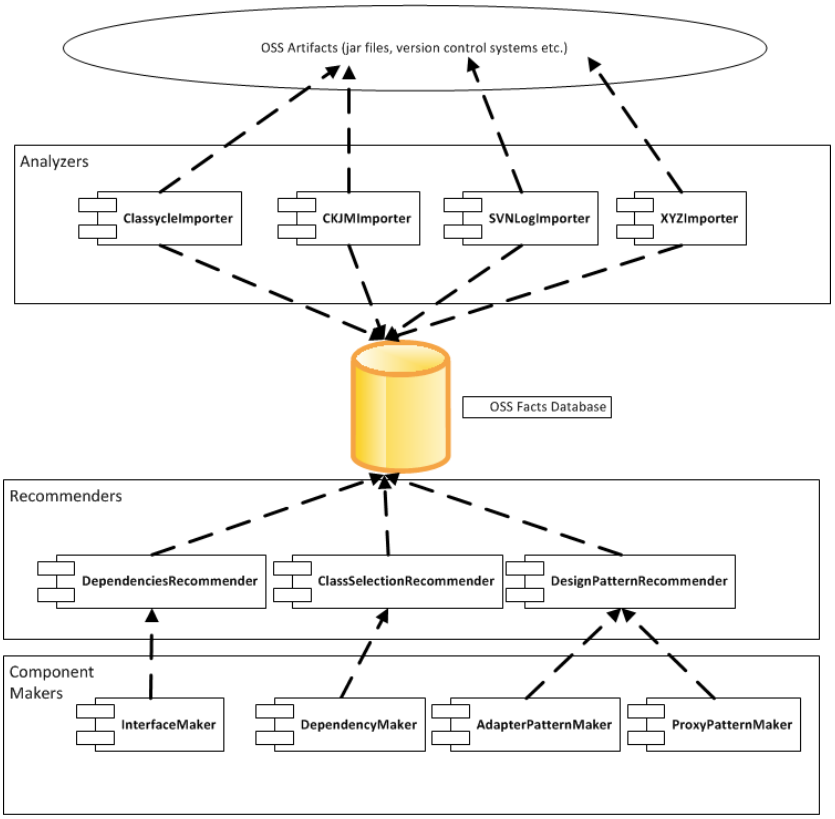


Figure 29: COPE Layers

1.3.4.1.1 Component Recommenders

Using the Cluster Recommendation options, the reuse engineer can easily come up with some recommendations of class clusters that could form possible components. For the time being COPE provides the following methods for recommending such class clusters:

- *Dependencies Recommender*: uses a genetic algorithm in order to form class clusters using the source code of the Reuse Project.

- *Pattern Recommender*: forms clusters based on design patterns detected in the source code of the Reuse Project. Patterns are detected using the approach and the tool described in [35]. Currently Adapter and Proxy design pattern instances are used as indications for recommendation of clusters. These two patterns were selected as more relevant for the purpose of

component identification. Other design patterns (e.g. Façade) may also be appropriate. The effectiveness of the different design patterns for component extraction is currently an active research area in our team.

- *Reusability Recommender*: Another very useful approach is to select a class and extract a component based on this class. The resulting component will have the interface of the public methods of the class and will include all the required classes for the reuse of this class. The reuse engineer can select this class based on the metrics that are presented in the main window, and especially the Cluster Size (i.e. the number of recursive dependencies of the class), the class Layer (i.e. how high or low is the class in the digraph of the project) and *R* (our own reusability index based on the Chidamber and Kemerer metrics suite for OO design complexity) metrics. Classes which are lower in the layered digraph of the project (have small layer value), have few dependencies (have small Cluster Size) and have larger R value (are more reusable) are good candidates for reusable components. The reuse engineer can extract components by right-clicking any class from the main window that seems promising based on the aforementioned metrics and extract a component for this class

All recommenders present a similar dialog to the reuse engineer who can examine the recommendations. In Figure 30 we can see this dialog.

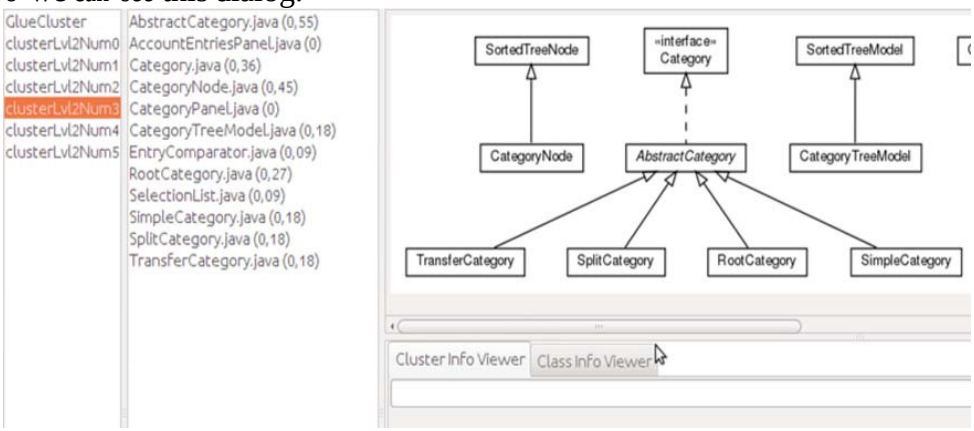


Figure 30: Cluster recommenders' dialog

The reuse engineer can select a class cluster (i.e. the recommendation) and examine the classes that are contained in it. In addition a class diagram is generated for visualization of the cluster. The reuse engineer can also examine information for the selected cluster and class including a tag cloud with terms encountered often for each class and cluster and a Latent Semantic Analysis (LSA) based index of the terms which are encountered in both cluster and class levels.
The visualization and the information are intended to provide the reuse engineer with a quick view of the cluster that is recommended and the high-level function of this cluster in a system.

1.3.4.1.2 Component Makers

Based on the analysis and recommendations carried out earlier the Reuse Engineer can now produce independent software components and then place these components in the repository using the 'Knowledge Manager' feature of COPE. Four different kinds of component makers are currently provided. The Interface Maker uses as input the clusters produced by the 'Dependencies Recommender'. The *Dependency Maker* presents all the classes of the project along with their reusability assessment and the reuse engineer can select a class and extract a component providing the functionality of the selected class. The *Adapter Pattern Maker* presents the clusters produced by the 'Pattern Recommender' and displays clusters involved in Adapter pattern instances. The *Proxy Pattern Maker* presents again the clusters produced by the 'Pattern Recommender' but this time it displays only clusters involved in Proxy pattern instances.

The user interface is again similar for the provided component makers as depicted in Figure 31.

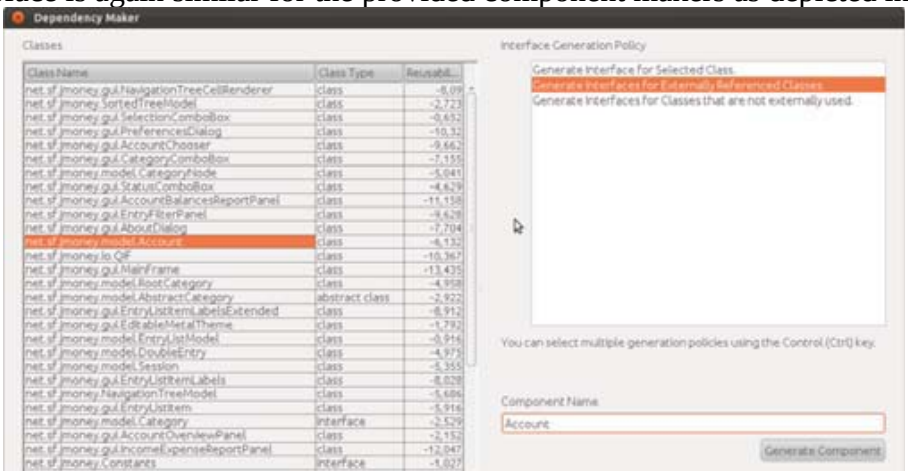


Figure 31: Dependency Maker Dialog

The reuse engineer can select a component as well as an interface generation policy (e.g. generation of an interface for the selected class, or generation of an interface for each externally referenced class) and provide a name for the component. The generated component contains all the required classes which are extracted from the project along with one or more generated interfaces for the component. Besides the original source code files and the generated interface or interfaces, the project libraries are also copied and an Ant build script is generated for the compilation of the component in an Integrated Development Environment (IDE).
Extracted components will be opened for further processing using an IDE (e.g. Eclipse or NetBeans). The reuse engineer will use the IDE to comprehend the component, create test cases for it or execution scenarios and discover further dependencies that are required which are not recoverable through static analysis alone (e.g. data dependencies). The component can then be tested dynamically using the test cases or execution scenarios that were developed by the reuse engineer as we explain in the following Subsection.

1.3.4.1.3 Component Testing & Validation

After the component source files have been extracted the reuse engineer will process the component further in an IDE. This is an essential program comprehension step in which unit tests or execution scenarios examining a specific functionality are created. Also it is important to resolve additional dependencies, such as data dependencies, that are required for the component to work.
After the reuse engineer has created some test cases for the component using the IDE and has resolved any additional dependencies which are necessary for the component to work independently, returning to COPE the feature of Dynamic Analysis will enable the reuse engineer to do the following:

1. Compute different types of test coverage based on the tests that were created. The types of coverage include Statement Coverage of the Component, Statement Coverage per Method of the Component, Linear Code Sequence and Jump (LCSAJ) coverage of the Component, and LCSAJ Coverage per Method of the Component.

2. Produce a Control Flow Graph per method of the Component which depicts the paths followed during the method execution of the test cases. CFGs are generated statically parsing the source code of the component. Aspect-Oriented instrumentation is then used to instrument the byte code and generate the trace of the execution. The instrumentation is necessary for tracing the execution path through the CFG and for calculating the LCSAJ and Statement coverage.

3. Perform validation which is a Model-Based Testing (MBT) [36] approach in which a large number of unit tests are generated automatically, utilizing method invariants provided by the Daikon invariant detector [37] and the component is then tested against the generated tests.

4. Produce the test HTML report which is a number of HTML pages, similar to JavaDoc, that package all the aforementioned information to an easily accessible format. The test HTML report will be included in the component package when it is uploaded in the component repository

1.3.4.1.4 Component Packaging & Classification

The component package that is generated from the usage of COPE is depicted in Figure 32. It includes the following: (a) A top directory with the component name, (b) A readme.txt file which contains information such as: A short description of the component, the originating OSS Project, license or licenses, the programming language and technology, other components it uses if any, and the domain and main concept of the domain the component provides, (c) Component source files, (d) Required Libraries, (e) Component Documentation generated by UML commercial or open source tools, and (f) The test HTML report which includes separate subdirectories for each test case along with the test results (coverage etc.).

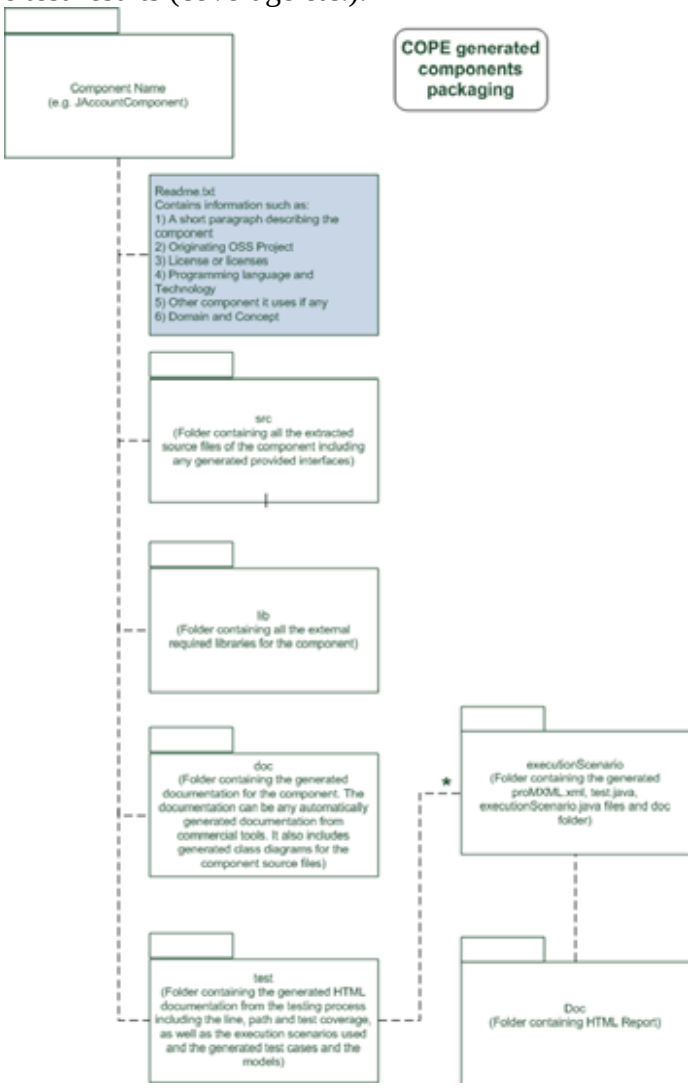


Figure 32: Component Package Directory Structure

The component package is then compressed to a file that is then classified using the Knowledge Manager feature of COPE and is uploaded in the Component Repository.
The Knowledge Manager (Figure 33) allows the reuse engineer to provide metadata for the component.

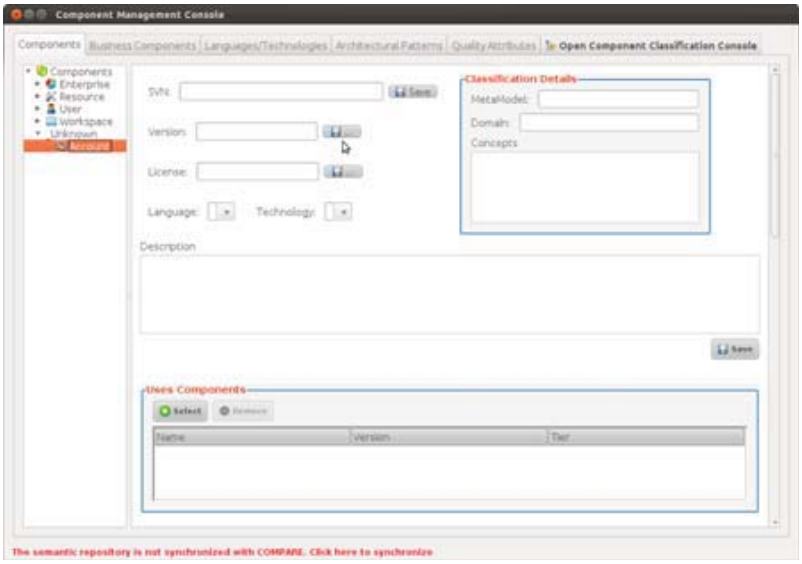


Figure 33: Knowledge Manager UI

The metadata for the component includes the following:

- The tier of the component. This is a characterization of the component's intended layer in the system. The component can be an Enterprise level component which encapsulates domain-specific functionality, a Resource level component which provides a generic service (e.g. database storage), a Workspace component which can, for example, coordinate different Enterprise level components in a workflow, or a User Interface component.
- The URL of the component package from which the reusers can download the component.
- The version of the component
- The programming language (e.g. Java) of the component and the technology (e.g. Java Enterprise Edition)
- The other components that the component uses, and
- The Domain metamodel under which the component was classified and the domain and concept that the component implements from this metamodel.

In addition the reuse engineer can use an 'Open Component Classification Console' to define domain metamodels for domains and concepts of these domains that are used when providing the aforementioned component metadata. Finally the reuse engineer can upload the component after this classification to the component repository (COMPARE) which makes it available to the reusers.

1.3.5 COMPARE

1.3.5.1 Introduction

COMPARE (Component Repository and Search Engine) is a tool that allows SME software re-users to search and discover the assets (software artefacts, technical documents, test suites, metamodels) produced by the Domain Engineering Process. COMPARE features an advanced search engine that can be used for searching among the components, according to specific needs and selection criteria ranging from desired features (functionality) to programming languages, execution frameworks, etc. Also, COMPARE supports the effective communication of structured information flows between the software re-users (asset consumers) and the reuse engineers (asset producers). These information flows allow re-users to place orders/requests and provide their feedback (e.g. bug reports) to the re-use engineers.

1.3.5.2 Technology platform

The COMPARE application is developed using the open source Apache-MySQL-PHP software stack. Also, COMPARE reuses extensively existing open source frameworks and web applications that provide various types of functionality. These tools are depicted in Table 10.

Tool	Functionality	URL	License
Joomla Framework	Software development framework for the development of web applications.	http://www.joomla.org/	GNU GPLv2
MediaWiki	Open source wiki package written in PHP, originally for use on Wikipedia.	http://www.mediawiki.org	GNU GPLv2
Apache Subversion	Software versioning and revision control system.	http://subversion.apache.org/	Apache License
WebSVN	Online subversion repository browser	http://www.websvn.info/	GNU GPLv2

Table 10: Tools comprising the COMPARE platform

1.3.5.3 Architecture Overview

The key objective of the application described in this document is to allow users to search and discover Software Components, allow users to upload new Software Components and provide community features regarding them. From the users point of view the application is a series of dynamic web pages, accessible through a web browser. On the backend, the application searches and retrieves data from a database and from other external sources through its external interfaces.

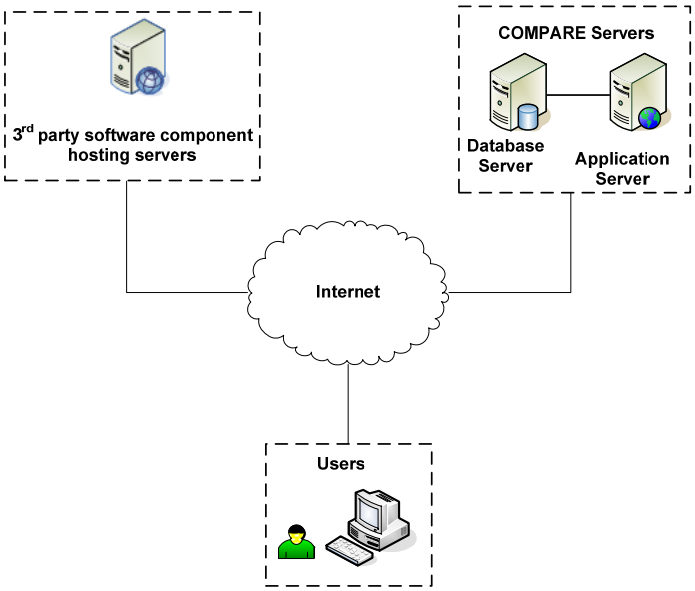


Figure 34: COMPARE network architecture

COMPARE is built on top of the Joomla framework and its architecture is heavily based on it. The Joomla architecture is decomposed in three tiers. The framework tier which includes the framework and the core plugins, the application tier which provides factory classes for application specific objects along with supporting APIs, and the extension tier which extends the functionality of the framework. Each Joomla extension that was created follows the MVC pattern depicted in Figure 35.

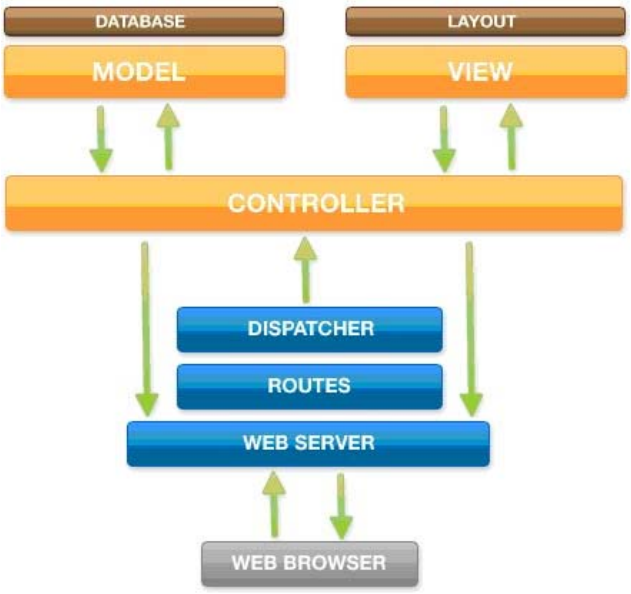


Figure 35: The Model View Controller pattern

In the MVC pattern, the model manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state. The view renders the model into a web page with which the user can interact with and the controller receives user input and initiates a response by making calls on the model. Finally, the controller accepts input from the user and instructs the model and view to perform actions based on that input.

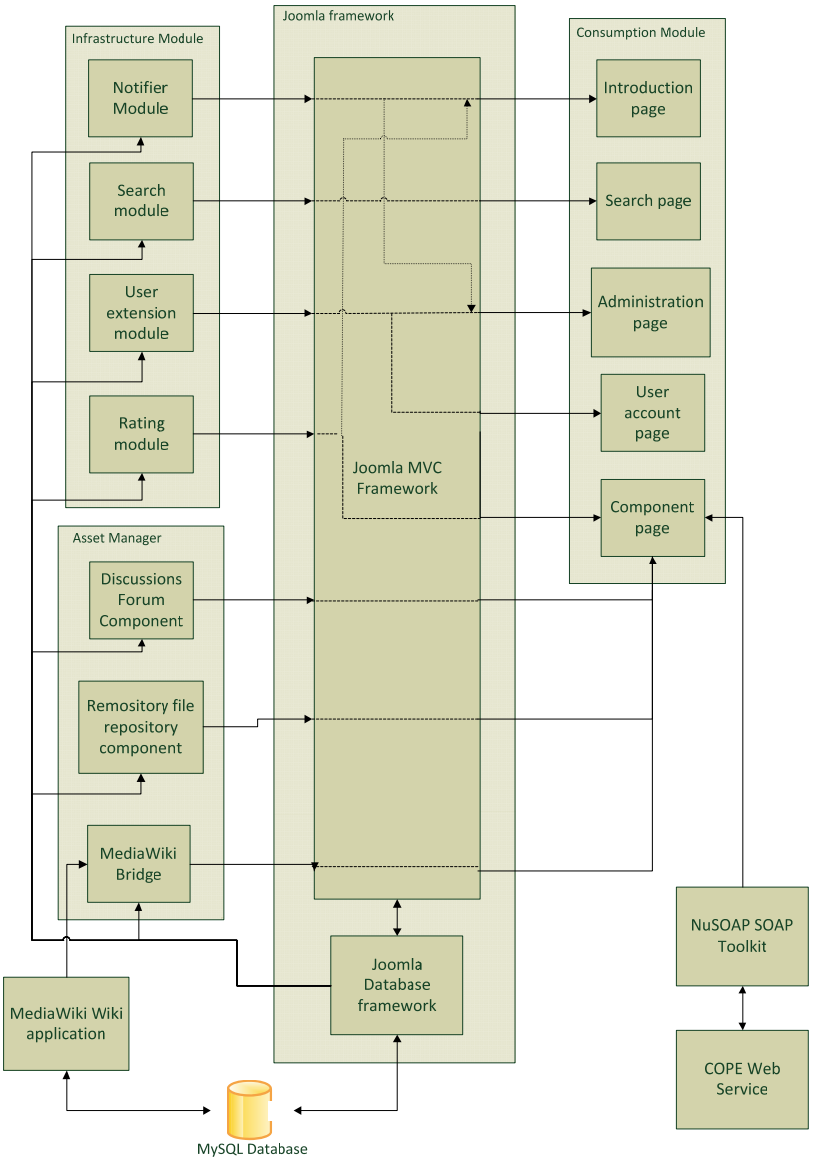


Figure 36: COMPARE Architecture

1.3.5.4 The Infrastructure Module

The Infrastructure Module provides a set of infrastructure services which are utilised by all other components of the COMPARE system. Specifically, it comprises the Asset Metadata Repository, the Asset Manager and the Notifier. Also, the Infrastructure Module provides access to statistical information about the platform and provides the template user interface on top of which the user interfaces of all the other components are rendered. Finally, the Infrastructure Module controls user access and permissions to the platform through the use of the User module Module.

1.3.5.5 Asset Metadata Repository

The Asset Metadata Repository is a relational database which COMPARE uses to store information about components. It is a MySQL database using the InnoDB engine and contains the tables of the Joomla framework, the tables of the third party integrated applications and the tables of the COMPARE extensions

1.3.5.6 Asset Manager

The Asset Manager provides access to the assets of each component. It is composed by the Component Page, the SVN access component and the social modules (Forum and Wiki components).

1.3.5.7 Notifier module

The Notifier Module is used to receive and update the recent activity of the hosted components. Also, this module generates an "activity index" based on the number of updates that were made in the last month.
A component update is considered to be made in the following actions:

- A change is made to a property of the component
- A new file is uploaded in the component's repository
- A change is made in the component's wiki page
- A new thread is started in the forum

1.3.5.8 User Module

The User Module handles all the functionality regarding the users of the platform. It is composed by the User Extension Module which is an extension to the User Component of the Joomla framework and the Component Rating Module which holds the rating information that the users apply to each component.

1.3.5.9 Consumption Module

The aim of the Consumption Module is to allow the software re-user to search, provide feedback and retrieve the software components that are hosted by the platform. The Consumption Module provides its features through a set of web pages which can be accessed via the World Wide Web (WWW). The Consumption Module comprises the Asset Searching Module, the Asset Retrieval Module and the Interest Management Module.

1.3.5.10 The Asset Searching Module

The Asset Searching Module provides methods for a software re-user to search and filter the software components hosted by the COMPARE. This module is accessible to the re-user via a web page, where the user submits his search terms, and the module uses them to search the Asset Metadata Repository and present the results. Also, the Asset Searching Module provides filtering mechanisms to filter the search results based on various criteria. The Asset Searching Module receives the search terms from the re-user and analyses them. Then, it composes search queries which are submitted to the Asset Metadata Repository. Afterwards, the Asset Searching Module receives the search results from the Asset Metadata Repository and applies a numerical weighting on each of them based on factors such as search term relevance and position. Finally, the Asset Searching Module communicates with the Feedback Management Module to receive information about the usage of each software component by the re-users. The Asset Searching Module is composed by the Search Module and the Search Page components which are described in the following sections.

1.3.5.11 Component Search Module

The Component Search Module is used to search for software components, from the Search page, asynchronously with the use of AJAX. Also, while the Search page is generating, it will use the model of this module.

1.3.5.11.1 Search page

The search page presents a search field which the user can use to search for CHSCs. Keywords entered in the search field will first be used to search a component that contains them in its name, then in its description and finally in its platform. For example, the keywords “COMPARE tool” will produce a search for a component which contains “COMPARE” and “tool” and then “COMPARE” or “tool” for the name, description and platform fields.

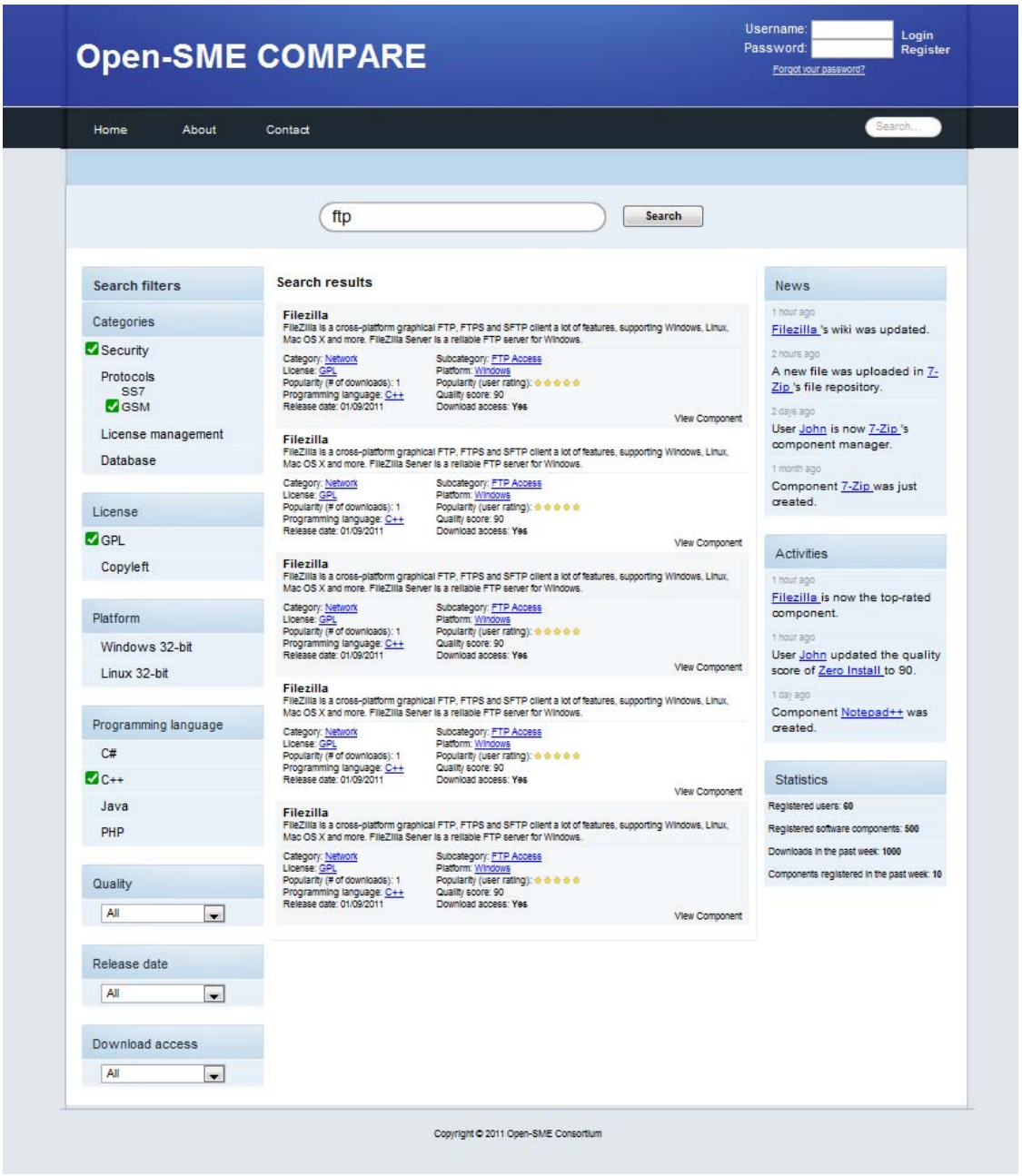


Figure 37: Search page

1.3.6 References

1. Hafedh Mili, Ali Mili, Sherif Yacoub and Edward Addy: “Reuse-Based Software Engineering: Techniques, Organization and Controls”, Wiley, 2002
2. J. Long, “Software Reuse Antipatterns,” SIGSOFT Softw. Eng. Notes, vol. 26, no. 4, pp. 68-76, July 2001

3. Bosch, J.: “The challenges of broadening the scope of software product families”, Commun. ACM vol. 49, no. 12, pp. 41-44, December 2006
4. E. Almeida et. al.: “Component Reuse in Software Engineering”, RISE, 2007 (available on-line http://cruise.cesar.org.br/index.html)
5. P. Clements and L. Northrop: “Software Product Lines: Practices and Patterns”, Addison-Wesley, 2002
6. K.C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, and M. Huh, “FORM: A feature-oriented reuse method with domain-specific reference architectures”, Annals of Software Engineering, vol. 5, no. 1, pp. 143-168, Springer, 1998
7. K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, “Feature-Oriented Domain Analysis (FODA) Feasibility Study”, Technical Report, CMU/SEI-90-TR-21, November 1990
8. M.A. Simos, “Organization domain modelling (ODM): formalizing the core domain modelling life cycle,” SIGSOFT Softw. Eng. Notes, vol. 20, pp. 196-205, August 1995
9. M. Simos, D. Creps, C. Klinger, L. Levine, and D. Allemang: “Organization Domain Modeling (ODM) Guidebook, Version 2.0”, Informal Technical Report for STARS, STARS-VC-A025/001/00, 14 June 1996
10. I. Jacobson, M. Griss and P. Johnsson: “Software Reuse: Architecture, Process and Organization for Business Success”, ACM Press, 1997
11. M. Griss, J. Favaro, and M. d'Alessandro, “Integrating feature modelling with the RSEB”, Proceedings of the Fifth International Conference on Software Reuse, pp. 76-85, IEEE, 1998
12. J. Bayer, O. Flege, P. Knauber, R. Laqua, D. Muthig, K. Schmid, T. Widen, and J. DeBaud, “PuLSE: a methodology to develop software product lines”, Proceedings of the 1999 symposium on Software reusability, Los Angeles, California, United States: ACM, 1999, pp. 122-131.
13. C. Atkinson, J. Bayer, C. Bunse, E. Kamsties, O. Laitenberger, R. Laqua, D. Muthig, B. Paech, J. Wust, and J. Zettel, Component-Based Product Line Engineering with UML, Addison-Wesley Professional, 2001.
14. C. Atkinson, J. Bayer, and D. Muthig, “Component-Based Product Line Development: The KobrA Approach”, 1st SOFTWARE PRODUCT LINE CONFERENCE, pp. 289-309, 2000
15. Ivica Crnkovic, M. Chaudron, and S. Larsson, “Component-Based Development Process and Component Lifecycle”, International Conference on Software Engineering Advances (ICSEA'06), p. 44, IEEE, 2006
16. T. R. Madanmohan and Rahul De’, "Open source reuse in commercial firms", IEEE Software, vol. 21, 2004, pp. 62-69.
17. G. Kakarontzas, P. Katsaros, I. Stamelos, "Component certification as a prerequisite for widespread OSS reuse", vol. 33, Electronic Communications of the EASST, 2010
18. J. M. Gonzalez-Barahona, A. Martínez, A. Polo, J. J. Hierro, M. Reyes, J. Soriano, and R. Fernández, "The Networked Forge: New Environments for Libre Software Development", in "Open Source Development, Communities and Quality", B. Russo, E. Damiani, S. Hissam, B. Lundell, and G. Succi (eds.), IFIP Advances in Information and Communication Technology, vol. 275, Boston Springer, 2008, pp. 299–306.
19. J. Howison and K. Crowston, "The perils and pitfalls of mining SourceForge", Proc. of the International Workshop on Mining Software Repositories (MSR 2004), 2004, pp. 7-11.
20. Krugle official website: http://www.krugle.com/
21. Koders official website: http://www.koders.com/
22. Merobase official website: http://www.merobase.com/
23. Free Softwarer Foundation official website: http://fsf.org/
24. Open Source Initiative official website: http://opensource.org/
25. OCEAN official website (beta): http://ocean.gnomon.com.gr/
26. OPEN-SME project official website: http://opensme.eu/
27. DEiXTo official website: http://deixto.com/
28. Selenium official website: http://seleniumhq.org/
29. Scherlen, (Balance Point column editor) J. Boyd, P. Pugh, M. Hampton, P. Morrison and F. Cervone, "The One-Box Challenge: Providing a Federated Search that Benefits the Research Process" Serials Review. 32 (4), 2006, pp. 247-254.
30. Xiaotian Chen, "Metalib, WebFeat, and Google: The strengths and weaknesses of federated search engines compared with Google", Online Information Review, 30 (4), 2006, pp.413–427.
31. Laender, B. Ribeiro-Neto, A.S. da Silva, and J. Teixeira, "A Brief Survey of Web Data Extraction Tools", ACM SIGMOD Record, 31 (2), 2002, pp. 84-93.
32. Franz-Josef Elmer: “Classycle: Analysing Tools for Java Class and Package Dependencies”, http://classycle.sourceforge.net, [Online; accessed 6-March-2012]
33. Shyam R. Chidamber and Chris F. Kemerer: “A Metrics Suite for Object Oriented Design”, IEEE Transactions on Software Engineering, vol. 20, no. 6, pp. 476–493, June 1994
34. Diomidis Spinellis: “Tool writing: A forgotten art?”, IEEE Software, vol. 22, no. 24, pp. 9–11, July/August 2005

35. Nikos Tsantalis et al.: “Design Pattern Detection Using Similarity Scoring”, IEEE Transactions on Software Engineering, vol. 32, no. 11, pp. 896–909, November, 2006
36. M. Utting and B. Legeard, “Practical Model-Based Testing: A Tools Approach”, Morgan Kaufmann, 2006
37. Michael D. Ernst et al.: “The Daikon system for dynamic detection of likely invariants”, Science of Computer Programming, vol. 69, no. 1–3, pp. 35-45, Dec. 2007
38. Hironori Washizaki and Yoshiaki Fukazawa: “A technique for automatic component extraction from object-oriented programs by refactoring”, Science of Computer Programming, vol. 56, no. 1–2, pp. 99-116, April 2005
39. Tom Mens and Tom Tourw´e: “A survey of software refactoring”, IEEE Transactions on Software Engineering, vol. 30, no. 2, pp. 126–139, February 2004
40. Simon Allier et al.: “Identifying components in object-oriented programs using dynamic analysis and clustering”, in proc. of the 2009 Conference of the Center for Advanced Studies on Collaborative Research (CASCON ’09), pp. 136-148, ACM,2009
41. Maurice M. Carey, Gerald C. Gannod: “Recovering Concepts from Source Code with Automated Concept Identification”, 15th IEEE International Conference on Program Comprehension (ICPC ’07), pp.27–36, 2007
42. Bas Cornelissen et al.: “A Systematic Survey of Program Comprehension through Dynamic Analysis”, IEEE Transactions in Software Engineering, vol. 35, no. 5, pp. 684–702, September 2009
43. P. Dugerdil and J. Repond, “Automatic generation of abstract views for legacy software comprehension”, in proc. of the 3rd India software engineering conference (ISEC ’10), p. 23, ACM, 2010

1.4 Potential impact, dissemination and exploitation of results

1.4.1 Potential Impact

Based on the capacities of the OPEN-SME repository and tools, a number of (bundles of) products and services can be offered to each customer segment (see Figure 38).

Figure 38: Elements contributing to value (Osterwalder & Pigneur)

The OPEN-SME tools & repository allow analysis services and quality assurance. If services that are exclusively based on the tools are considered, OPEN-SME can offer help to solve legacy issues. The repository only allows offering components. Finally, the tools themselves can potentially be sold.

Next to the direct outcomes of the OPEN-SME project a number of services can be offered to the target groups, such as training and knowledge (initially by AUTH), support and consultancy (also initially by AUTH), domain engineering services, and brokerage.

A third group of offerings relates to building up a stack of expertise, as OPEN-SME allows generating experts in OSS reuse and reusable OSS components, in OSS (components) integration, expert users, and domain engineering experts. Though there are a number of OSS reuse tools available, the unique selling point of the OPEN-SME approach is the combination of highly integrated reuse analysis tools on the one hand and the provision of a repository that allows direct access to reusable components with a so far unknown level of granularity. In this sense, newness and highly improved functionality are two core value propositions of OPEN-SME.

Another value proposition is performance, as the RODE process that is implied in the OPEN-SME approach towards OSS reusability allows improved process performance (systematic and efficient identification and testing of OSS code for reusability that goes far beyond what is possible today). Another feature resulting in improved performance is ease of identification of reusable software and its classification (categories). In addition, the metrics applied or generated in the OPEN-SME approach will improve the identification and selection of best practices. Finally, the establishing of a code-reuse-oriented community will allow to externalize a number of tasks from companies / the OPEN-SME partners to other members of the community, which

could particularly accelerate the growth of the number of components in the OPEN-SME repository. As a result, customized products (components, test results) will be available earlier than this is possible today, and components can be used systematically in OSS development, which is expected to significantly reduce the development time of new OSS products and services. The latter point leads to a third value that can be offered to clients, which is customization. This is achieved through tailoring the RODE process to domain-specific and company-specific needs, which may include the modification of tools.

A fourth value to be offered through the OPEN-SME business model is the capacity to help companies that so far are not able to perform effective code reuse analyses to get this job done. Overall, the partners intend to establish the OPEN-SME repository and tools as a brand. Given their newness and uniqueness, their qualification for a branding strategy is unquestionable. However, a comprehensive branding strategy depends on all partners' needs and capacities and has to be clarified and developed in a mid-term perspective (1-1.5 years). Challenges that have to be mastered in this regard are the name, which should reflect the core functionalities of the OPEN-SME repository and tools, a slogan, and a logo. "OPEN-SME" might not be appropriate, in this regard. However, other relevant cornerstones of a branding strategy have been identified: application fields / markets and the unique selling points are clarifies, as laid out above. The branding strategy might benefit from applying Kano`s model of customer satisfaction (see D26b) distinguishing 'attractive quality' from 'one-dimensional quality', 'must-be quality', 'indifferent quality' and 'reverse quality'.

The sixth value provided by OPEN-SME is design, as the implied focus on components eases and improves good software design. Seventh, price is an important value to be offered by OPEN-SME, since tools and repository are OSS, which implies that the costs related to these elements are comparably low. However, it should be noted that the efficient usage of the repository and the tools requires high level expertise, which might result in relatively high prices for OPEN-SME services.

The latter point is however countered by the eights value OPEN-SME can offer, which is cost reduction. The outcomes of the OPEN-SME code reuse analyses are a broad set of well analysed software and software components that are unlikely to produce in-house by most of the potential customers. This effect should outweigh expenses for high level expertise and overall result in lower production cost through

- larger supply with reusable code
- shorter development time
- ease of legacy management (for applications)
- less coding effort

However, these cost reductions might not be perceived by customers (due to unawareness of costs aligned with no or bad code reuse). In addition, cost reduction might be countered by high learning costs and possibly high transaction costs (when introducing the RODE process in business processes). The ninth value provided by OPEN-SME is risk reduction, as IPR issues become more transparent, extensive testing reduces the number of bugs in OS software and components, and the OSS reuse community and social network provides potentially a 24/7 service infrastructure. Especially SMEs will benefit from the latter. The tenth value provided by OPEN-SME is accessibility. OPEN-SME will ease the access to reusable software, components and test results through the Internet.

Finally, the eleventh value that will be offered with the OPEN-SME business model is convenience / usability, as the OPEN-SME tools and repository make it easier for firms and individuals to identify reusable code and components. Though the learning curve for handling the repository and tools effectively, it must be considered that so far OSS reusability analyses are

performed by a rather eclectic trial and error approach that very likely overlooks many reusable components and does not provide comprehensive insights in the reusability features of the code under scrutiny. In this sense, the highly integrated tools and the OPEN-SME repository will turn out relevant information on reusable code in a faster and more comprehensive way in shorter time than the code analysis practices especially SMEs are used to so far.

These offerings help to solve a number of typical problems potential customers have when OSS code reuse is considered. In the first place, the OPEN-SME approach helps to structure the process of code reuse. In addition, OPEN-SME provides additional documentation of code that is not available otherwise. Furthermore, OPEN-SME provides customers with knowledge of software architecture that is lacking at the customer's side. The OPEN-SME tools and repository also help to increase scalability and to enter new markets. Another problem that can be solved by OPEN-SME is ease of training new employees and of knowledge transfer. Overall, OPEN-SME helps companies to focus on their core tasks while OSS code reusability analysis can be effectively outsourced. SMEs benefit from OPEN-SME in particular through help in solving problems related to

- using and maintaining OSS efficiently
- time to market
- accessibility to code, high quality software, information about reusability of code
- tools
- skills
- knowledge

Individual developers will benefit through improvements of their

- status
- knowledge
- reputation

Against this background, following customer needs have been identified that can be satisfied by the OPEN-SME business model:

- Improvements of existing products
- Ease generation of new products
- Quality improvements
- Process optimization
- Decrease time to market
- Accelerated response to customer needs / requests
- Ease of support and maintenance

1.4.2 Exploitation plans

1.4.2.1 Overview

As laid out in Deliverable D2.6a, the various actors in the OSS value network play different roles. In our case, there are two key actors in the value network of the OPEN-SME toolset: the technical academic partners of the OPEN-SME project provide the developers of a toolset for OSS reuse and reuse services, and the OPEN-SME-AGs in the consortium provide the distributors of the toolset and these services. The technical/academic partners, primarily AUTH and TELETEL, compile and analyse a set of existing tools for the identification and evaluation of reusable OSS code and OSS components. These existing tools are transferred into a suite that allows fast and

comprehensive reusability checks of OSS code and components, which is not offered by any single tool underlying the suite. This act provides the key value creation process within the OPEN-SME project. However, in a second step the suite has been adapted to the capacities and needs of the SME-AGs within the OPEN-SME consortium, which play the role of the key distributors of the OPEN-SME suite, as the RTD partners within the consortium do not dispose of the required distribution channels and distribution expertise. The end users – primarily the target groups of the OPEN-SME-AGs, usually other SMEs and start-ups – either receive the results of an OSS reusability analysis carried out with the OPEN-SME suite by another actor (an SME-AG, a technical academic partner like AUTH, another company) based on requirements specified by the end user, use the OPEN-SME suite themselves in order to evaluate OSS code or components they want to reuse, or offer reusability services based on the OPEN-SME suite provided to them by an SME-AG.

Given the diversity of the OPEN-SME-AGs in the OPEN-SME project consortium, they dispose of very different capacities to distribute the toolset / services and they pursue diverse strategies with this toolset. For instance, while VSP has a number of OSS-related start-ups in its portfolio and OSS plays a significant role in the Swedish / Scandinavian economy, other partners, like ETEK or the Serbian SME-AG ISS first have to raise awareness of OSS among their members as well as in their members' domestic and regional key markets (see D26a for details).

Further advancements and differentiation of the OPEN-SME value network is currently subject to ongoing discussions. In principle it is possible and preferable to establish additional distribution channels for the OPEN-SME toolset in order to accelerate and broaden the market diffusion. One possible way, in this regard, is to establish, for instance, AUTH, TELETEL, GNOMON or BITGEAR – as core developers of the toolset – as a vendor of OSS reuse services, which may require alternative distribution channels outside the OPEN-SME consortium. Additional distribution channels could, for instance, be provided by academic institutes in the field of computer sciences, by one or more OSS communities, by other SME-AGs, and by companies.

The composition of a value network around the OPEN-SME suite and the roles the various actors in such a network play are thus depending on the capacities, objectives and strategies of the SME-AGs. If a technical partner like AUTH, TELETEL, GNOMON or BITGEAR decides to operate the OPEN-SME suite in alternative value networks outside the OPEN-SME consortium in order to push the diffusion of the suite and to enhance the efficiency of OSS development in the European software industry, it is possible that a number of new value networks will be created, which again will differ by the requirements and capacities of the key distributor and the need of the end users served by the distributor. This also involves IPR and license issues (see next section for a discussion of these points).

Depending on the composition and objectives of the value networks that are formed around the OPEN-SME suite, business models must be created that meet the requirements of these value networks. For instance, depending on the capacities and context constraints of the SME-AGs in the OPEN-SME consortium, it must be decided whether the SME-AG sells the right to use the suite or sells services based on the OPEN-SME suite, or distributes the suite for free. Actors within the OPEN-SME value networks, especially the SME-AGs as key distributors of the OPEN-SME suite, can choose from various Open Source Strategies in order to deal with the underlying community. A detailed overview of these strategies is provided in D2.6, here we would like to limit the discussion to the fact that SME-AGs will find ways to collaborate with the underlying community or to circumvent constraints set by the community by either follow a road that is independent of the community (e.g. by forking a community) or that makes the community

dependent on one or more of the other actors of the value network (e.g. by taking over the community).

1.4.2.2 Proposed SME-AGs Business Strategy

Based on the analysis of the position and role of three SME-AGs that belong to the OPEN-SME project consortium in business ecosystems and OSS value networks, first recommendations of suitable OSS reuse business models for these (and similar) SME-AGs can be given. Overall, VSP shows a very commercial orientation and must be considered as integral and important part of the business ecosystem in Västerås, in which OSS development and reuse are widespread. **Conclusively, VSP plans to take over an active and commercial role in the distribution and implementation of the OSS reuse tools and services based on these tools by advancing itself into a software vendor for the OPEN-SME tools / suite.** Business models developed for this sort of SME-AG should put the SME-AG in the centre of the model and strive to generate sustainable revenues directly for the SME-AG.

In contrast to VSP, EMYPEE, ETEK and ISS are not part of their members' value network. However, their position in the business ecosystem of its members qualifies those organizations as distributors of the OPEN-SME suite. Further activities that imply playing a commercial role seem primarily to be limited by the governance structures and traditional tasks of the organizations and by the underdeveloped market for OSS in the two regions. Under such conditions, a suitable business model for a SME-AG should try to focus on commercial members of the SME-AG that are capable to play a leading role in the distribution, application and advancement of the OPEN-SME suite, while the SME-AG itself should rather serve as a non-commercial distribution and information platform. The latter may imply to advance the service offerings of the organization in the direction of training courses and networking activities. These activities could be organized in collaboration with member organizations. In fact, such activities take already place, but they are organized informally by the members. In the case of the OPEN-SME suite, institutionalized information events and training courses appear a more effective means to achieve an effective distribution and implementation of OSS reuse tools and services in the Greek and Cypriot economy (EMYPEE/ETEK case).

In the case of ISS the relative small size of the organization's portfolio creates a natural limit to the distribution and exploitation of the OPEN-SME suite. Therefore, the business model should focus on a commercial partner that is capable to utilize ISS' huge network of business contacts in order to create a broader use base and thus ground for sustainable revenues from OSS reuse tools and services.

1.4.2.3 OPEN-SME Business Model and Exploitation Strategy

Being aware of the fact that the market introduction of a complex product like the OPEN-SME repository and tool needs time and a strategy, the partners have agreed to start the "OPEN-SME business" at a rather small scope, with VSP as key player for familiarizing, testing and implementing the OPEN-SME repository and tools in the robotics domain of the Science Park. In this initial phase, training and consultancy shall be provided by AUTH. The roll-out, which provides the second phase, is intended to happen in different directions. The first one is collaboration with the SMEs and SME associations in the OPEN-SME consortium. To this end, VSP and the other OPEN-SME partners involved in the OPEN-SME business model will survey their members in order to find out to which degree and in which way OSS is used within their portfolios. Based on the survey results, good starting points for the roll-out of the OPEN-SME repository and tools can be identified. The second direction for the roll-out is provided by other Science Parks, as they have been identified as powerful multipliers with a perfectly matching

portfolio of companies and domains in which the OPEN-SME repository and tools can be applied.

1.4.2.4 Customer Segments

A number of relevant customers have been identified. In the initial phase, the most important customers will be the VSP members, specifically those ones in the field of robotics. This approach has been chosen in order to familiarize with the OPEN-SME repository and tools in a controllable area. The robotics domain of VSP is particularly useful for the introduction and testing of the OPEN-SME tools and repository because these members of VSP have a lot of knowledge of OSS, so that the learning curve is assumed to be less steep than in other domains. In the second phase, when VSP has accumulated enough knowledge about the OPEN-SME tools and repositories, other Science Parks and Incubators will be approached. The International Association of Science Parks (IASP) has currently 388 members with overall 128,000 member companies,[3] thus providing a perfect platform for disseminating and applying the outcomes of OPEN-SME. In a mid-term perspective SMEs (outside Science Parks) with a lack of reuse engineers (and maybe domain experts, too) shall find a possibility to directly receive OSS reuse services from the SME partners or other Science parks. Finally, in the long run, large companies shall find opportunities to receive large scale support (training, reuse service) for OSS reuse analyses.

The precondition for successful offerings to SMEs and large companies is an effective and well-maintained website and a self-sustained OSS reuse community, with expertise in a broad range of domains. The value that can be created within the OPEN-SME business model serves, in the initial phase, three clusters within the VSP portfolio: robotics, smart grid, OSS. After the initial phase, following other actors will benefit from the value created by OPEN-SME

- wider VSP network
- other Science Parks
- OPEN-SME consortium
- Public sector
- SME clusters
- Software producing companies (not only software houses)
- Consulting companies
- Platform providers
- Quality assurance service providers (OSS & proprietary software)
- Individual developers / “geeks”
- OSS projects
- Academia (universities, students)

Besides robotics, other relevant domains for the OPEN-SME repository and tools are CRM, e-commerce, and banking, i.e. the OPEN-SME stakeholders will have to establish contact points to these domains and market the OPEN-SME outcomes in these areas. For the geographical dimension of the roll-out strategy, the partners have decided to start on local scope, then develop markets on national and international scope. Multipliers, in this regard, are national contact points of the OPEN-SME partners and the International Association of Science Parks.

[3] http://www.iasp.ws/web/guest/facts-and-figures

1.4.2.5 Channels

There are three types of channels – distribution, communication and sales – that serve different purposes and play a role at different points in time. Figure 39 illustrates this for the OPEN-SME business model.

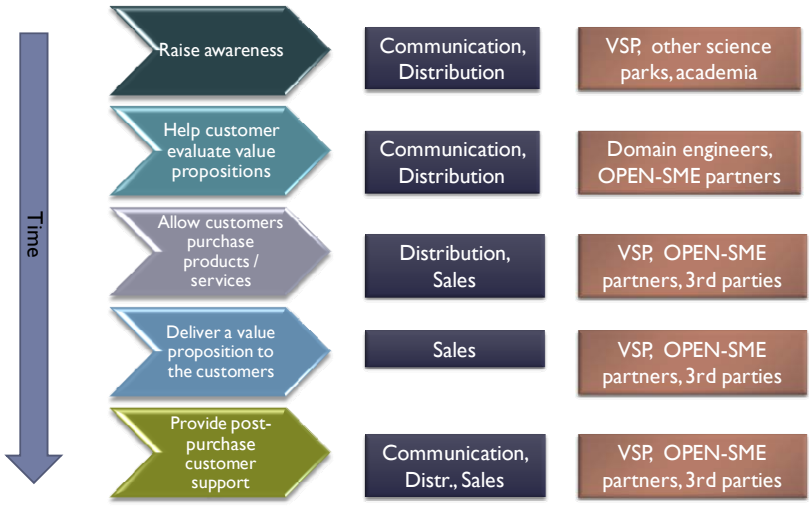


Figure 39: Purposes, phases and types of channels in the OPEN-SME business model

The OPEN-SME partners identified the following channels through which potential customers (target groups) presumably want to be reached.

- Internet (webpage, email)
  - Software communities
  - SME clusters / groups
  - Thematic forums
- Social media (Facebook, LinkedIn, Twitter etc.)
  - Registered "followers" from industry, academia and software communities
- Phone
  - Companies
  - Science Parks
  - EU networks
  - Industry Associations
- Face-to-face
  - VSP
- Teaching / courses
  - Academia
  - Industry associations / chambers of commerce
- Academia and industry collaboration
  - Master theses
  - Internships
- Events
  - Industry events
  - Software community events, e.g. FOSSDEM (fossdem.org)
  - Domain-specific events (e.g. conferences in the robotics area)

Since there is no similar service established within the partners of the SME consortium, it has to be evaluated which channels will be most effective. To this end, a number of measures have been discussed. During the test and pilots phase, events shall be broadcasted on the Internet. Challenges and opportunities shall be identified through benchmarking the success of different launches. Science Parks and SME clusters shall be attracted through direct contacts in existing networks. Showcases shall be created (prototypes, customer testimonials), and a download repository will be provided. Measures that shall be taken particularly in the pre-market phase are presentations at GeekMeets and evaluation of feedback received from there, conferences in relevant industry domains, and a "Beta-version workshop" with early adopter champions from various companies (through Science Parks and SME clusters, partners' networks).

In addition, EU networks and national and international events of / with other science parks and incubators shall be tapped. Finally, the partners decided to involve themselves in OSS associations and related events and in industry events, e.g. in the field of embedded systems (e.g. through ARTEMIS[4]). These channels are not considered as means that work only in one way. Overall, the partners are interested in feedback on which components are used, characteristics of components' life-cycle, members' roles and flexibility, and how to establish continuous contact to users / customers / developers through active involvement.

Regarding the integration of existing channels, the focus of the discussion was laid on the infrastructures at VSP, as these are most decisive for the start of the business model and for the later roll-out. There is an established and well-tested communication strategy for VSP members that can be reused and integrated in a wider OSS reuse communication strategy. This includes the usage of VSP's CRM system, though this requires categorization of member types.

Based on VSP's infrastructure and the capacities of the OPEN-SME partners, following channels have to be integrated (integration is led by VSP):

- Established personal relations to key companies
- Personal contact points for distributing OPEN-SME outcomes
- Email, mobile apps, webpage

The integration of the OPEN-SME channels with customer routines shall be achieved through the creation of the "big picture" of OSS reuse. Invitations to cooperate in order to create this big picture shall be distributed to the target groups. Furthermore, a SME component pool shall be generated. The latter requires as a precondition the establishment of a critical mass of SMEs involved / interested in OSS reuse

1.4.2.6 Customer Relationships

Figure 40 shows the different types of customer relationships that can be distinguished.

[4] http://www.artemis.eu/

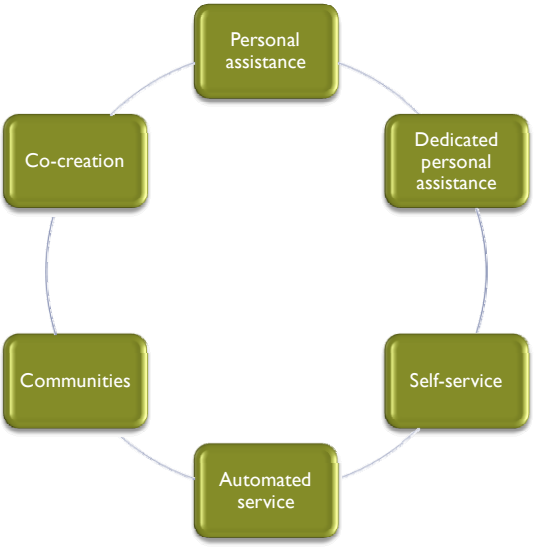


Figure 40: Types of customer relationships

The establishment of a self-sustained OSS reuse community is considered to be the key for all customer relations in the OPEN-SME business model. Regarding the types of relationships, the partners agreed that fully and semi-automated relationships should be avoided, as the complexity of the tasks probably does not allow for the level of standardization that would be necessary for these types of relationships. Within the community itself, self-service relationships may be an option, as the level of expertise within the community should be high enough. However, the default setting for customer relationships should be personal relationships, maybe with dedicated personal assistance as a special case in domains or for large companies or SME clusters. There are already a number of relationships established that can be used for the OPEN-SME business model: These relationships exist between

- VSP members
- other OPEN-SME SME AGs and their members
- OPEN-SME partners
- VSP members and OSS communities
- VSP and other Science Parks
- VSP / VSP members and industry associations
- VSP and government institutions
- VSP and academia
- OPEN-SME SME-AGs and industry associations
- OPEN-SME SME-AGs and government institutions
- OPEN-SME SME-AGs and academia

1.4.2.7 Key Activities

Key activities that must be performed in order to run the OPEN-SME business model successfully are twofold, on the one hand they have to help preparing the market for the OPEN-SME tools and repository and the services based thereof, on the other hand they have to secure and advance the value propositions offered to the target groups. One key activity that is important in the initial phase is a survey / overview of OSS activities within the portfolio of the SME-AGs and SMEs of the OPEN-SME consortium. This survey would provide an initial overview of the markets for the OPEN-SME tools, repository and services and contact points for entering these markets. Other activities related to market preparation are community building, the

provision of experts and expertise, problem solving capacities (directly or through portfolio members), sharing of investment costs, organizing events and training (initially by VSP, either in Västerås or in Stockholm), and the dissemination to other Science Parks and SME clusters, industry associations and the like. To the same end, key partners have to identify contact points in relevant domains, provide software components, testing, promotion (including academic and commercial publications, such as journal articles and whitepapers), and distribution. Activities related to securing and advancing the value propositions are updates of existing software, software extensions, integration of additional functionalities in existing software, and certification services for special high quality software and components.

1.4.2.8 Key Partnerships

There are different types of key partnerships that serve different purposes, as illustrated in Figure 41 .

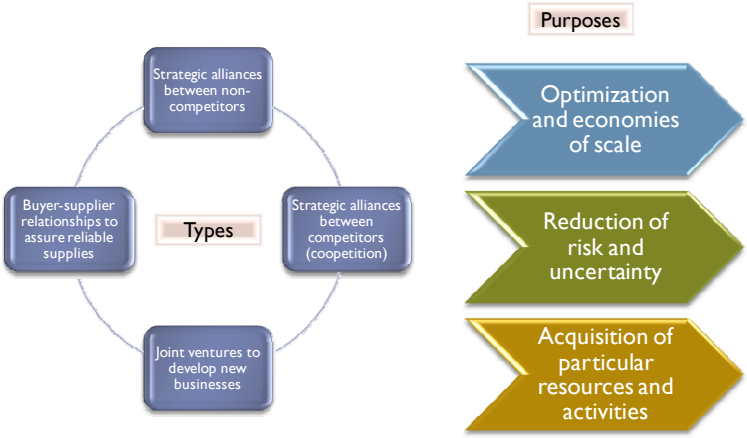


Figure 41: Types and purposes of key partnerships

The key partners in the OPEN-SME business model are, in the initial phase, the partners of the OPEN-SME consortium and the VSP member companies (especially in the field of robotics). These partnerships can at current be considered as informal (as not based on a contract) strategic alliances between non-competitors. At a later stage, when a critical level of OSS reuse expertise has been built up at VSP and OPEN-SME consortium partners, additional contact points in relevant domains (which have to be identified by the partners), in particular other Science Parks have to be integrated in the business model as key partners. In this case, other forms of partnerships may be chosen, and the relationships might get formal (i.e. based on contracts).

A special key partner is academia, as academia does not strive for commercial revenues but plays a vital role with regard to quality assurance, branding, publications and promotion of OPEN-SME. The key suppliers of the business model are, in the initial phase, the AUTH-team (reusability analysis, training), later the key suppliers will be part of a self-sustained community of SMEs, freelancers and volunteers, related to VSP members and other Science Parks, OPEN-SME partners and academia. The key resources to be required from partners are

- Software components
- Trust building / branding capacities and efforts
- Manpower / expertise
- Networks / contact points

1.4.2.9 Key Resources

There are a number of key resources required by the OPEN-SME value propositions. In the first place, there is an essential need for domain experts, first in the field of robotics, later in other

domains, too. In addition, hardware is needed for server and storage capacity. Cloud computing was considered to be an inexpensive and efficient and flexible option, in this regard. Other key resources are assistance in building the OSS reuse the community / network and clarifying IPR conditions (rights to OPEN-SME repository and tools).

In the introductory phase there is an "enabler" needed, i.e. initially one person in charge for introducing the OPEN-SME tools and repository at VSP. This person has most likely to be provided by AUTH. Finally, a clearly defined timeframe and network, in which the OPEN-SME tools and repository will be applied in the initial phase and later roll-out, is required. Key resources required by the customer relationships are

- Clarification of target groups
- Identification of domains
- Businesses and contact points
- Network
- Branding (through existing distribution channels)
- Grassrooting / community building (as part of marketing)
- Regularly updated webpage with relevant information
- Timely information with regard to components etc.

Key resources required by the distribution channels are

- Survey of VSP companies
- Marketing capacities
- Mapping of target markets
- Branding experts
- Networks
- Identification of relevant events (industry events, academic events, policy events etc.)
- Contacting and coordination with other Science Parks, surveying their OSS capacities and needs
- Strategy: what to do in which order
- Financial resources
- Early adopter champions

With regard to the market introduction of the OPEN-SME repository and tools, for which the identification and approach of early adopters is extremely important, Mohan [41] warns that a blog post or a launch at a startup event or a press article will not suffice to succeed. He suggests "a disciplined 3-step approach":

- Profiling and Identification (persona creation)
    - For B2B, 4 important characteristics to profiled and identify early adopters:
        - Location
        - Title of buyer (for the OPEN-SME business model, decision-makers for software development and software purchases are probably most relevant, but the survey should validate this)
        - Industry/domain (the survey has to identify the OSS-reuse-intensive domains)
        - Size of company (according to Mohan, mid-sized companies and a few

large companies tend adopt new innovations faster compared to smaller companies)

- For B2C , additional characteristics to consider are, inter alia, age, location, gender, monthly income among others.

- Interaction and Introduction - make an initial connect with early adopters through (one of) following three mechanisms:
  - "Engagement online: Following them and posting thoughtful (real human) comments (not spam or robot messages) on twitter or their blog.
  - Events: Instead of presenting at a booth when your startup is not ready, demo your mock-up or early version to them at events (as an attendee) to get feedback.
  - Introductions from other early adopters. Early adopters know each other well and tend to be connected to each other well. They are usually open to sharing new, innovative ideas with other early adopters."
- Nurturing and Engagement – get feedback from early adopters and offer them to influence the product direction with the goal to categorize early adopters into 3 types and focus on making your champions successful with your product :
  - "Champions: They like your product, think it solves a problem and are willing to provide feedback on what they would like, to make it better. Your goal should be to make these users the most happy with your service, be very responsive and introduce features they desire quickly. You can find them by looking at the # of times they return to use your service after the launch day.
  - Bandwagoners: They typically join since some other early adopter has joined who mentioned the product. They will come if the product is free, test it for an initial period, then will usually never show up until it is "more mainstream" or "many bugs have been worked out".
  - Naysayers: They have something negative to say about every new product, so while its best to ignore them, be thoughtful and respond to their feedback, but don't focus on them a lot. They will highlight many features that you currently don't have or plan to have. They are most likely to compare it to other solutions and in a negative light."

1.4.2.10 Revenue Streams

Revenue streams can be generated in various ways, as illustrated in Figure 42.

Figure 42: Ways to generate revenues

Given the interview results it is obvious that customers are not easily willing to pay for OSS reuse analysis and services. However, the workshops have identified a number of values that appear attractive enough to be paid for by the target groups. The first value in this regard is certification, as this service provides a sort of guarantee that the software or component does

what it is supposed to do. The idea of the OPEN-SME partners is to provide a medium-level certification that can be issued based on extensive testing but without going through the time consuming procedure of strictly formal certification, like by ISO standards. Another value that target groups are expected to pay for is tested components. Here, customers have to pay for the tests, not the components, as these are OSS.

Thirdly, extra documentation seems to be a value companies and freelancers would probably be more likely to pay for. Premium models with extra information, exceeding the information generally provided to everyone, could also provide a value customers are willing to pay for.

Other such values are:

- Security
- Established and trusted brand
- Test and quality assurance
- Basis for demand of services: reference implementations and reputation
- Tools (if partners decide to sell tools)
- In SME clusters: additional service that can be provided to members' customers

Regarding what services and products potential customers (here: VSP members) are currently paying, it turned out that this applies to hiring of internal programmers, consultancy (to a limited amount), commodity software, and available components (very rarely). As a general rule, if a product or service does not serve the core business the willingness to pay is rather low. However, when problems arise or cost savings become evident the willingness to pay increases. Regarding preferences of types of payment there was a strong agreement that one time payments have to be the default, as subscriptions and licenses are usually rejected by the potential customers.

1.4.3 Dissemination

1.4.3.1 Project Web Site

The OPEN-SME consortium established a website, (http://opensme.eu) for the support of the dissemination activities. This site provides public access to general information on the project (objectives, partners, scope, etc.), and to its public deliverables and presentations. Also the site accommodates restricted sections accessible only by the consortium members. The project web site is updated with information and content on a regular basis (Figure 43).

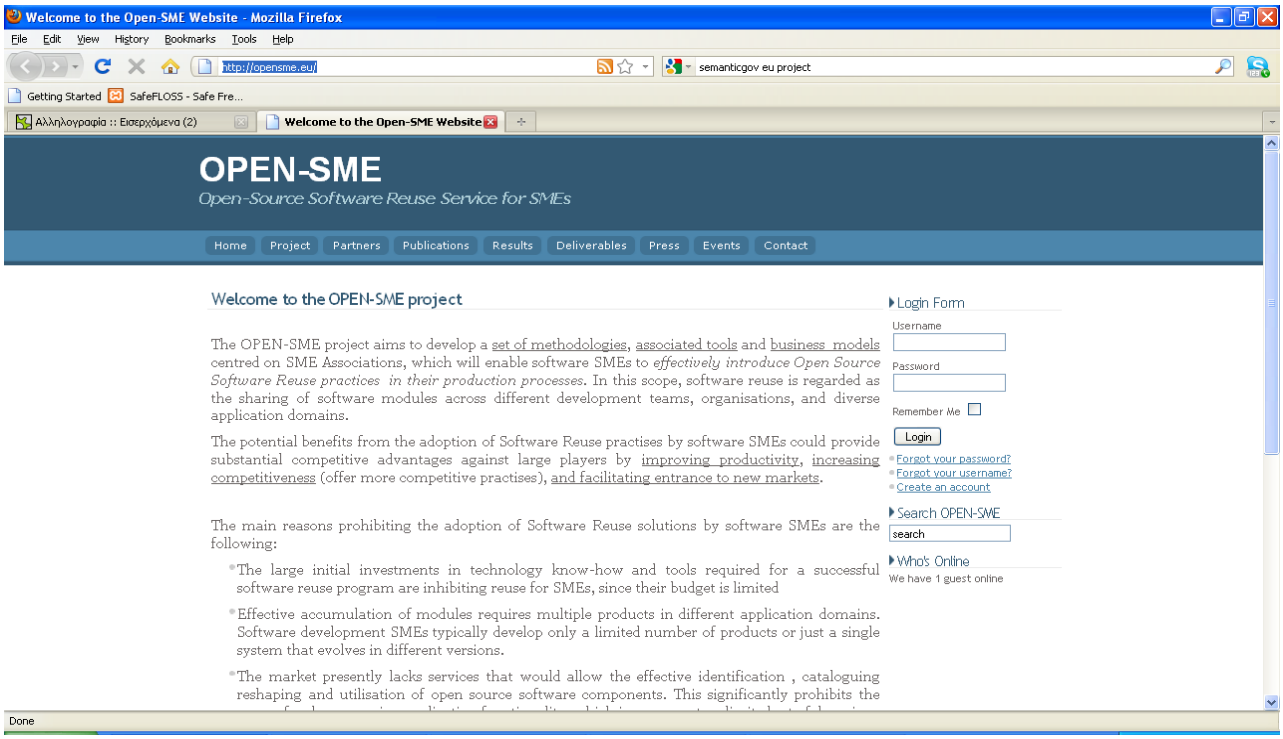


Figure 43 OPEN-SME project web site – Welcome page

The project website also includes the online training material prepared for the OPEN-SME processes (Figure 44, Figure 45).

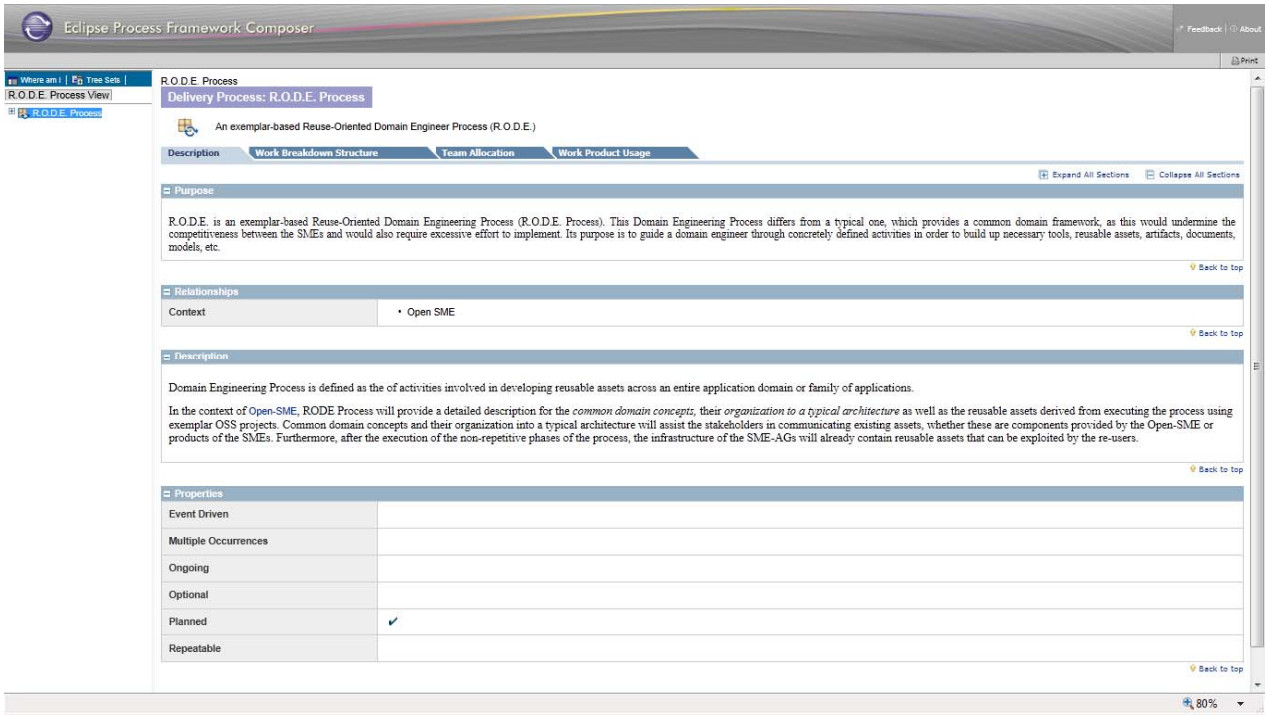


Figure 44: The RODE training page

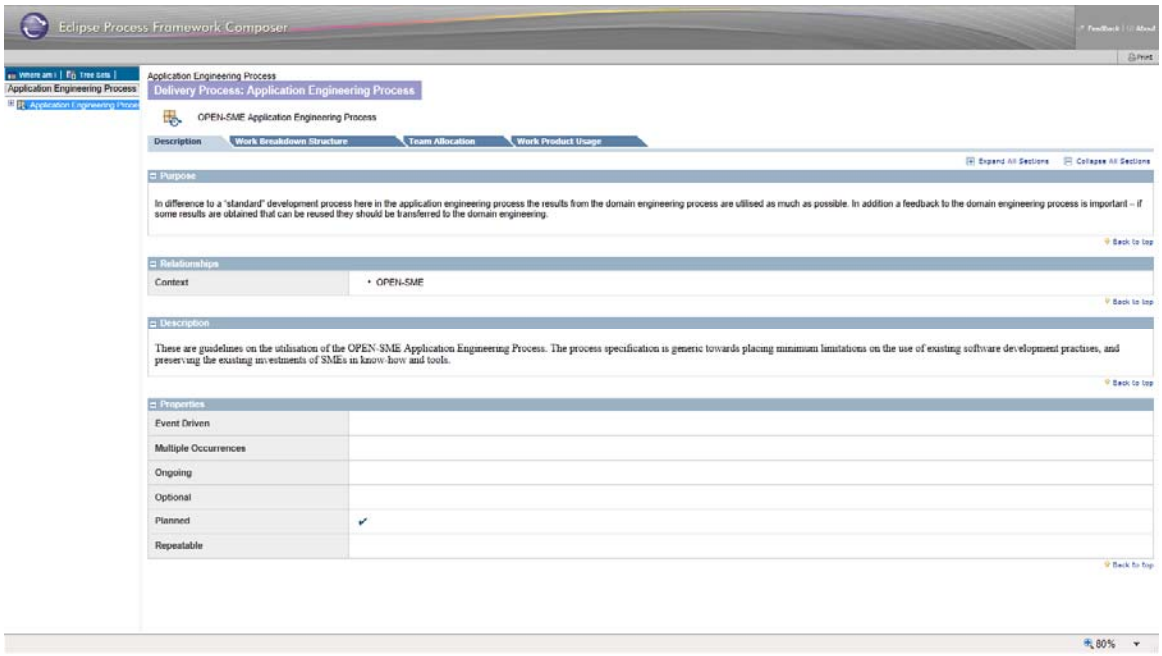


Figure 45: The AEP training page

To increase the visibility of our results and attract researchers and software engineering practitioners around the world, we initiated a Special Interest Group (SIG) mailing list. Those interested in OPEN-SME findings wishing to explore the deliverables in the respected category on the site, are automatically subscribed to the Open-SME SIG list. Currently there are 247 members subscribed.

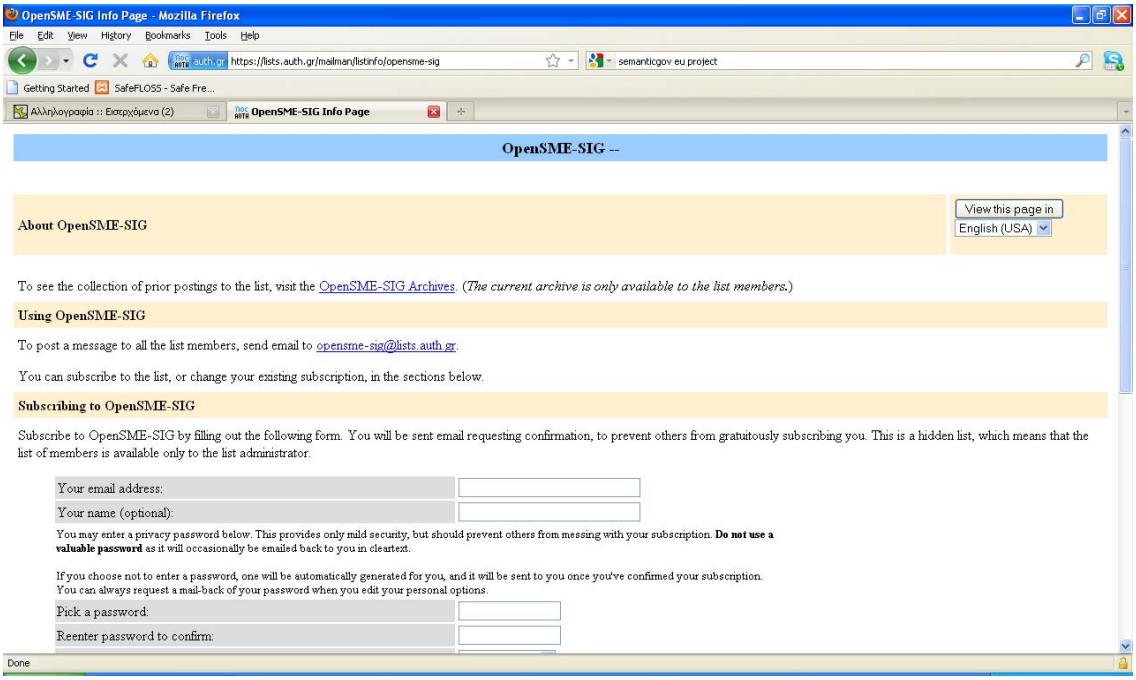


Figure 46 OPEN_SME Special Interest Group Subscription page

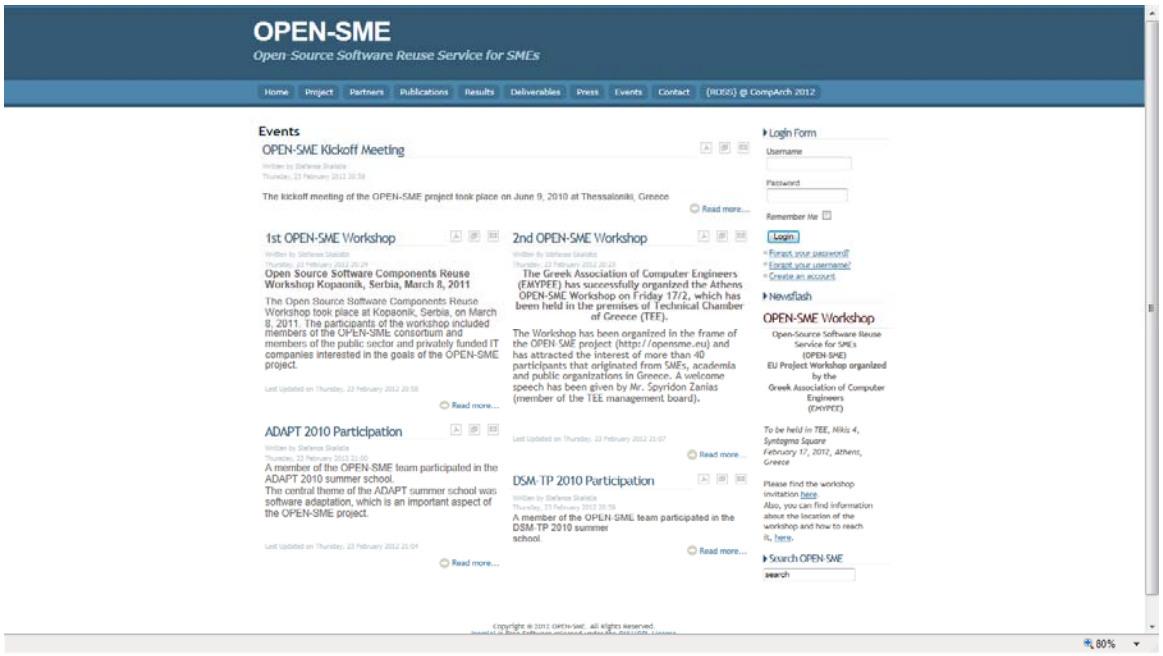


Figure 47 OPEN_SME Events Page

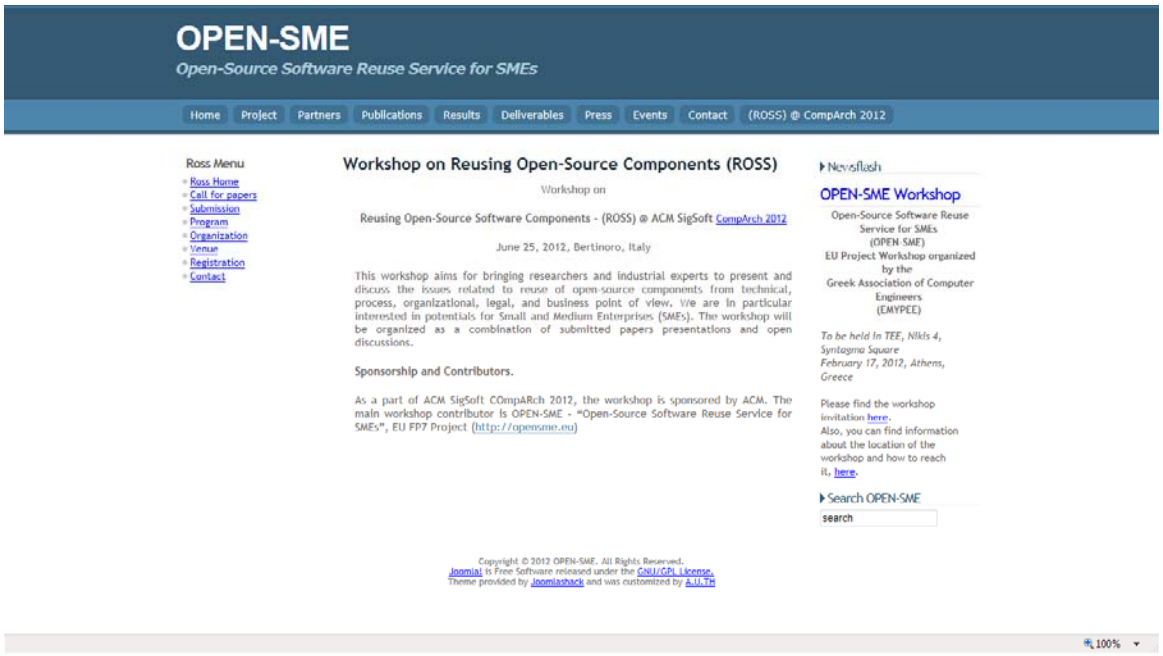


Figure 48 Final Workshop Page

1.4.3.2 Dissemination Events

During the project a large number of dissemination activities took place from the majority of the partners. Furthermore, all the kinds of dissemination activities have been covered by the partners.

- A member of the OPEN-SME team participated in the DSM-TP 2010 summer school. The main concept of the DSM-TP summer school was Domain Specific Model (DSM) and Domain Specific Languages (DSL) which are an important aspect of the OPEN-SME

project regarding the role of the re-use Engineer. Details on the topics of the school can be found at the DSM-TP 2010 summer school webpage: (http://ctp.di.fct.unl.pt/DSM-TP/).

- A member of the OPEN-SME team participated in the ADAPT 2010 summer school. The central theme of the ADAPT summer school was software adaptation, which is an important aspect of the OPEN-SME project. Details on the topics of the school can be found at the ADAPT 2010 summer school webpage (http://userpages.uni-koblenz.de/~adapt/summerschool2010/)

The ADAPT 2010 Participants

- Members of the consortium attended conferences and workshops of high importance in respect to OPEN-SME project:

1. QUATIC 2010 (7th International Conference on the Quality of Information and Communications Technology), Oporto, Portugal, 29 September to 2 October 2010.
2. ENASE 2010 (5th International Conference on Evaluation of Novel Approaches in Software Engineering), Athens, Greece, 24-25 July 2010.
3. Presentation entitled "Software Recycling", by Prof. I Stamelos, at the **University of Groningen**, NL on July 2nd. 2012.
4. Presentation entitled “OPEN-SME Project”, by Prof. M. Ivkovic at **International Conference ICIST 2012**, ISBN 978-86-85525-10-0, Pages 46-58, 29/2-3/3/2012. Kapaonik.

- **Open Source Software Components Reuse Workshop Kopaonik, Serbia, March 8, 2011**

The Open Source Software Components Reuse Workshop took place at Kopaonik, Serbia, on March 8, 2011. The participants of the workshop included members of the OPEN-SME consortium and members of the public sector and privately funded IT companies interested in the goals of the OPEN-SME project. The topics discussed during the workshop included an overview of the OPEN-SME project and its goals, primarily centered on Open Source Software components reuse from SMEs and the related business advantages. There were also presentations on the technical aspects of the OPEN-SME project and more specifically the software comprehension tools and approaches and the domain engineering process.

- **VSP Workshop**

The workshop took place at VSP, Vasteras, Sweden on 25 and 26 January, 2012, with the participation of AUTH. MDU and UM-MERIT, which delivered a whole day seminar regarding the OPEN-SME business models. The agenda of the workshop also included a presentation of the OPEN-SME OSS Reuse Platform and Repository, VSP's plans to make use of OPEN-SME, usage preconditions (skills and capacities) and roles / collaboration

The event in Vasteras Science Park (VSP) in which AUTH, VSP, UM-MERIT, MDU members participated highlighted the need for the robotics domain which resulted in component extraction from the ROSJava project.

- **Second OPEN-SME workshop**

The Greek Association of Computer Engineers (EMYPEE) has successfully organized the Athens OPEN-SME Workshop on Friday 17/2/12, which has been held in the premises of Technical Chamber of Greece (TEE). The Workshop has attracted the interest of more than 40 participants that originated from SMEs, academia and public organizations in Greece. A welcome speech has been given by Mr. Spyridon Zanias (member of the TEE management board). The Workshop program contained 10 presentations and a round table discussion. The presentations contained results and ongoing developments of the OPEN-SME project, guest speeches and the results of the “Open Source” Working Group (WG) that is introduced and tasked by EMYPEE aiming to analyze the opportunities for the Greek IT engineers with respect to usage of open source solutions, and to provide suggestions and best practices for exploiting open source projects in the SMEs, public organizations and educational institutes. The round table discussion was particularly live and attracted the interest of the participants.

Figure 49 Athens Workshop Photo

- **Third OPEN-SME workshop**

The 3rd OPEN-SME workshop took place on the 30th of May in Nicosia, organized by ETEK. The workshop was attended by 30 members of the IT Community of Cyprus and was addressed by the General Cashier of ETEK, Mr. Antonis Valanides. There was considerable interest from the participants in both the OPEN-SME toolset and the VSP business practices.

Figure 50 Nicosia Workshop Photo

- **Final OPEN-SME workshop**

The final workshop was sponsored by ACM and organised as part of the ACM SigSoft COmpARch 2012 conference (http://opensme.eu/ross), bringing researchers and industrial experts to present and discuss the issues related to reuse of open-source components from technical, process, organizational, legal, and business point of view. The focus was on the potential benefits for Small and Medium Enterprises (SMEs). The workshop was organized as a combination of submitted papers presentations and open discussions in Bertinoro, Italy on 26 June 2012. In the event we had the participation of large companies such as Siemens, ABB and Ericsson as well as the participation of important academic institutions (e.g. the developers of the Merobase search engine from the University of Mennheim).

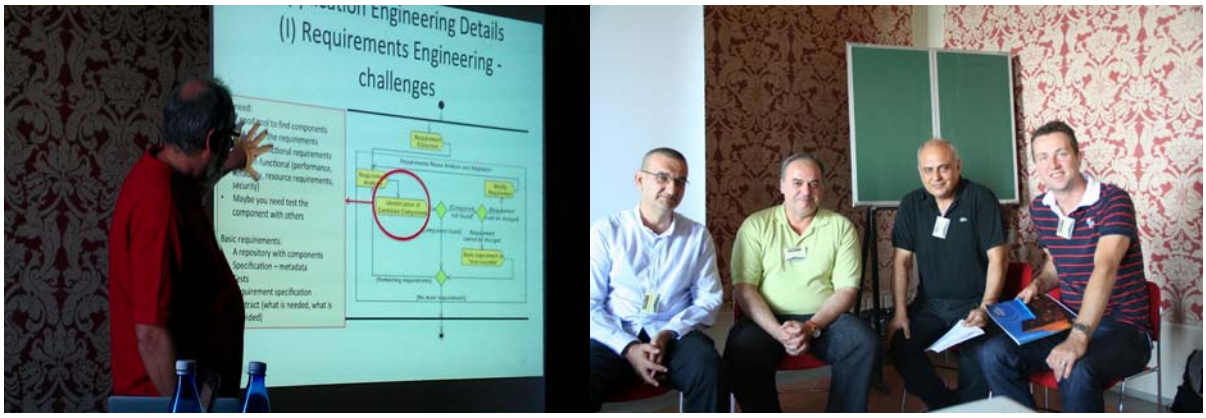


Figure 51 Photos from ROSS Workshop (COPE Presentation and Panel)

- **Preparation of 1st and 2nd OPEN-SME newsletter.**
- **OPEN-SME poster and presentation at the EMYPEE annual conference in the University of Patras, Greece on 18/12/2011**
- **Preparation of SIG questionnaire**
    - Questionnaire providing an overview of the project, identifying the main research areas, and requesting contact details and feedback on the interest in specific areas. The OPEN-SME partners distributed the SIG questionnaire to selected business and research partners.
- **Formulation of OPEN-SME SIG (more than 70 questionnaires were returned).**
- **Creation of OPEN-SME SIG mailing list and communication of information on the project results. More than 247 members**
- **Organisation of first EMYPEE SIG meeting in Athens on 11/3/2011**
    - 26 EMYPEE SIG members participated and were presented the rationale, the expected results and current progress of OPEN-SME

1.4.3.3 Publications

The consortium achieved the following publications:

Journals:

1. Apostolos Ampatzoglou, Apostolos Kritikos, George Kakarontzas, Ioannis Stamelos: "An empirical investigation on the reusability of design patterns and software packages", Journal of Systems and Software, Volume 84, Issue 12, December 2011, Pages 2265-2283, http://dx.doi.org/10.1016/j.jss.2011.06.047
2. George Kakarontzas, Eleni Constantinou, Apostolos Ampatzoglou and Ioannis Stamelos: "Layer Assessment of Object-Oriented Software: A Metric Facilitating White-Box Reuse", accepted for publication in the Journal of Systems and Software, Elsevier, 2012

3. George Kakarontzas, Panagiotis Katsaros and Ioannis Stamelos: "Component Certification as a Prerequisite for Widespread OSS Reuse", Electronic Communications of the EASST, Volume 33: Foundations and Techniques for Open Source Software Certification 2010, http://journal.ub.tu-berlin.de/eceasst/article/view/449/433/

Conferences:

1. George Kakarontzas, Vassilis C. Gerogiannis, Ioannis Stamelos, and Panagiotis Katsaros: “Elastic Component Characterization with Respect to Quality Properties: An Intuitionistic Fuzzy-Based Approach”, In Proceedings of the 15th Panhellenic Conference on Informatics (PCI '11), pp. 270-274, IEEE, 2011 http://dx.doi.org/10.1109/PCI.2011.27, AWARD: BEST PAPER AWARD FOR PCI 2011
2. Apostolos Kritikos and Fragkiskos Chatziasimidis: “SFparser: A Tool for Selectively Parsing SourceForge”, In Proceedings of the 15th Panhellenic Conference on Informatics (PCI '11), pp. 161-165, IEEE, 2011 http://dx.doi.org/10.1109/PCI.2011.42
3. Eleni Constantinou, George Kakarontzas, and Ioannis Stamelos: “Towards Open Source Software System Architecture Recovery Using Design Metrics”, In Proceedings of the 15th Panhellenic Conference on Informatics (PCI '11), pp. 166-170, IEEE, 2011, http://dx.doi.org/10.1109/PCI.2011.36
4. Eleni Constantinou, George Kakarontzas, Ioannis Stamelos: “Open Source Software: How Can Design Metrics Facilitate Architecture Recovery?”, 4th Workshop on Intelligent Techniques in Software Engineering, 5 September 2011 at the European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases (ECML-PKDD), http://arxiv.org/abs/1110.1992v1
5. Skalistis Stefanos, Stamelos Ioannis, Kakarontzas George: *“R.O.D.E. Process: A Configurable Reuse-Oriented Domain Engineering Process”*, International Conference ICIST 2012, ISBN 978-86-85525-10-0, Pages 46-58, 29/2-3/3/2012. Kapaonik, http://www.e-drustvo.org/icist/2012/html/pdf/585.pdf
6. George Kakarontzas, Ioannis Stamelos, Stefanos Skalistis and Athanasios Naskos,*'Extracting Components from Open Source: The Component Adaptation Environment (COPE) Approach'*, In 38th Euromicro Conference on Software Engineering and Advanced Applications, September 5-8, 2012, Cesme, Izmir, Turkey
7. Fotios Kokkoras, Konstantinos Ntonas, Apostolos Kritikos, George Kakarontzas, Ioannis Stamelos, "*Federated Search for Open Source Software Reuse*". In 38th Euromicro Conference on Software Engineering and Advanced Applications, September 5-8, 2012, Cesme, Izmir, Turkey
8. Adnan Causevic, Daniel Sundmark, Sasikumar Punnekkat, “Impact of Test Design Technique Knowledge on Test Driven Development: A Controlled Experiment”, International Conference on Agile Software Development, XP2012, p 138-152, Springer, Malmö, Sweden, Editor(s):C. Wohlin, http://dx.doi.org/10.1007/978-3-642-30350-0_10
9. Adnan Causevic, Sasikumar Punnekkat and Ivica Crnkovic: “An Application Engineering Process Enabling Open-Source Reuse”, presented in the Reusing Open-Source Software Components – (ROSS) Workshop @ ACM SigSoft CompArch 2012, June 25, 2012, Bertinoro, Italy.
10. Apostolos Kritikos, George Kakarontzas, Ioannis Stamelos. "A semi-automated process for open source code reuse". In 5th International Conference on Evaluation of Novel Approaches in Software Engineering (ENASE '10), 24-25 July 2010, Athens, Greece, http://users.teilar.gr/~gkakaron/AkritikoEtAl-SemiAutomatedProcessForOSSReuse.pdf

1.5 OPEN-SME Public Web Site and Contact

For further details please contact:
Prof. Michalis Loupis, Greek Association of Computer Engineers
Greek Association of Computer Engineers, Karageorgi Servias 7, Athens 10563, Greece
Tel: +30-210 3325230, **Fax:** +30-210 7560324
E-mail: open-sme@computer-engineers.gr
Project website address: http://opensme.eu

2 Use and dissemination of foreground

2.1 Section A (public)

TEMPLATE A1: LIST OF SCIENTIFIC (PEER REVIEWED) PUBLICATIONS, STARTING WITH THE MOST IMPORTANT ONES

NO.	Title	Main author	Title of the periodical or the series	Number, date or frequency	Publisher	Place of publication	Year of publication	Relevant pages	Permanent identifiers[5] (if available)	Is/Will open access[6] provided to this publication?
1	Component Certification as a Prerequisite for Widespread OSS Reuse	George Kakarontzas	Electronic Communications of the EASST	Vol 33	EASST		2010		http://journal.ub.tu-berlin.de/eceasst/article/view/449/433 /	yes
2	An empirical investigation on the reusability of design patterns and software packages	Apostolos Ampatzoglou	Journal of Systems and Software	Vol 84, Issue 12	ELSEVIER		2011	2265-2283	http://dx.doi.org/10.1016/j.jss.2011.06.047	no
3	Layer Assessment of Object-Oriented Software: A Metric Facilitating White-Box Reuse	George Kakarontzas	Journal of Systems and Software		ELSEVIER		2012		accepted	no
4	'Extracting Components from Open Source: The Component Adaptation Environment (COPE) Approach'	George Kakarontzas	38th Euromicro Conference				2012			no
5	Elastic Component Characterization with	George Kakarontzas	PCI2011		IEEE		2011	270-274	http://dx.doi.org/10.1109/PCI.2011.27	no

[5] A permanent identifier should be a persistent link to the published version full text if open access or abstract if article is pay per view) or to the final manuscript accepted for publication (link to article in repository).

[6] Open Access is defined as free of charge access for anyone via Internet. Please answer "yes" if the open access to the publication is already established and also if the embargo period for open access is not yet over but you intend to establish open access afterwards.

	Respect to Quality Properties: An Intuitionistic Fuzzy-Based Approach									
6	SFparser: A Tool for Selectively Parsing SourceForge	Apostolos Kritikos	PCI 2011		IEEE		2011	161-165	http://dx.doi.org/10.1109/PCI.2011.42	no
7	Towards Open Source Software System Architecture Recovery Using Design Metrics	Eleni Constantinou	PC1 2011		IEEE		2011	166-170	http://dx.doi.org/10.1109/PCI.2011.36	no
8	Open Source Software: How Can Design Metrics Facilitate Architecture Recovery?	Eleni Constantinou	4th Workshop on Intelligent Techniques in Software Engineering, at (ECML-PKDD)				2011		http://arxiv.org/abs/1110.1992v1	yes
9	R.O.D.E. Process: A Configurable Reuse-Oriented Domain Engineering Process	Stefanos Skalistis	ICIST 2012		ISBN 978-86-85525-10-0		2012	46-58	http://www.e-drustvo.org/icist/2012/html/pdf/585.pdf	yes
10	Federated Search for Open Source Software Reuse	Fotios Kokkoras	38th Euromicro Conference				2012			no
11	Impact of Test Design Technique Knowledge on Test Driven Development: A Controlled Experiment	Adnan Causevic	XP2012		Springer		2012	138-152	http://dx.doi.org/10.1007/978-3-642-30350-0_10	no
12	An Application Engineering Process Enabling Open-Source Reuse	Adnan Causevic	ROSS – CompArch 2012		ACM		2012			no
13	A semi-automated process for open source code reuse	Apostolos Kritikos	ENASE 2010		ENASE		2010		http://users.teilar.gr/~gkakaron/AkritikoEtAl-SemiAutomatedProcessForOSSReuse.pdf	yes

TEMPLATE A2: LIST OF DISSEMINATION ACTIVITIES								
NO.	Type of activities[7]	Main leader	Title	Date/Period	Place	Type of audience[8]	Size of audience	Countries addressed
1	Conference	*AUTH*	*ENASE 2010*	*24/07/2010*	*Athens*	*Scientific community (higher education, Research)*		*EU*
2	Web sites/Applications	EMYPEE	Project Web Site	30/07/2010	Athens	Scientific community (higher education, Research)	10.000.000	World
3	Conference	AUTH	DSM-TP 2010 Summer School	13/09/2010	Lisbon	Scientific community (higher education, Research)		EU
4	Conference	AUTH	ADAPT 2010 Summer School	26/09/2010	Koblenz	Scientific community (higher education, Research)		EU
5	Conference	AUTH	QUATIC 2010	29/09/2010	Oporto	Scientific community (higher education, Research)		EU
6	Workshops	ISS	Open Source Software Components Reuse Workshop	08/03/2011	Kapaonik	Scientific community (higher education, Research) - Industry	30	Serbia, Greece
7	Publication	EMYPEE	1st OPEN-SME Newsletter	08/03/2011	Athens	Scientific community (higher education, Research) - Industry - Civil society - Policy makers - Medias	120	EU
8	Thesis	MAELARDALENS HOEGSKOLA	Software Testing in Agile Development	02/07/2011	Malardalen	Scientific community (higher education, Research)	1000	Sweden
9	Web sites/Applications	MAELARDALENS HOEGSKOLA	University Web Site	02/09/2011	Malardalen	Scientific community (higher education, Research)	1000000	Sweden
10	Workshops	TEKNIKBYN	OPEN-SME Workshop	26/01/2012	Vasteras	Scientific community (higher	40	Sweden

[7] A drop down list allows choosing the dissemination activity: publications, conferences, workshops, web, press releases, flyers, articles published in the popular press, videos, media briefings, presentations, exhibitions, thesis, interviews, films, TV clips, posters, Other.

[8] A drop down list allows choosing the type of public: Scientific Community (higher education, Research), Industry, Civil Society, Policy makers, Medias, Other ('multiple choices' is possible).

		SCIENCE PARK VASTERAS AB				education, Research) - Industry		
11	Conference	ISS	ICIST 2012	29/02/2012	Kapaonik	Scientific community (higher education, Research) - Industry	60	EU
12	Workshops	EMYPEE	OPEN-SME Workshop	17/02/2012	Athens	Scientific community (higher education, Research) - Industry	40	Greece, Cyprus, Serbia, Netherlands
13	Workshops	ETEK	OPEN-SME Workshop	30/05/2012	Nicosia	Scientific community (higher education, Research) - Industry	30	Cyprus, Greece, Sweden
14	Conference	MAELARDALENS HOEGSKOLA	ACM SigSoft COmpARch 2012 Conference - ROSS	25/06/2012	Bertinoro	Scientific community (higher education, Research) - Industry	30	Greece, Sweden, Italy
15	Publication	EMYPEE	2nd OPEN-SME Newsletter	30/06/2012	Athens	Scientific community (higher education, Research) - Industry - Civil society - Policy makers - Medias	500	EU
16	Presentations	AUTH	Software Recycling	02/07/2012	Groningen	Scientific community (higher education, Research)		Netherlands

2.2 Section B (Public)

Part B1

TEMPLATE B1: LIST OF APPLICATIONS FOR PATENTS, TRADEMARKS, REGISTERED DESIGNS, ETC.					
Type of IP Rights[9]:	Confidential Click on YES/NO	Foreseen embargo date dd/mm/yyyy	Application reference(s) (e.g. EP123456)	Subject or title of application	Applicant (s) (as on the application)
	NOT APPLICABLE				

[9] A drop down list allows choosing the type of IP rights: Patents, Trademarks, Registered designs, Utility models, Others.

Part B2

Type of Exploitable Foreground[10]	Description of exploitable foreground	Confidential Click on YES/NO	Foreseen embargo date dd/mm/yyyy	Exploitable product(s) or measure(s)	Sector(s) of application[11]	Timetable, commercial or any other use	Patents or other IPR exploitation (licences)	Owner & Other Beneficiary(s) involved
Commercial exploitation of R&D results	The OPEN-SME generic and customisable Domain Engineering Process which guides the definition, implementation, and maintenance of Domain Architectures for any given Application Domains,as well as the identification and adaptation of available OSS components to these Domain Architectures.	No		Domain Engineering Process (Method)	Software Development	In use	Patenting examined	All SMEs/SME-AGs
Commercial exploitation of R&D results	The OPEN-SME generic and customisable Application Engineering Process that will guide the exploitation of the Domain Engineering outcomes in the framework of a systematic, component-based, and reuse-oriented Software Product Development Methodology	No		Application Engineering Process (Method)	Software Development	In Use	Patenting examined	All SMEs/SME-AGs
Commercial exploitation of R&D results	The Open Source Search Engine (OCEAN) that provides unified access to existing OSS search engines and allows the initial retrieval of open source software modules to be	No		OCEAN Web-Site	Software Development	In Use	Licensing based on GPL	All SMEs/SME-AGs

[19] A drop down list allows choosing the type of foreground: General advancement of knowledge, Commercial exploitation of R&D results, Exploitation of R&D results via standards, exploitation of results through EU policies, exploitation of results through (social) innovation.

[11] A drop down list allows choosing the type sector (NACE nomenclature) : http://ec.europa.eu/competition/mergers/cases/index/nace_all.html

Type of Exploitable Foreground[10]	Description of exploitable foreground	Confidential Click on YES/NO	Foreseen embargo date dd/mm/yyyy	Exploitable product(s) or measure(s)	Sector(s) of application[11]	Timetable, commercial or any other use	Patents or other IPR exploitation (licences)	Owner & Other Beneficiary(s) involved
	analysed and adapted by the Domain Engineering Activities. The OCEAN portal is made publicly available (as a service) for use by third-party organisations.							
Commercial exploitation of R&D results	The OPEN-SME Component Adaptation Environment (COPE) generic tooling framework specification and the different COPE instances (as per selected Application Domain requirements) that support in a holistic manner the different activities of the Domain Engineering Process	No		COPE tool	Software Development	In Use	Licensing based on GPL	All SMEs/SME-AGs
Commercial exploitation of R&D results	The OPEN-SME Component Repository and Search Engine (COMPARE) allows software re-users to effectively search, browse, and retrieve the assets produced by the Domain Engineering Process. It also provides a communication bus supporting the effective exchange and processing of structured information flows between the software reuse stakeholders.	No		COMPARE Web Site	Software Development	In Use	Licensing based on GPL	All SMEs/SME-AGs
General advancement of knowledge	The OPEN-SME generic Business Models and Cost Models pertaining to the commercialisation of the OPEN-SME approach in real-world	No		OPEN-SME generic Business Models and Cost Models	Open Source Software	In Use	Available publicly	All SMEs/SME-AGs

Type of Exploitable Foreground[10]	Description of exploitable foreground	Confidential Click on YES/NO	Foreseen embargo date dd/mm/yyyy	Exploitable product(s) or measure(s)	Sector(s) of application[11]	Timetable, commercial or any other use	Patents or other IPR exploitation (licences)	Owner & Other Beneficiary(s) involved
	business cases.							
Commercial exploitation of R&D results	The OPEN-SME generic and customisable Domain Engineering Process which guides the definition, implementation, and maintenance of Domain Architectures for any given Application Domains, as well as the identification and adaptation of available OSS components to these Domain Architectures.	No		Domain Engineering Process (Method)	Software Development	In use	Patenting examined	All SMEs/SME-AGs

For further details on the results, please refer to section 1.3

For further details on the exploitation strategy, please refer to section 1.4

2.3 Report on societal implications

Replies to the following questions will assist the Commission to obtain statistics and indicators on societal and socio-economic issues addressed by projects. The questions are arranged in a number of key themes. As well as producing certain statistics, the replies will also help identify those projects that have shown a real engagement with wider societal issues, and thereby identify interesting approaches to these issues and best practices. The replies for individual projects will not be made public.

A General Information *(completed automatically when **Grant Agreement number** is entered.*	
Grant Agreement Number:	FP7-SME-2008-2 – 243768
Title of Project:	OPEN-SME
Name and Title of Coordinator:	Dr. Michalis Loupis

B Ethics	
1. Did your project undergo an Ethics Review (and/or Screening)? • If Yes: have you described the progress of compliance with the relevant Ethics Review/Screening Requirements in the frame of the periodic/final project reports? Special Reminder: the progress of compliance with the Ethics Review/Screening Requirements should be described in the Period/Final Project Reports under the Section 3.2.2 *'Work Progress and Achievements'*	***0Yes XNo***
2. Please indicate whether your project involved any of the following issues (tick box) :	***YES***
RESEARCH ON HUMANS	
• Did the project involve children?	
• Did the project involve patients?	
• Did the project involve persons not able to give consent?	
• Did the project involve adult healthy volunteers?	
• Did the project involve Human genetic material?	
• Did the project involve Human biological samples?	
• Did the project involve Human data collection?	
RESEARCH ON HUMAN EMBRYO/FOETUS	
• Did the project involve Human Embryos?	
• Did the project involve Human Foetal Tissue / Cells?	
• Did the project involve Human Embryonic Stem Cells (hESCs)?	
• Did the project on human Embryonic Stem Cells involve cells in culture?	
• Did the project on human Embryonic Stem Cells involve the derivation of cells from Embryos?	
PRIVACY	
• Did the project involve processing of genetic information or personal data (eg. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	
• Did the project involve tracking the location or observation of people?	
RESEARCH ON ANIMALS	
• Did the project involve research on animals?	
• Were those animals transgenic small laboratory animals?	

• Were those animals transgenic farm animals?	
• Were those animals cloned farm animals?	
• Were those animals non-human primates?	
RESEARCH INVOLVING DEVELOPING COUNTRIES	
• Did the project involve the use of local resources (genetic, animal, plant etc)?	
• Was the project of benefit to local community (capacity building, access to healthcare, education etc)?	
DUAL USE	
• Research having direct military use	0 Yes X No
• Research having the potential for terrorist abuse	0 Yes X No

C Workforce Statistics

3. Workforce statistics for the project: Please indicate in the table below the number of people who worked on the project (on a headcount basis).

Type of Position	Number of Women	Number of Men
Scientific Coordinator	0	1
Work package leaders	1	5
Experienced researchers (i.e. PhD holders)	0	6
PhD Students	0	1
Other	0	8

4. How many additional researchers (in companies and universities) were recruited specifically for this project?	**2**
Of which, indicate the number of men:	2

D Gender Aspects

5. Did you carry out specific Gender Equality Actions under the project?

○ Yes
■ No

6. Which of the following actions did you carry out and how effective were they?

		Not at all effective				Very effective
❑	Design and implement an equal opportunity policy	○	○	○	○	○
❑	Set targets to achieve a gender balance in the workforce	○	○	○	○	○
❑	Organise conferences and workshops on gender	○	○	○	○	○
❑	Actions to improve work-life balance	○	○	○	○	○
○	Other:					

7. Was there a gender dimension associated with the research content – i.e. wherever people were the focus of the research as, for example, consumers, users, patients or in trials, was the issue of gender considered and addressed?

○ Yes- please specify

■ No

E Synergies with Science Education

8. Did your project involve working with students and/or school pupils (e.g. open days, participation in science festivals and events, prizes/competitions or joint projects)?

○ Yes- please specify

■ No

9. Did the project generate any science education material (e.g. kits, websites, explanatory booklets, DVDs)?

■ Yes- please specify: On-line training material

○ No

F Interdisciplinarity

10. Which disciplines (see list below) are involved in your project?

■ Main discipline[12]: 2.3

■ Associated discipline[12]: 2.2 ○ Associated discipline[12]:

G Engaging with Civil society and policy makers

11a Did your project engage with societal actors beyond the research community? *(if 'No', go to Question 14)*

○ Yes
■ No

11b If yes, did you engage with citizens (citizens' panels / juries) or organised civil society (NGOs, patients' groups etc.)?

○ No
○ Yes- in determining what research should be performed
○ Yes - in implementing the research
○ Yes, in communicating /disseminating / using the results of the project

[12] Insert number from list below (Frascati Manual).

11c In doing so, did your project involve actors whose role is mainly to organise the dialogue with citizens and organised civil society (e.g. professional mediator; communication company, science museums)?	○ ○	Yes No

12. Did you engage with government / public bodies or policy makers (including international organisations)

- ■ No
- ○ Yes- in framing the research agenda
- ○ Yes - in implementing the research agenda
- ○ Yes, in communicating /disseminating / using the results of the project

13a Will the project generate outputs (expertise or scientific advice) which could be used by policy makers?

- ○ Yes – as a **primary** objective (please indicate areas below- multiple answers possible)
- ○ Yes – as a **secondary** objective (please indicate areas below - multiple answer possible)
- ■ No

13b If Yes, in which fields?

Agriculture		Energy		Human rights	
Audiovisual and Media		Enlargement		Information Society	
Budget		Enterprise		Institutional affairs	
Competition		Environment		Internal Market	
Consumers		External Relations		Justice, freedom and security	
Culture		External Trade		Public Health	
Customs		Fisheries and Maritime Affairs		Regional Policy	
Development Economic and Monetary Affairs		Food Safety		Research and Innovation	
		Foreign and Security Policy		Space	
Education, Training, Youth		Fraud		Taxation	
Employment and Social Affairs		Humanitarian aid		Transport	

13c If Yes, at which level?

- ○ Local / regional levels
- ○ National level
- ○ European level
- ○ International level

H Use and dissemination

14. How many Articles were published/accepted for publication in peer-reviewed journals?	**13**
To how many of these is open access[13] provided?	**4**
How many of these are published in open access journals?	**1**
How many of these are published in open repositories?	**3**
To how many of these is open access not provided?	**9**
Please check all applicable reasons for not providing open access:	
■ publisher's licensing agreement would not permit publishing in a repository ❑ no suitable repository available ❑ no suitable open access journal available ❑ no funds available to publish in an open access journal ❑ lack of time and resources ❑ lack of information on open access ❑ other[14]: ……	

15. How many new patent applications ('priority filings') have been made? *("Technologically unique": multiple applications for the same invention in different jurisdictions should be counted as just one application of grant).*		**NONE**
16. Indicate how many of the following Intellectual Property Rights were applied for (give number in each box).	Trademark	
	Registered design	
	Other	
17. How many spin-off companies were created / are planned as a direct result of the project?		**NONE**
Indicate the approximate number of additional jobs in these companies:		

18. Please indicate whether your project has a potential impact on employment, in comparison with the situation before your project:

■	Increase in employment, or	■	In small & medium-sized enterprises
❑	Safeguard employment, or	❑	In large companies
❑	Decrease in employment,	❑	None of the above / not relevant to the project
❑	Difficult to estimate / not possible to quantify		

19. For your project partnership please estimate the employment effect resulting directly from your participation in Full Time Equivalent (*FTE = one person working fulltime for a year*) jobs:	*Indicate figure:* *2*

[13] Open Access is defined as free of charge access for anyone via Internet.
[14] For instance: classification for security project.

Difficult to estimate / not possible to quantify	❑

I Media and Communication to the general public

20. As part of the project, were any of the beneficiaries professionals in communication or media relations?

○ Yes ■ No

21. As part of the project, have any beneficiaries received professional media / communication training / advice to improve communication with the general public?

○ Yes ■ No

22 Which of the following have been used to communicate information about your project to the general public, or have resulted from your project?

❑	Press Release	■	Coverage in specialist press
❑	Media briefing	❑	Coverage in general (non-specialist) press
❑	TV coverage / report	❑	Coverage in national press
❑	Radio coverage / report	❑	Coverage in international press
■	Brochures /posters / flyers	■	Website for the general public / internet
❑	DVD /Film /Multimedia	■	Event targeting general public (festival, conference, exhibition, science café)

23 In which languages are the information products for the general public produced?

❑	Language of the coordinator	■	English
❑	Other language(s)		

***Question F-10*:** Classification of Scientific Disciplines according to the Frascati Manual 2002 (Proposed Standard Practice for Surveys on Research and Experimental Development, OECD 2002):

FIELDS OF SCIENCE AND TECHNOLOGY

1. NATURAL SCIENCES

1.1 Mathematics and computer sciences [mathematics and other allied fields: computer sciences and other allied subjects (software development only; hardware development should be classified in the engineering fields)]

1.2 Physical sciences (astronomy and space sciences, physics and other allied subjects)

1.3 Chemical sciences (chemistry, other allied subjects)

1.4 Earth and related environmental sciences (geology, geophysics, mineralogy, physical geography and other geosciences, meteorology and other atmospheric sciences including climatic research, oceanography, vulcanology, palaeoecology, other allied sciences)

1.5 Biological sciences (biology, botany, bacteriology, microbiology, zoology, entomology, genetics, biochemistry, biophysics, other allied sciences, excluding clinical and veterinary sciences)

2 ENGINEERING AND TECHNOLOGY

2.1 Civil engineering (architecture engineering, building science and engineering, construction engineering, municipal and structural engineering and other allied subjects)

2.2 Electrical engineering, electronics [electrical engineering, electronics, communication engineering and systems, computer engineering (hardware only) and other allied subjects]

2.3. Other engineering sciences (such as chemical, aeronautical and space, mechanical, metallurgical and materials engineering, and their specialised subdivisions; forest products; applied sciences such as

geodesy, industrial chemistry, etc.; the science and technology of food production; specialised technologies of interdisciplinary fields, e.g. systems analysis, metallurgy, mining, textile technology and other applied subjects)

3. MEDICAL SCIENCES

3.1 Basic medicine (anatomy, cytology, physiology, genetics, pharmacy, pharmacology, toxicology, immunology and immunohaematology, clinical chemistry, clinical microbiology, pathology)

3.2 Clinical medicine (anaesthesiology, paediatrics, obstetrics and gynaecology, internal medicine, surgery, dentistry, neurology, psychiatry, radiology, therapeutics, otorhinolaryngology, ophthalmology)

3.3 Health sciences (public health services, social medicine, hygiene, nursing, epidemiology)

4. AGRICULTURAL SCIENCES

4.1 Agriculture, forestry, fisheries and allied sciences (agronomy, animal husbandry, fisheries, forestry, horticulture, other allied subjects)

4.2 Veterinary medicine

5. SOCIAL SCIENCES

5.1 Psychology

5.2 Economics

5.3 Educational sciences (education and training and other allied subjects)

5.4 Other social sciences [anthropology (social and cultural) and ethnology, demography, geography (human, economic and social), town and country planning, management, law, linguistics, political sciences, sociology, organisation and methods, miscellaneous social sciences and interdisciplinary , methodological and historical S1T activities relating to subjects in this group. Physical anthropology, physical geography and psychophysiology should normally be classified with the natural sciences].

6. HUMANITIES

6.1 History (history, prehistory and history, together with auxiliary historical disciplines such as archaeology, numismatics, palaeography, genealogy, etc.)

6.2 Languages and literature (ancient and modern)

6.3 Other humanities [philosophy (including the history of science and technology) arts, history of art, art criticism, painting, sculpture, musicology, dramatic art excluding artistic "research" of any kind, religion, theology, other fields and subjects pertaining to the humanities, methodological, historical and other S1T activities relating to the subjects in this group]

3 Final Report on the Distribution of the European Union Financial Contribution

This report shall be submitted to the Commission within 30 days after receipt of the final payment of the European Union financial contribution.

