PL.O.T EDV Planungs- und HandelsgesmbH
Franz-Josefs-Kai 33
1010 Wien
Austria

eMS
Harmonised Implementation Tools
Software Architecture

Version:0.7
Date: 24.06.*2014*

Version history

Änderung			Name	Anmerkung (geänderte Kapitel, Art der Änderung)
Lfd. Nr.	Datum	Version		
01	18.03.2014	0.1	Horst Foidl	First Draft
02	28.03.2014	0.2	Horst Foidl	Chapter 1 to 4 completed in first version
03	02.04.2014	0.3	Horst Foidl	First draft of Chapter 5
04	08.04.2014	0.4	Horst Foidl	Details on 5.2 and 5.11
05	18.04.2014	0.5	Horst Foidl	Details on Details on 5.13 and 6
06	06.06.2014	0.6	Horst Foidl	Internal Review wit Dev-Team
07	24.06.2014	0.7	Horst Foidl	Authentication, Authorization, Threats with Dev-Team

Table of Contents

1 Introduction

1.1 Background

With the project eMS (eMonitoring System) INTERACT Point Vienna will provide a tool to European Territoral Cooperation Programs. Those programs are developed by two or more EU member states and eve member candidates and have the objective to support territorial cooperation. This tool will be provides to future programs to help them operating the program and fulfil the controlling and reporting tasks on a project and program level.

1.2 Purpose of this Document

This document describes the software architecture of eMS starting at a bird's eye view. The logical view of the system's architecture gives an overview of the modules the system is composed of. Based on the requirements corresponding to the software architecture the technologies and tools used for developing the project are explained.
This document supports the development process of the project as well as the maintenance phase giving a perfect stating point to software developers to get deeper into the system's source code.

1.3 Management Summary

TBD

2 Software Architecture Requirements

2.1 General

Software architecture is one main factor influencing quality of a software system as it determines the "construction" of the system. This construction determines the behaviour of a system in many aspects and therefore needs to be considered carefully. The following activities during the project implementation are as important as the architecture. It is important to understand, however, that architecture alone cannot guarantee functionality or quality. Poor downstream design or implementation decisions can always undermine an adequate architectural design. The quality assurance process of a project has to cover these topics.

2.2 Non-functional Requirements

A non-functional property denotes a feature of a system that is not covered by its functional description. A non-functional property typically addresses aspects related to the reliability, compatibility, cost, ease of use, maintenance or development of a software system.

Non-functional property	**Referenced requirement (Chapter 9)**
maintainability	3.17, 3.18, 7.15, V31, V22, VC12.2
flexibility	1.1, 1.2, 1.5, 1.15, 10.14, 11.5, 13.3, V25.2, V27, K3.5, A21.03.01, A27.03.01
adaptability	1.17, 1.18, 1.19, 1.20, V28, V29
reusability	
reliability	1.37, 1.38, 1.39, 1.40, 3.11
robustness	1.35, 1.52, 1.61
understandability	1.4, 1.12, 2.9, A17.03.01
scalability	PB1*, PB2*, A17.03.02*
security	1,62, 1.64, 1.65, 1.66, 2.1, 3.7, 3.29, 4.20, V25.3, V25.4
stability	
manageability	1.9, 1.10, 1.11, A17.03.03
accessibility	
portability	
interoperability	K1.3
testability	
traceability	
interchangeability	
performance	
througput	
latency	
capacity	PB1*, PB2*, A17.03.02* , V34

effciency	
degradation	

* Conflicting capacity requirements specifying the number of concurrent users is converted into requirement for scalability. In terms of requested capacity the maximum value is considered as the requirement to be fulfilled.

2.3 Functional Requirements

The functional requirements listed in this chapter are those relevant for the software architecture. The entire requirements relevant for the requested functionality are covered by the functional design documents.

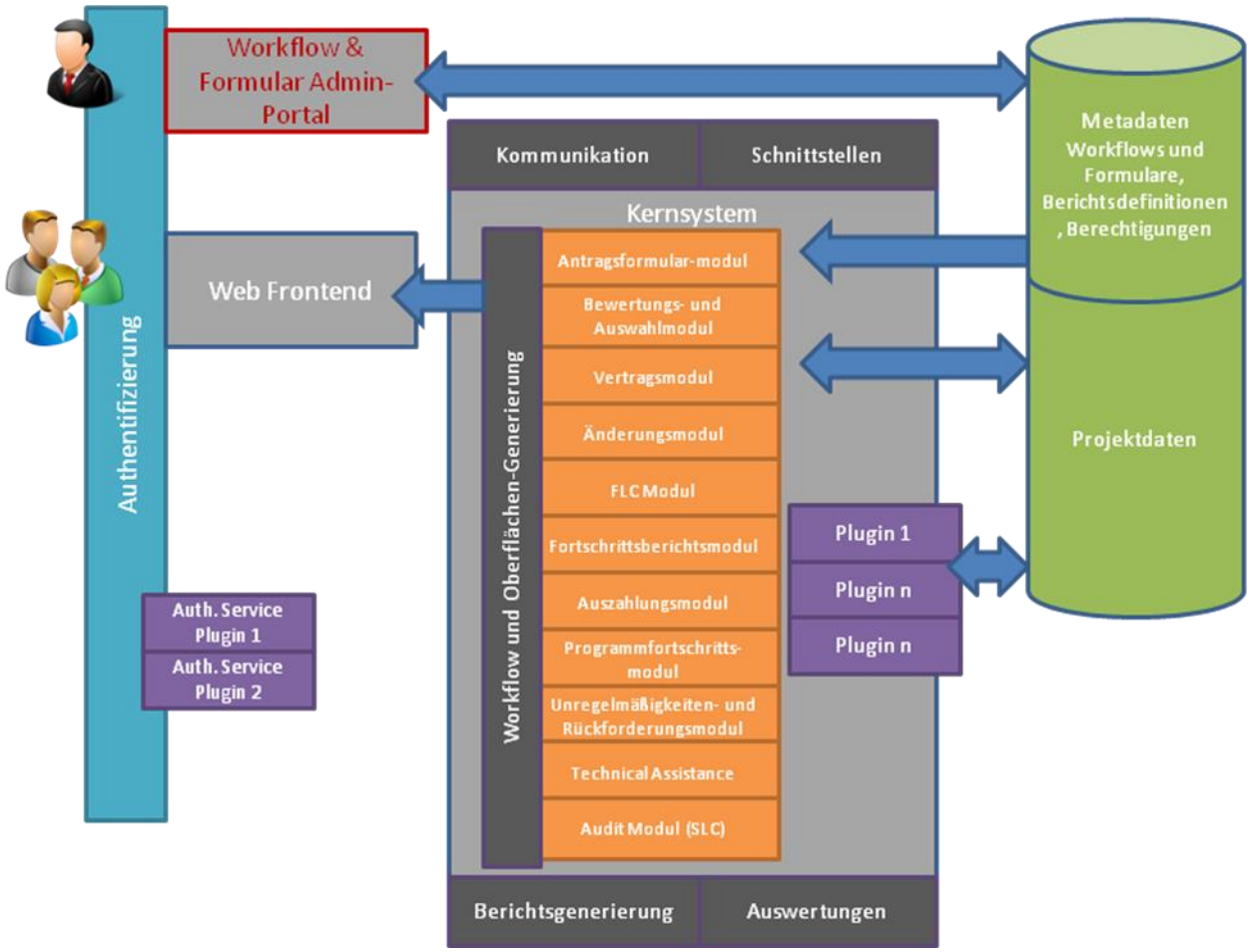
Functional Requirement	**Requirements relevant for architecture (Chapter 9)**
functionality	1.13, 1.14, 1.71, 1.72, 1.73, 1.74, 2.2, 2.7, 2.15, 3.26, V25.1, A17.03.04

3 Logical View on System Architecture

3.1 Top Level Overview

To gain the requested flexibility the solution's core component is composed of a flexible database driven workflow module. Some parts of the processes are under strict regulation some of them need to be very flexible in terms of adaptability to different programs. Additionally the architecture takes into account for the special requirements of the solution's lifecycle as a multi site installation.

The other components of the core module are communication, interfaces, statistics and reporting. The user interface for the end users, the administrative user interface and data storage complete the list of modules from a functional view. Authentication module and the Plugins are technology driven components guaranteeing flexibility and maintainability of the solution.



The following paragraphs describe the top level building blocks of the software architecture. The following technical design maps these building blocks to individual software components. Each component's functionality is specified in the corresponding functional design documentation.

Requirements to fulfil: 1.64

3.2 Workflow

The workflow module needs to cover the ability to adapt the functional processes to the needs of individual programs. Some of the processes are strongly regulated; some of the processes have to be very flexible. This refers to the structure of the processes as well as to the information gathered from the user in the individual process steps.
The architecture of the workflow module respects this situation by holding the configuration data about process structure, data entry requirements and workflow control in a specific segment of the database.
The design of the functional processes decides on the amount of flexibility versus the implementation based workflow rules.
To allow individual programs to setup their own functional rules the workflow module provides a plug-in interface to snap-in individual business rules.

Requirements to fulfil: 1.1

3.3 Web-Frontend

The web-frontend presents the use4r interface to the user in a easy to use and intuitive way. The user can easily identify which process he is working on, which step he is currently processing. The integrated help feature offers at any time descriptive information to explain expected input and functionality.
The development of the user interface is done according to WCAG) 2.0 (http://www.w3.org/TR/2008/REC-WCAG20-20081211/) and additional features like autocomplete and keyboard shortcuts ease data entry.
The core system elements allow a flexible configuration of processes and a customization of the user interface. All information presented in the user interface (text elements, labels, help information, tool tips, etc) are presentenced in the language the user chooses. The administrator has tools available to prepare the languages available to the users and customize processes.
The implementation of the user interface is according to WCAG and ARIA guidelines and uses responsive design methodology. So the user interface is available to be uses on desktop PCs and mobile devices as well. It automatically adapts to the properties of the screen used and also with regard to the mobile devices to orientation landscape or portrait.
Apart from displaying the user interface in different languages the core system is able to handle multilingual input data and stores input accordingly in the database. So it is possible to generate reports in different languages.

Requirements to fulfil: 1.2, 1.4, 1.17, 1.18, 1.20, 1.37, 1.38, 1.39, 1.40, 1.71, 1.73, 3.11, V28, V29, K1.3, K3.5, A17.03.01, PB2, A17.03.02, A17.03.04

3.4 Datastore

The Datastore module consists of different segments holding different type of information facilitating different technologies.

Information to be held by the Datastore module is:

- Functional data (structured) gathered from the users during the execution of the processes
- Unstructured information gathered from the users during the execution of the processes
- Workflow definition for the functional processes of a program
- Functional Configuration data

The main issue for the design of the Datastore is that all segments need to be kept in a consistent state. Depending on the technology chosen to implement the storage, special care has to be taken on the transactional behavior of the storage.
The security concept has to ensure that the storage can be accessed only according to the user rights as granted in the administrative portal.
The administrator has to be provided with functions to determine the maximum amount of storage usable for unstructured data.

Requirements to fulfil: 1.35, 1.52, 1.61, 1.62, 3.18, V34

3.4.1 Structured Program- and Project Data

The structured storage holds information about workflow meta data, process data and configuration data. The data model for the process data has to be a kind of hybrid model, holding information of the strictly specified processes in a specific segment of the data model, which is enhanced by a generic data model, holding the information of the generic type of processes.

3.4.2 Unstructured Program- and Project Data

Unstructured storage holds information gathered in meta data, process data and configuration data in terms of templates, documents, attachments etc.

3.4.3 Workflow Definition Data

The workflow definition data is a specific type of structured information. It describes the workflow at the necessary level depending on the implementation (fixed/generic) of the corresponding functional process.

3.4.4 Functional Configuration Data

The functional configuration data is a specific type of structured information. It is kept strictly separated from the technical configuration data. Refer to chapter **Error! Reference source not found.** for requirements on functional configuration.

3.5 Admin-Frontend

This user interface is foreseen to be used by the system administrators to customize the system to the needs of a specific program. Via this user interface the individual processes are defined and prepared for use by end users: turn on/off optional steps, optional/additional data input, validation rules, help information etc.

Requirements to fulfil: K3.5

3.5.1 Workflow Definition

The workflows of the individual processes and the input forms associated with them are prepared by the administrators. Administrators can modify the number and the sequence of input elements in an intuitive graphical user interface. Input elements are described with their data type, validation rules, optional/mandatory, default value and the specific help information presented to the user in different languages as needed.
With regard to process definition the administrator can define the status information attached to individual steps and the possible change of status information.

Requirements to fulfil: 1.5, 3.17, 3.26, 3.29, 7.15, A21.03.01, A21.03.01

3.5.2 Managing Reporting Templates

This spart of the administrative frontend offers functionality to define reporting templates, which are used to generate reports. With integrated WYSIWYG Editor the administrator can easily manage the reporting templates. Placeholders within the template define the location where structured information and/or unstructured information from the database will be inserted into the report.

Requirements to fulfil: 1.9, 1.10, 1.12, 1.14

3.5.3 Managing Statistics Templates

In the same way as it is done for reports the administrator can manage statistics templates as well. Statistics reports can be in form of tables and/or business charts.

Requirements to fulfil: 1.10

3.5.4 Managing Users and Privileges

Managing users and privileges covers all modules. User roles are defined for administrators and end users in the same way. Based on the workflow definition of the individual processes, the privileges of specific user roles can be specified down to field level. The authorisation concept covers the authorisation of individual functions like: confirmation of process steps, generating reports, manages

user privileges, etc. This information is used to dynamically generate the user interface according to a specific user's membership to user roles. To respond to the great flexibility of the customisation process user privileges are organised into different levels. This allows to assign user privileges according to the correct granularity to reflect different organisational models of individual programs.
On the basis of the defined user roles individual users are managed giving them membership to specific user roles or not.

Requirements to fulfil: 1.15, 3.7, 4.20, V25.3

3.5.5 Online Help System

The online help system is part of the user interface and provides information to the user. This information is managed as part of workflow definition and can be prepared in different languages. Help information is structured using and HTML based editor. The core system elements of the user interface links this information to the correct interface element, like buttons, input field, entire screens etc.

Requirements to fulfil: 1.74

3.5.6 Functional Configuration

Nur die architekturrelevanten Konfigurationsmöglichkeiten, sofern nicht bereits in den anderen Modulen beschrieben sind

Requirements to fulfil: 1.1, 7.15, 1.2, 1.5 20, 1.9, 1.17, 1.18, 1.19, 1.20, 1.15, 10.14, 11.5, 13.3, 3.17, 3.18, V31, A17.03.03

3.6 Communication Module

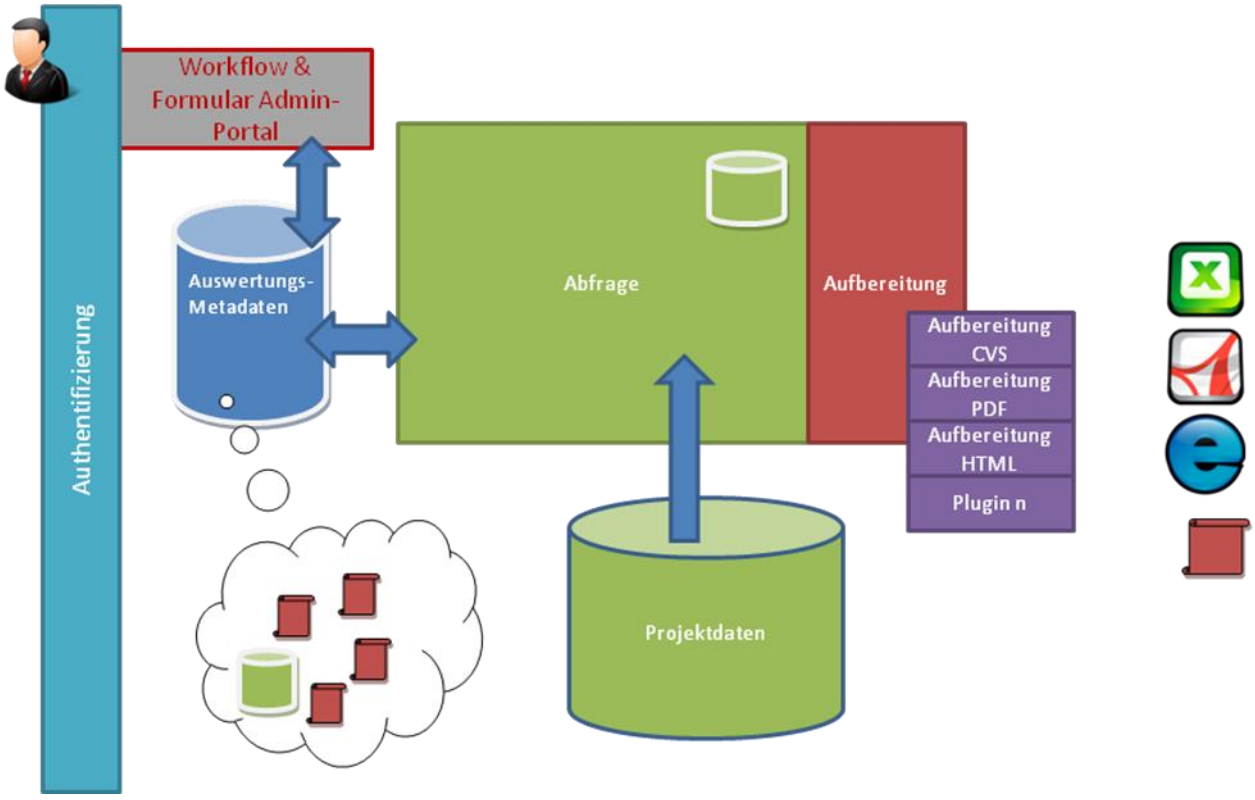
The Communication Module consists of all functions that allow exchange of messages between participants of processes. Messages can be entered manually or triggered automatically by the system. The user interface is designed like a web mail system, so it can be used very easily.
Messages can consist of text and additionally attachments of any document types. Users may prepare message templates to facilitate repeating tasks. Messages are automatically linked to processes and stored in the correct context. On the receipt of a message a notification can be sent to a user's email address.
The system generates automatically messages based on a time schedule to inform users about specific situations and necessities (reporting period). The administrator manages the time schedule ans the corresponding message templates.

Requirements to fulfil: 2.1, 2.2, 2.7, 2.9, 2.15

3.7 Statistics

The statistics module provides all functions necessary to generate statistics reports. Reports can be generated manually via the user interface or based on a time schedule. Report templates can be prepared and stored by the administrator for later execution.

Templates may include data tables and/or business graphics generated from data stored in the monitoring database. Users granted with the correct privileges may execute the templates and generate statistics. The result is formatted as PDF document and can additionally be downloaded as CSV file.



Every data field of the monitoring database can be included into a report. All available fields are organized in a pool which is structured in a way that the end user need not know details on the database structure.

The statistics module is based on the well known reporting engine Jasper-Reports. This is available as a open source solution and as commercial solution. This project uses the open source solution, which can be integrated seamlessly into the application. Reporting templates are stored in XML documents on the server.

This concept allows individual programs to upgrade to the commercial edition of Jasper-Reports (Jaspersoft BI Professional oder Jaspersoft BI Enterprise) in case they want extend the reporting featured up to a data warehouse solution.

Requirements to fulfil: 1.11

3.8 Reports

Reports are based on reporting templates prepared by the administrator and generated from data stored in the database and/or documents. They can be generated on demand using the user interface or based on a time schedule. Reports can be modified after generation and stored in a final version as PDF- or HTML-document.

Requirements to fulfil: 1.11, 1.13, 1.19, 10.14, 11.5, 13.3

3.9 Functional Processes

The functional processes (workflows, e.g.: Application Form) are prepared by the administrator using the configuration tools provides in the administrative portal. All the necessary parameters reflection specific needs in the user interface, statistics, reporting etc. Are provides by the core system of eMS. In case of missing parameters needed by a specific program, the core system provides a programming interface allowing the integration of individual source code (plug-in).

All workflows described in the terms of reference are implemented by the means described above. Processes are configured using the configuration tools and at some points will be enhanced by plug-ins. Details on the description of the individual workflows are described in the functional specification corresponding to the processes.

Requirements to fulfil: V22, V25.1, V25.2, V27, VC12.2, PB1

3.10 Interfaces

eMS provides data exchange interfaces to transfer data from/to the European Commission. The EU provides a set of web service interfaces to accomplish this task. The following data is exchanged:

- Progress Reports
- Payment Claims, Irregularities and recoveries
- Currency Exchange rates

Requirements to fulfil: 1.72

3.11 Authentication

eMS is prepared to support different authentication methods in an installation. Authentication via Userid and Password is available out-of-the-box. Additionally eMS provide a methodology to implement more sophisticated authentication mechanisms.

Requirements to fulfil: 1.65, 1.66, V25.4

3.12 Plug-ins

As mentioned above – although eMS provides a very broad range of parameterisation – at some specific situations this will not be sufficient during the customisation of functional processes. This is why eMS provides the plug-in interface. Using a Plug-in additional data and/or algorithms can be integrated into specific workflows for individual programs. Plug-ins are capable of:

- Provide additional elements within the user interface,
- Customize validation routines,
- Enhance the database model,
- Functional enhancements to existing workflows (e.g. electronically sign documents).

Details on the specification of the plug-ins needed are described in the functional specification of the corresponding processes. Plug-ins implemented by individual programs are reviewed from INTERACT to ensure compatibility with the core system.

Requirements to fulfil: V22, V25.1, V25.2

4 Dynamic Behaviour of the Architecture

4.1 General Considerations

In Terms of flexibility of the processes to be implemented, a hybrid architecture was designed. There are processes – or to speak exactly “levels of processes” -, which are designed strictly without the need of individual adaption by a program and others which need to be adapted by individual programs referring to the structure of the process and the amount of data entered/processed in specific sprocess steps.

Functional design of every single functional process is responsible to find the correct usage of fixed and generic implementation to gain the optimum. Architecture offers both implementations.

4.2 Dependency of Processes

At a Program level processes are of a fixed structure. Individual steps are driven by a time schedule an can be repeated as often as needed (first call, second call,...). Application forms associated to a call need to be adapted in terms of amount of data entered, the structure of the form is fixed. The Evaluation process of the submitted project proposals needs to be very flexible as a different number of evaluators and criteria to evaluate are to be respected. Proposals being evaluated positively are lead to the contracting process, which is of a fixed structure. After finalizing contracting projects are started and projects deliver reports periodically. The frequency and type of reports requested depend on the program and are mainly depended on a time schedule. Payment is dependent on the reporting periods. Payment requests are to be checked during an approval process, which needs to be defined in a flexible way in terms of authorities involved. Projects terminate after the final report and the final payment.

4.3 Generic vs. Fixed Model

As describes in the preceding paragraph a hybrid model of describing functional processes and re correlated input forms has to be used. There are a lot of requirements on flexibility and adaptability of the user interface, which can be fulfilled easily in a generic model. On the other hand generating reports and statistics can easily be created on a fixed data model, whereas this task can be painful on a generic data model.

Details on how the architecture deals with this situation, can be found in the following chapter describing the implementation guidlines.

5 Implementation Guidelines

5.1 Architectural Paradigm

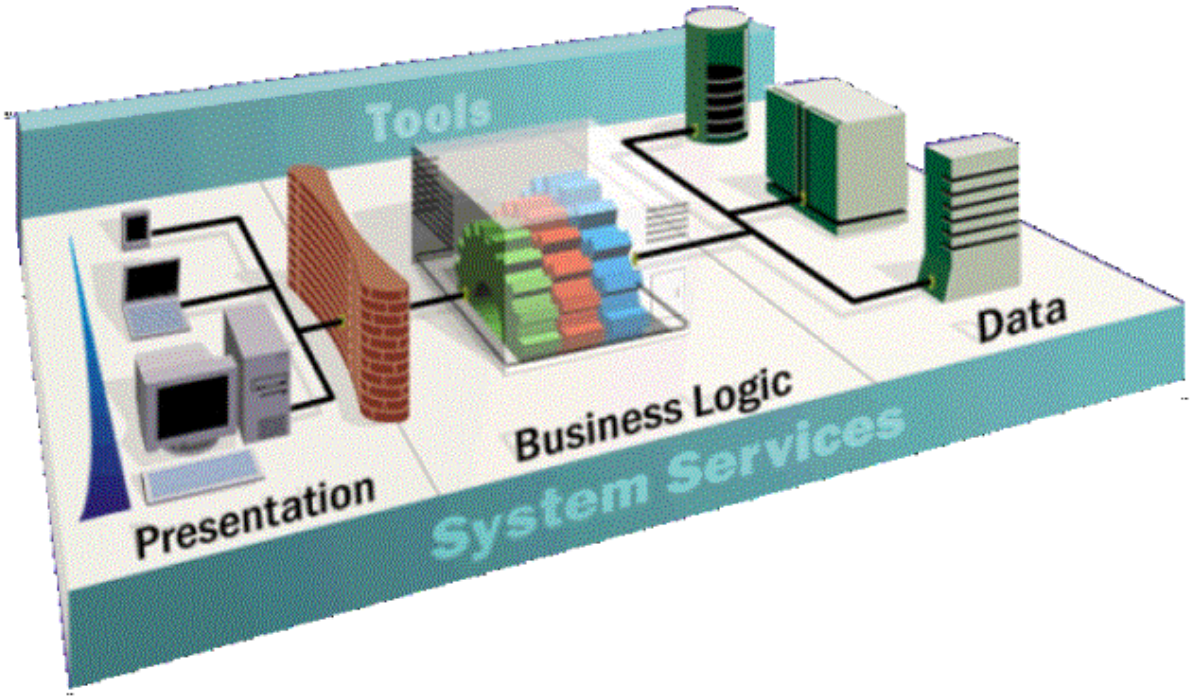
In the basic principles of eMS the development of a web based application was requested. The architectural patterns use to implement eMS follow state of the art J2EE patterns. At those points where specific modifications and/or enhancements are made, these are explained in detail. Otherwise only the used pattern will be referenced. The most commonly known paradigms are the Multi-Tier Application model and the MVC (Model View Controller) concept. Both of them are the fundament of the software architecture used in eMS.
In general terms the eMS application is a web portal offering different entry-points facilitating stronger or weaker authentication technologies. Those entry-pints may be used by individual users or by a CMS based information web site of a program.

5.2 eMS Software Architecture

5.2.1 Three Tier Application Model

The software architecture follows the Multi-Tier-Application model. This guaranties the scalability and robustness of a web based application. This model divides the application into three distinct service layers. Each service layer has a specific task an communicates with the other layers through wel defined interfaces.
Apart from the task a service layer has to accomplish, service layers have to follow specific technical guidelines. The three layers are:

- Presentation Layer
  This layer implements the user interface including all functions that present information and interact with the user. In a typical implementation this layer is session related.
- Business Logic Layer
  This layer implements the business logic representing the algorithms for processing and checking the data. This layer has to be implemented as a stateless transactional service following the ACID-principle.
- Persistence Service Layer
  This layer implements retrieving and storing data. A typical storage implementation is the use of a relational DBMS, but it is not limited to using a DBMS. This has to be done within the transactional rules defined by the Business Logic Layer.



The main aspect for a successful implementation of this concept is the consequent separation of the service layers by well defined interfaces. This has to be especially stressed between Presentation Layer and Business Logic Layer, as shown as a wall in the picture above. This wall has to guarantee the clear separation of the user interface from the business logic.

User Interface Components

Referring to a web based application you have to keep in mind that the implementation of the user interface spreads across different machines facilitating different technologies.

Business Logic Services

These Services implement the algorithms necessary to fulfil the specified business functionality. Each service request is executed as a single stateless transaction. Based on specific products/technologies used to implement the service transactions may be nested.

Persistence Services

These components are typically used by the business logic services and retrieve and store data. As they are running within the transactional context of the business logic services they have to implement the same transactional rules as defined by the business logic layer.

5.2.2 Model View Controller Concept

This concept is another paradigm with the objective to separate user interface implementation from business logic implementation. It is not such a high level concept as the Tree-Tier-Application-Model shown in the preceding paragraph. It more focused on the implementation concept for a GUI based applicaiton.

Again there are three different components involved:

The Model

This component holds the information about a business object and offers its properties to retrieve information about the object and methods to manipulate the object including retrieving and storing information from the storage. So the model is typically session related.

The View

The View implements the component that generates the user interface based on the information held by The Model. All algorithms necessary to present the information are implemented within the view.

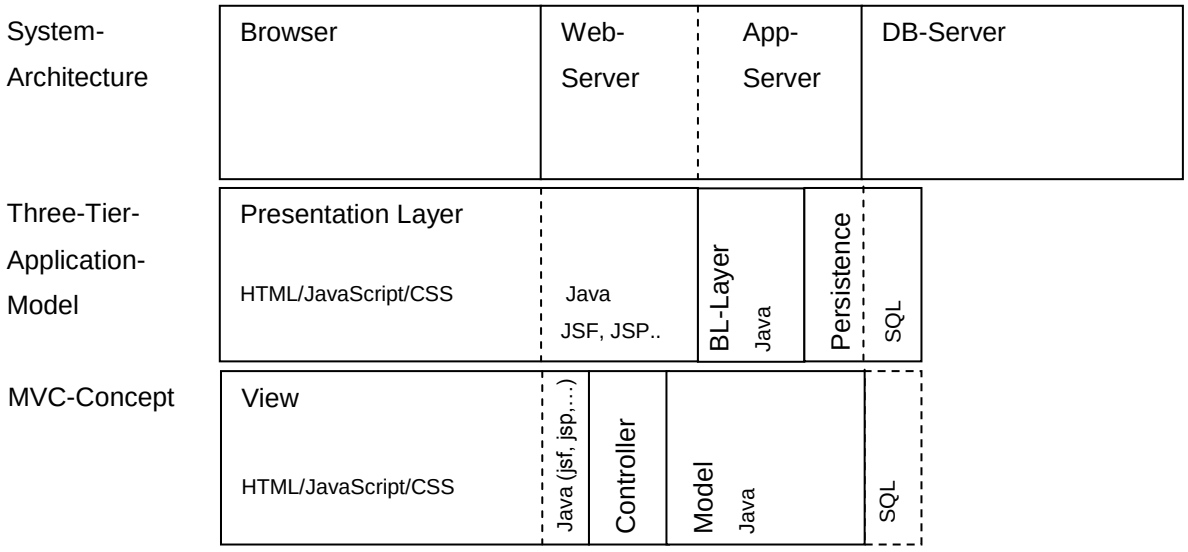
The Controller

The controller implements the workflow through the application including all available branches based on decisions taken by the user or depending on the constellation of data processing.

On the contrary to the Three-Tier-Application model where no commonly available framework exists to implement the concept, there are some frameworks available to implement the MVC-concept in Java and other programming languages.

5.2.3 Putting it all together

The following illustrations show the mapping to the elements of the system architecture of a typical web based application. For reasons of clarity and simplification Web-Server and Application-Server are shown as one box.



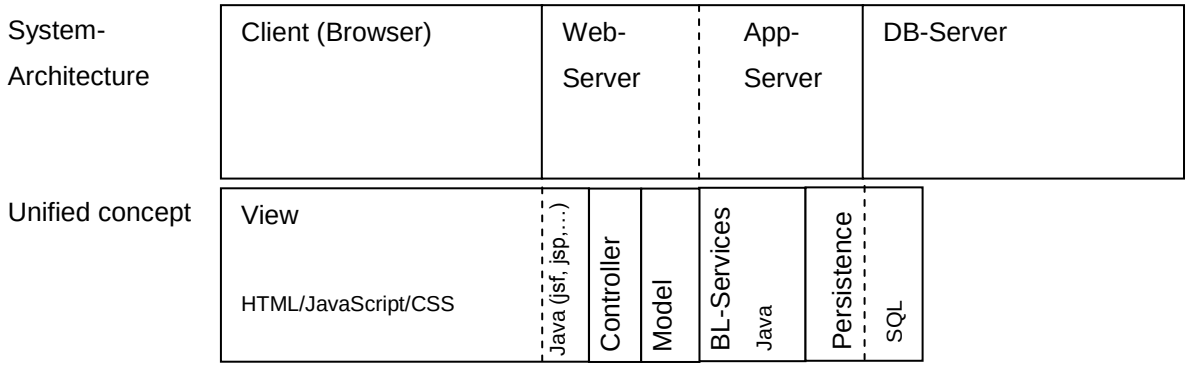
As expected based on the different approach of the two concepts the components proposed by the two concepts have different scope within the entire application. In the Tree-Tier-Application-Model the Presentation spans across the browser on the user's machine and the Web-Server. Technologies involved on the client side part of the presentation layer are HTML, JavaScript and CSS. The server

part is implemented using Java. Business Logic Layer and Persistence Layer are implemented in Java as well. Persistence Layer is generating SQL-Statements to retrieve and store data. Within the MVC Concept the view spreads again across the user's browser and the web-Server. The implementation follows the same rules as the presentation layer. The controller resides completely within the scope of the web-server. The model spans across Web server and application server including business logic and persistence tasks.

5.2.4 Architecture Guidelines

Based on the principles shown in the chapter above, the guidelines for implementing the software architecture in eMS is shown in this paragraph. The architecture guidelines do not reflect all the requirements shown in chapter 2. It concentrates on the overall principles. The following chapters are showing the implementation guidelines on component level, reflecting the related requirements.

The following illustration shows the unified concept integrating a Three-Tier-Aplication model in an MVC-Implementation.



Having the same scope in both concepts the view has exactly the same role in the unified concept as before. The controller implements the workflow through the application and links the view together with the other components of the architecture. The implementation of the model is limited to the task of holding the information of objects on a session level and providing the interface to the view. The manipulation of object data is moved out to a stateless transactional service implementation, which guarantees the transactional context and utilizes the persistence component to retrieve and store data. The implementation of the persistence service generates the appropriate SQL-statements and may additionally use stored procedures to span its scope to the DBMS.

As mentioned above there are frameworks available to implement the MVC pattern. One of these is "spring". Spring is a powerful framework, which covers a lot more than the MVC pattern, but also handling transactions among other elements. So spring is the framework that fits best the requirements for a successful implementation of eMS. Spring itself doesn't cover the implementation of the view, but offers different options implementing the view. A very flexible methodology implementing user interfaces is JSF (Java Server Faces). This allows the separation of design elements of the user interface and the implementation of the presentation logic. So JSF is the technology to be uses for implementing the user

interface of eMS. A similar situation exists for implementing the persistence layer. Spring itself does not cover the scope if the persistence layer, but integrates very well with other frameworks. The persistence framework uses in eMS is hibernate.

Presentation Layer

The Presentation Layer is built following the MVC concept. It is a stateful component utilizing different technologies and languages and the implementation spans across the client machine and the web server.

Business Logic Services

The business logic service – implemented as a stateless transactional Java Bean – following the spring implementation rules for services is activated by the controller. The controller passes all the models needed to fulfil the business transaction.

Persistence Layer

The data access objects (DAOs) used in the persistence layer – implemented as stateless transactional Java Beans – utilize hibernate generated entity classes. For each data table of the database model an entity bean is generated. The methods provided by the DAOs basically reflect the CRUD (create, update, delete) functionality. A specific DAO may offer specific methods based on the functional specification. Sometimes it may make sense to implement some parts of the persistence logic with the DMS utilizing stored procedures or triggers.

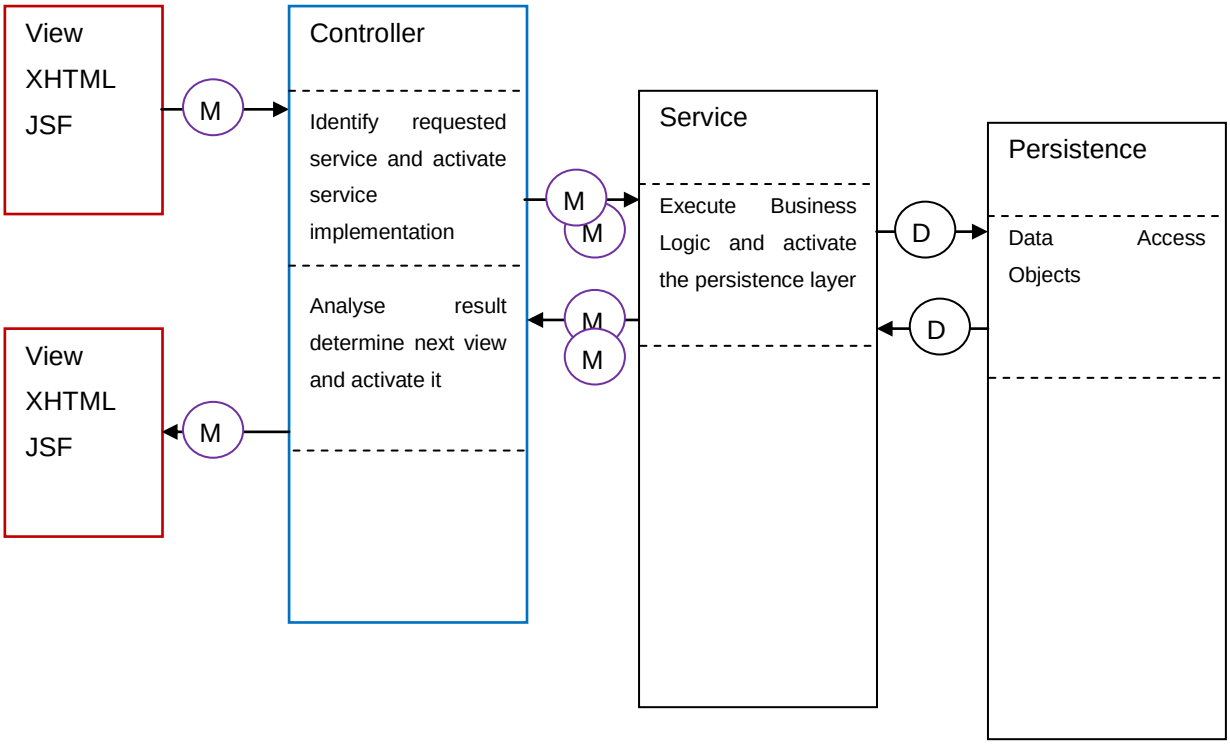
5.3 Application Level

5.3.1 Application Workflow

The entire web application is constructed by a set of individual flows each representing a functional workflow, that can be initiated by a user. It is the responsibility of the controller to implement this workflow, with all its branches depending on the user's interaction and the success of the actions triggered by the user.

Single User interaction flow

The following illustration shows the cooperation of the top level architectural components, when processing a user input.

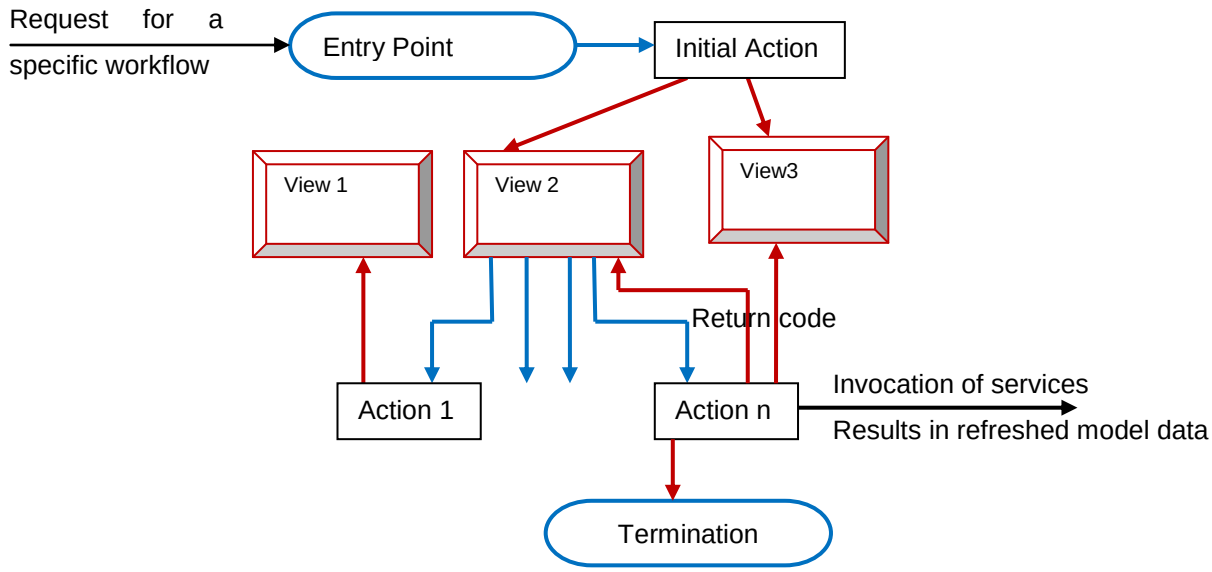


After user has completed data entry the view maps the data from the HTML page to the involved models and activates the associated controller. The controller identifies the request and activates the requested service passing all the models needed to fulfil the request. Within the execution of the business logic the service invokes the persistence layer by activating the appropriate data access objects.

The implementation of this concept has to keep in mind, that the service layer is responsible for mapping DAOs to Models correctly – in both directions. Additionally, when developing the system architecture, the overhead of object serialisation taking place when putting the service layers to different machines has to be considered carefully.

Entire functional workflow

An entire functional workflow is composed of a number of screens and possible actions triggered by user input. The correct flow through all those screens is defined by a set of workflows the user may activate. Every single workflow has one specific entry point, a network of flows through the screens and actions.



The execution of a workflow is requested either by a user request or by processing a workflow. The first step of a workflow is the processing of an initialisation action. The action invokes services, which present data in the models used later on by the view. The return code of the action - derived from the service success - is evaluated by the controller (red arrows) and the initial view is displayed to the user. The view itself may have its own initialisation sequence to deal with the data presented by the models. The user enters data and triggers one of the available actions (blue arrows). Again the return code of the action is evaluated by the controller and the succeeding view is selected and presented to the user.

5.3.2 The Controller

The implementation of the controller is done following the rules of the framework spring. Each individual functional workflow (fixed dialog based workflow for a single task within the user interface) is implemented via a spring web-flow description. The controller implementation is linked to the view to react upon a user having entered data and triggered an action.

The controller relies on the fact that the user interface implementation has filled the correlated models with the data the user has entered and activates the service layer associated with the specific user request.

The controller is implemented by using the spring web-flow component, and consists of:

- The core spring web-flow components

- One specific <Flow-Name>.xml document describing the flow in the way shown in 5.2.4 Entire Application Workflow.
- One <Flow-Name>Flow.java source code document implementing all lifecycle methods managing the flow (e.g. Initialisation of DTS, determining initial view,..).

The flow is initiated with an initial action-state repeats the flow from action-state to view-state to action-state....and so on. The user may trigger actions within the view, which are evaluated within the flow to determine the correct action-state. The action-state invokes methods in the <View-Name>View.java.class (see next paragraph), which returns a return code to the flow. The return code is evaluated within the flow and the succeeding view is selected and presented to the user.
The DTOs necessary to pass data between the Services and Views used in the flow are defined as variables at FLOW-SCOPE within the flow.

5.3.3 View

The view is implemented utilizing JSF. The descriptive part of the presentation is done in a XHTML-document. Different implementations of the JSF specification are available. The one used in eMS is Primefaces. The view displays data held within the models and stores user entered data into the model. Specific presentation logic necessary within the view can be done:

- Within the XHTML-template of a view
- Within the generated JavaScript code of the View
- Within the view implementation class

Every single view is implemented using the prime faces JSF framework, which integrates very well with spring and consists of:

- The core spring and primefaces components
- One <View-Name>.XHTML document holding the view layout information
- One <View-Name>View.java source code document implementing all algorithms for user interaction and the service invocation methods, passing DTOs to the services.
- One optional <View-Name>ViewState.java source code document holding the view state information necessary during user interaction.

Within the service invocation methods, the DTOs, having a flow level lifecycle, are passed as parameters to service. These methods are triggered from within the action-state connected to a view.

5.3.4 Model

The models – implemented as Java Bean – at session level holds the information retrieved from database / entered by the user / to be handed over to services / to be displayed to the user via a view.
The DTOs used to pass data between the views and the services are used as models. The implementation follows the J2EE DTO pattern. For details have a look at: http://www.oracle.com/technetwork/java/transferobject-139757.html (included in Annex 10.2).

5.4 User Interface

5.4.1 Client side Part

The client side part of the user interface is represented by the HTML code generated by the view (implemented using JSF). The HTML code is generated according to HTML5 specification. Additionally the code is compliant with WCAG guidelines and the rules for responsive design following the rules defined by the framework bootstrap.

In addition to the functional processes the server side view implementation has to cover:

- Accepting user navigation request via keyboard
- Display timeout intervals and warnings to the user
- Autosave the data input
- Include a map service for displaying geographical data
- Display help information on field level

Detailed implementation guidelines will be added in further releases of this document. This will cover the frameworks to be used and their correct integration (CSS styles to be used, CSS macro frameworks, JavaScript Frameworks, jQuery, JSON, TinyMCE, etc.)

5.4.2 Server side Part

The server side implementation utilizes JSF technology to generate the HTML/JavaScript/CSS source code representing the user interface presentation. The implementation follows the rules of the framework spring. In addition to the functional processes the server side view implementation has to cover:

- Generating multilingual user interface
- Accepting multilingual user input
- Link every data field, process step to the help system

Detailed implementation guidelines will be added in further releases of this document. This will cover additional frameworks/tools to be used and their correct integration (wro4j, sass, less, Primefaces).

5.4.3 Style Guide Implementation

The style guide implementation is referenced by server side and client side components. As there is the following requirement:

- Offer the defined flexibility in user interface configuration available to individual programs

The implementation has to respect the fact that individual programs have the option to adapt the style guide to their needs. This adaption refers to the following options:

- Adaption of a set of predefined CSS-classes (Colours, Fonts, sizes, ...)
- Replacement of a set of images of defined size and resolution

In addition to the classes and images mentioned above the eMS will use fixed classes and images, which may not be altered during the adaption to the needs of individual programs. Therefore those classes that may be modified are located in a separated CSS file containing inline comments. The same goes for the images, which are located in a separated folder within the eMS folder hierarchy.

5.4.4 Overall considerations

Overall considerations to be covered by the user interface implementation are:

- The possibility to open more than one window during an user session
- Offer entry points to user interface to be linked to CMS bases portals of the individual programs
- Handle concurrent data entry in different sessions
- Implementing eMS plug-ins

Detailed implementation guidelines will be added in further releases of this document.

5.5 Business Logic

The implementation of the business logic is done within stateless transactional java bean classes. These get activated by the spring controller to react to a user input. The implementation is split into an interface definition and an implementation class.

Detailed implementation guidelines will be added in further releases of this document covering the spring configuration methods to be used and other coding standards.

5.6 Persistence / Data Storage

5.6.1 Database related storage

Database related storage is accessed utilizing the framework hibernate and its tools. Based on the database model entity classes are generated. These classes are used in the implementation of the data access objects. The implementation of the data access objects (DAO) is done according to the java bean standards and. The DAOs have to act within the transactional context defined by the business services.

Detailed implementation guidelines will be added in further releases of this document covering the methods to be used for defining queries, handling caching tasks and connection pooling and other coding standards.

5.6.2 File system related storage

File system related storage is integrated into the software architecture implementing the correlated data access objects. These classes are implemented manually without the usage of a persistence frame work. This implementation has therefore the responsibility to handle transactional context.

Detailed implementation guidelines will be added in further releases of this document.

5.7 Statistics

Statistics are generated utilizing the following tools:

- jFreeChart
- JasperReports
- Quartz
- Primefaces Tables & Exporting

Detailed implementation guidelines will be added in further releases of this document.

5.8 Reporting

Detailed implementation guidelines will be added in further releases of this document.

- itext, html2pdf
- Quartz

5.9 Messaging

Detailed implementation guidelines will be added in further releases of this document.

- itext, html2pdf
- Quartz

5.10 Logging

Detailed implementation guidelines will be added in further releases of this document.

5.10.1 Error Log

This log has to be done using log4j and must be configurable to reflect different error levels and includes error information.

5.10.2 Information Log

This log has to be done using log4j and must be configurable to reflect different information levels and includes general information. Log points are at least at the transaction boundaries of services and data access objects.

5.10.3 Performance Log

This log has to be done using log4j and must be configurable to reflect different information levels and includes timestamps and execution time of individual processing steps. At least at the transaction boundaries of services and data access objects.

5.10.4 Audit log

This log includes functional activities of individual users. The log entries have to correspond to elements of the user interface. The log has to be written to the database.

5.11 Coding standards

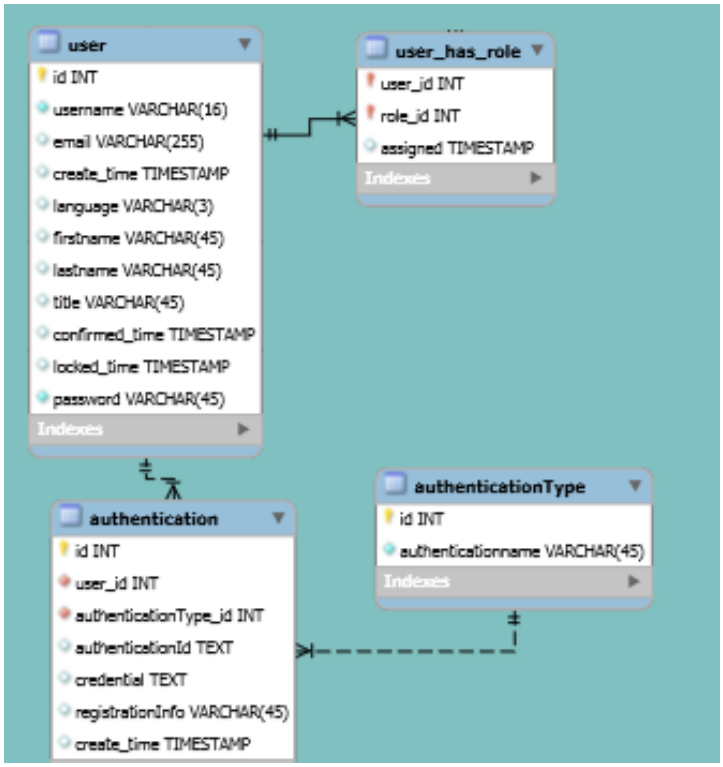
Detailed implementation guidelines will be added in further releases of this document covering all aspects of coding guidelines, error handling etc. A general information on coding guidelines can be found in Annex 10.1.

5.12 Security

5.12.1 Authentication

The implementation of the user authentication uses spring security implementation standard. Therefore eMS contains the EMSAuthenticationProvider.class – an implementation of the spring-security AuthenticationProvider.class. New Authentication Methods in EMS have to be derived from the EMSAuthenticationProvider.class and have to reply with an EMSAuthentication.class Object (derived from Springs Authentication.class). The login-flow handles the user-input and provides the login form. The standard implementation of the flow allows the user to enter user name and password.
When adding additional login provider, the login-flow and form have to be adapted and may reuse the default login form defined for EMSDBUserAuthenticationProvider.class.

User management takes place via the eMS administrative portal, where users are created, removed and their credentials are maintained. Every user is identified by a unique user ID in the EMS database. Every authentication implementation is responsible to map its user identification attributes to the corresponding user ID by maintaining the corresponding database tables within the EMS database (see the picture below). The table authentication hold the information to map an authenticated user to the internal ID.



This robust standard based implementation gives the option for future program specific implementations to use its own authentication methods.

5.12.2 Authorization

Flows are configured to be based on a specific set of user privileges, where a user needs to have at least one, in order to be able to activate a flow.

```
<secured attributes="SHOW_APPLICATIONS,CREATE_APPLICATIONS,SHOW_ALL_APPLICATIONS" match="any"/>
```

Privileges are assigned to user roles. Once the user is authenticated, the logon mechanism created a list of user privileges based on roles, the user is member of. Additional attributes of the role membership are not loaded into the user's session information, but are retrieved from eMS data base as needed and cached in the session information.
A detailed description of the user authorisation requirements for each individual flow based on the specific user privileges is part of the functional specification documentation.

5.12.3 Threats

To reflect a state-of-the-art implementation of a web based application the implementation hast o be according to the owasp guidelines (www.owasp.org) covering to top 10 threats for this type of application.

A1 - Injection:
eMS Uses Hibernate to connect to the database. Hibernate is using owasp.org's guidelines for security (https://www.owasp.org/index.php/Hibernate)

A2 - Broken Authentication and Session Management
No session related information is used in a parameter. Password input fields are using primefaces´s password input component and the session times out in a configured time (with a user information before invalidating the session). In a production environment eMS will be set up on a server that has https configured. eMS uses Spring Security.

A3 - Cross-Site Scripting (XSS)
JSF has always had builtin XSS prevention. eMS uses the outputText tag to display information. This tag automatically escapes scripts.

A4 - Insecure Direct Object References
Direct references to objects (database objects or uploaded binaries) are not used in eMS

A5 - Security Misconfiguration
eMS is developed using state of the art frameworks and components. Those components should be checked and maintained ongoing in a productive scenario.

A6 - Sensitive Data Exposure
The application will use user-friendly error messages. Detailed error information will not be shown to the user.

A7 - Missing Function Level Access Control --> Flow Engine

Spring security handles the user authentication in eMS. Navigation entries and action buttons are only available to users with the corresponding privileges.

A8 - Cross-Site Request Forgery (CSRF) --> JSF 2.0
eMS uses JSF. JSF 2.x has already builtin CSRF prevention.

A9 - Using components with known Vulnerabilities
eMS is developed using state of the art frameworks and components. Those components should be checked and maintained ongoing in a productive scenario.

A10 - Unvalidated Redirects and Forwards
eMS does not use redirects and forwards. After the user registration the activation servlet forwards the user to eMS login. This url points to the server eMS is running on.

5.13 Adaptability

5.13.1 System Parameters

The system parameters determine the functionality of the eMS in a specific instance. Parameters are held within the database. All modifications of a parameter value are reflected immediately by eMS. There is no need to restart the services. Depending on the semantic of a parameter the modification may impact running workflows or only those started after the modification. Details on this are specified in the corresponding functional design documentation. The entry point to these system parameters is configured in the system configuration.

5.13.2 System Configuration

The system configuration covers all technical information (DNS, connection parameters, certificates, etc.) that is needed to execute a specific instance of eMS. One eMS installation consists of a number of workflows available. Every workflow is explicitly enabled/disabled per installation.
The basic system configuration is stored in configuration files. Those configuration files will be kept in a specific folder. The path to this root folder will be configured as start-up parameter in the system's start-up script. The structure inside this root folder is preset and must not be changed by individual programs. Changes of configuration files usually need a restart of the application instance.

Appearance Configuration is done via instance-specific images and CSS folder. The root paths to those folders are also configured in the system configuration files. The CSS classes that are available to modifications by individual programs will be listed in a separate document.

5.14 Quality Management

Guidelines regarding QM processes to comply with in the project can be found in Annex 10.4.

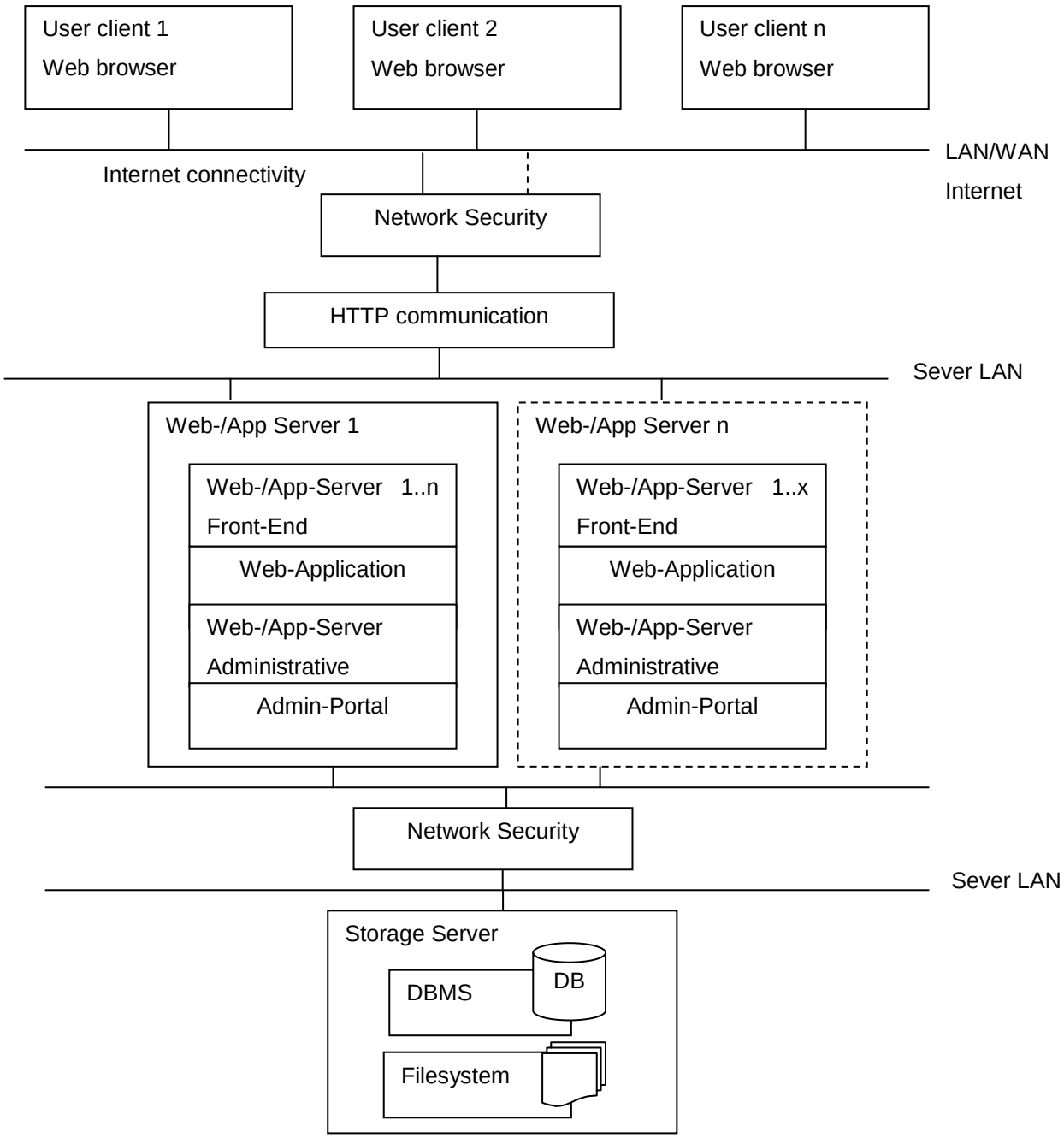
5.15 Maintenance

The system is developed as an Open Source solution, which is made available to other programs. The Solution consists of a reference installation hosted by Interact in Vienna, the source code in a source repository and a pre-configured development environment and ticket system. The processes foreseen in further system development and maintenance can be found in chapter 8.

6 System Architecture

6.1 Server-/Network-Concept

The following picture shows how the individual modules may be distributed to different servers. In this proposal the HTTP communication component is installed on a server located in the DMZ. This communication component uses at least one internet connection. To improve availability and/or performance additional connections may be used. Hidden behind the firewall and additional optional network security elements web server(s) and application server(s) are located in the server LAN. The storage server – hosting the DBMS and the file storage – is installed on a separate server and optionally hidden behind a firewall.



The components of the system architecture have individual tasks to accomplish within the architecture and therefore different characteristics in terms of resource consumption. The following table summarizes the components' characteristics.

System component	Characteristics	Description
User Client	HTML5 compliant web browser	The required resources on the client machine are typically low. Display of big data tables ore heavy use of animation features my increase the amount of resources needed.
HTTP communication	Handling of a great number of http connections and management load balancing	CPU resources are typically very low. When using SSL encryption increase CPU resources needed. To minimize internet bandwidth outgoing data stream may be compresses, which slightly increases CPU resources
WEB-Server	Handling of user sessions and generating user interface	Caused by session management and caching user data within the sessions, the resource bottleneck will be the address space. Therefore multiple instances of the WEB-server may be configured on a single server to use the installed memory.
App-Server	Handling the user requests	Typically processes the user requests within a transaction invoking the services of the storage server. CPU will be the limiting resource.
Storage server	Storing and retrieving data	The storage server provides two different storage services. First a DBMS storing the structured data and file system storage storing unstructured data.
Network security / Server LAN	Hardware and software elements of the networking infrastructure	These elements are not part of the eMS project, but are to be supplied by the operator of the individual program sites.

6.2 Redundancy of services

The core services of the networking concept shown above – http communication, WEB server, Application server – can easily by run in multiple instances. This will improve availability and performance at the same time. This can be done by using network load balancing. For the storage server a full featured cluster installation is needed, when running in multiple instances.

HTTP communication

It is very unlikely that an eMS installation will need HTTP communication clustering. So the proposal is to use one HTTP communication service installed on a single machine.

Web server

The WEB server is consuming a lot of address space. To use the memory installed into the WEB server machine it is an option to have multiple instances of it running on a single WEB server machine. When using network load balancing to spread the workload across the instances, session affinity has to be turned on. This guaranties that once a session is created for a user all subsequent requests are directed to the same server. In case of a node failing the user gets a new session created on another node. The session is not transferred to the new node; therefore the user will receive a session timeout message.

Application server

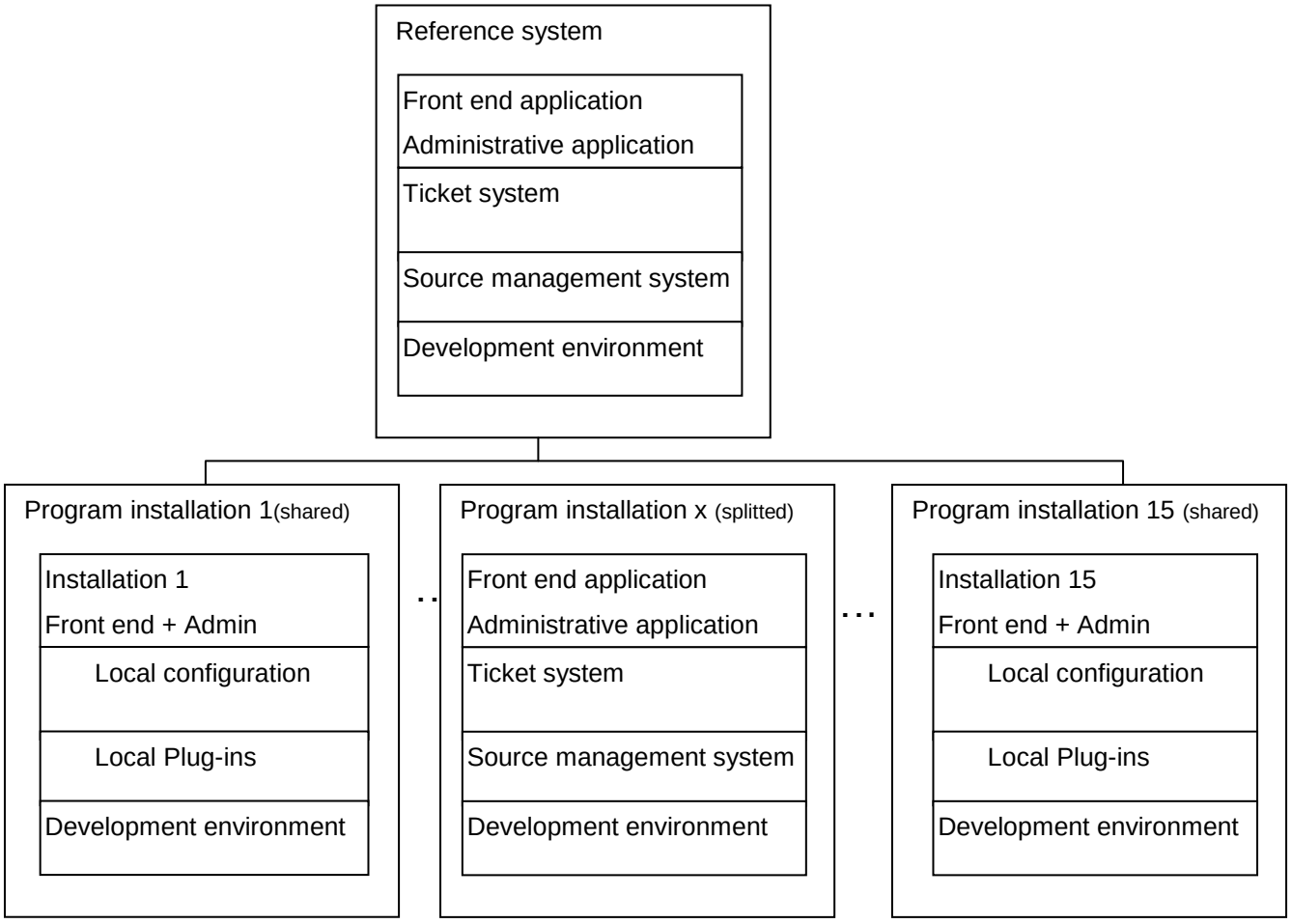
The application server is consuming a lot of CPU resources depending on the mixture of user requests. The requests are processes as stateless transactions. Network load balancing can be used to spread the requests across multiple instances increasing the availability of the service. When using network load balancing session affinity must not be turned on. No session must be created for a processing request.

Storage server

The easiest way to increase availability and performance of the storage system is to use an according hardware having redundant components installed. From the point of view of the software, there is only the option to use a full featured cluster solution, beginning from the operating system level to the DBMS.

6.3 Multiple Installations

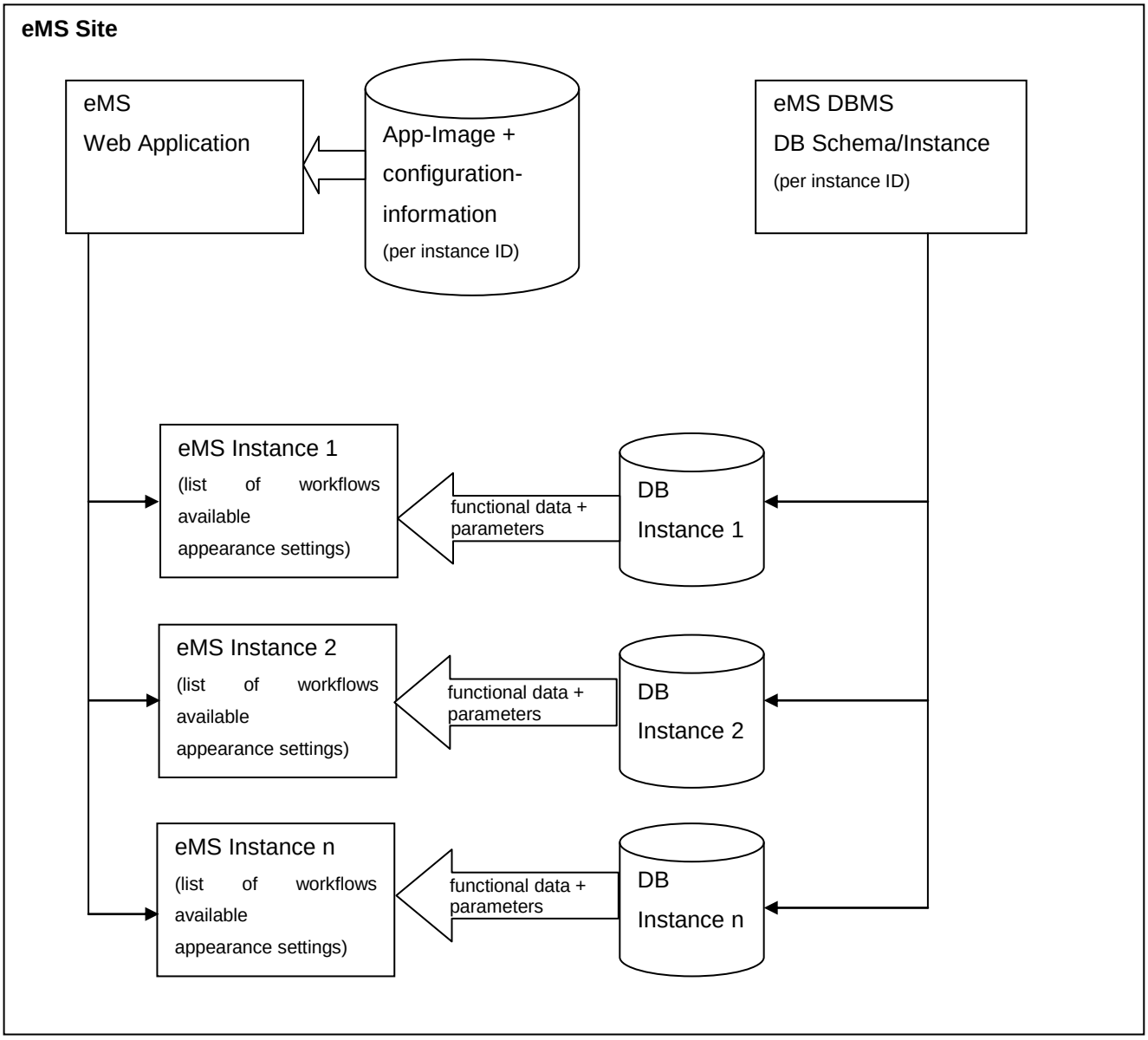
eMS is run in multiple instances; one for each individual program. Each program can use its own configuration and develop individual source code:



Programs by decide to take part on a common development process for further releases of eMS sharing their development or to split and have their own system based on a starting release. When sharing development work the program installations use their own development environment, but still stay connected to the central source managing system and the ticket system. In case a program decides to split and develop its own future releases it runs its own instances of source management system and ticket system.

6.4 Multiple Instances

One eMS site can host several eMS instances. This can be achieved by following the concept shown in chapter 5.13:



6.5 Software Requirements

The implementation of eMS uses the cross platform technology Java. Java is available on a number of platforms including commercial and open source platforms. Additionally Java offers a lot of tools – built into Java's core modules or on open source basis – that guarantees a very efficient and cost effective way to implement web based applications. The following table list the products used for eMS implementation:

System Component	**Product/Remarks**
Operating System	Linux
HTTP-communication	Open source product Apache HTTP-Server
WEB server	Open source product Apache Tomcat
Application server	None, Application container supplied with Tomcat
Web applications	State-of-the-art Web applications following the MVC-Paradigm using well known frameworks and technologies (Java 1.7, Spring, JSF, …)
DBMS	Open source product MySQL
Source management	SVN
Ticket system	Redmine
Development env.	Eclipse, for additional frameworks and tools see chapter 5

7 Justification of architecture

This chapter will be added in a future release of the document.

8 Software Maintenance and Development

This chapter will be added in a future release of the document.

9 Requirements traceability

This chapter contain a list of all requirements relevant for the software architecture. Please keep in mind, that these requirements may be relevant for the functional design as well. This is not covered by this document.

As these requirements are documented in the documents of the tender, they are not translated from German to English.

9.1 MUSS und SOLL-Kriterien

1	**Kern-System** **Der Kern des Systems umfasst die Basisfunktionalitäten der Applikation und muss als erstes ausgeliefert werden.**	Relevant	Chapter
1.1	Flexibilität und Anpassbarkeit: Das System muss erlauben einzelne Schritte im Arbeitsablauf auszulassen, falls diese nicht benötigt werden (einfaches Ein-/Abschalten von einzelnen klar definierten Teilschritten im workflow)	A	3.2, 3.5.6
1.2	Es muss möglich sein, Texte im user interface leicht zu ändern.	A	3.3, 3.5.6
1.4	Einzelne – klar identifizierte – Tabellen müssen sortierbar sein	A	3.3
1.5	Das System muss bis zu 20 verschiedene Projekttypen verwalten können. Diese müssen von Systemadministratoren definiert werden können hinsichtlich: A) einzelner Eingabefelder: sichtbar/unsichtbar und lesend/schreibend – zB Antragsformular , Projektpartnerfortschrittsbericht, Projektfortschrittsbericht, Checklisten, B) Berichte/Reports	A	3.5.1, 3.5.6
1.9	Das System muss erlauben, Muster/Templates von Berichten vom Systemadministrator vorzudefinieren und zu konfigurieren. Das umfasst auch die Sortierung bestimmter Inhalte.	A	3.5.2, 3.5.6
1.10	Automatische Generierung von Statistiken und Berichten: Das System muss die automatische zeitlich gesteuerte Generierung (vorzugsweise in der Nacht) von vordefinierten statistischen und inhaltlichen Berichten unterstützen und diese im System gespeichert zur Verfügung stellen.	A	3.5.2, 3.5.3
1.11	Das System muss die Möglichkeit bieten, benutzerdefinierte statistische Auswertungen und Berichte zu jeder Zeit zu erstellen.	A	3.7, 3.8
1.12	Es muss möglich sein, die vom System erzeugten Berichte manuell nachzubearbeiten	A	3.5.2
1.13	Generierte bzw. manuell nachbearbeitete Berichte müssen exportierbar sein in: 1) editierbares Format und 2) nicht-editierbares Format (zB .pdf)	A	3.8
1.14	Berichte müssen einfache Diagramme und Grafiken enthalten können, welche auf Basis der ausgewerteten Daten erzeugt werden.	A	3.5.2

1.15	Es muss ein gruppen- und rollenbasiertes Rechte-Verwaltungsystem zur Verfügung stehen. Dementsprechend muss es möglich sein, einzelne Felder oder Files für ausgewählte User sichtbar bzw. editierbar zu machen.	A	3.5.4, 3.5.6
1.17	**Mehrsprachigkeit:** es muss möglich sein, das System in allen offiziellen Sprachen der EU zu verwenden. Sowohl das User Interface als auch die Textfelder und Berichte müssen alle Alphabete und Zeichensätze, die in der EU gebräuchlich sind, unterstützen.	A	3.3, 3.5.6
1.18	Mehrsprachiger Inhalt („Multi-lingual content“): Das System muss die Eingabe von Text in bis zu drei Sprachen unterstützen. Hierzu müssen bis zu 3 Textfelder für bis zu drei Sprachen vorgesehen werden. Diese Funktionalität betrifft ausschließlich Textfelder, in denen Freitext verfasst werden kann. Numerische Felder oder Felder mit Datumseingabe sowie spezielle Eingabefelder wie zum Beispiel Email-Adressen sind davon nicht betroffen.	A	3.3, 3.5.6
1.19	Es muss auch möglich sein, Berichte in bis zu 3 Sprachen zu erstellen und zu speichern.	A	3.5.6, 3.8
1.20	Es muss möglich sein, die Anzahl der Eingabesprachen pro Installation auf ein, zwei oder drei festzulegen. Die dadurch eventuell nicht benötigten Schaltflächen müssen dann nicht sichtbar sein.	A	3.3, 3.5.6
1.35	Die Konsistenz und Integrität der Daten muss sichergestellt sein.	A	3.4
1.37	Die automatische periodische Sicherung der ausgefüllten Felder soll sichergestellt sein.	A	3.3
1.38	Die Session muss nach einem definierbaren Zeitraum auslaufen (Session time out von z.B. 30 Minuten ohne Aktivität).	A	3.3
1.39	Der Zeitraum bis zum Session-Timeout muss für den Benutzer des Systems ersichtlich sein.	A	3.3
1.40	Vor dem Session Timeout muss eine Warnung erscheinen: z.B.: Warnung: Ihre Sitzung wird bald ablaufen, bitte vergessen Sie nicht zu speichern!	A	3.3
1.52	Dokumentenablage: Eine zentrale Dokumentenablage muss verfügbar sein, in der alle Dokumente, die im System hochgeladen wurden, zur Verfügung gestellt werden. Es muss möglich sein, diese Dokumente in einer vordefinierten Struktur abzulegen (z.B. pro Projekt, Pro Dokumenttyp).	A	3.4
1.61	Es soll eine Ablage der Dokumente auf Filesystemebene erfolgen.	A	3.4
1.62	Der Zugang zu bestimmten Bereichen der Dokumentenablage oder zu Dokumenten unterschiedlicher Art, muss durch Userrechte beschränkbar sein.	A	3.4
1.64	Sicherheit und Vertraulichkeit der übermittelten Daten muss gegeben sein.	A	3.1
1.65	Sicherer Login mit Username und Passwort muss unterstützt werden. Mindestsicherheit für Username/Passwort muss überprüft werden.	A	3.11
1.66	OPTION: Das System muss eine Schnittstelle zur nachträglichen Implementierung sicherer Loginmechanismen wie e-signatur bieten. Darüber hinaus muss es möglich sein mehrere verschiedene Loginmechanismen wie e-signatur für verschiedene an einem ETZ Programm teilnehmende Länder zu ermöglichen. Die Implementierung muss möglich sein, ohne die Gewährleistung des Auftragnehmers für das restliche System einzuschränken oder zu verlieren.	A	3.11

1.71	Benutzer sollen die Möglichkeit haben, Shortcuts zu definieren, um die von Ihnen oft benutzten Funktionen schnell erreichen zu können.	A	3.3
1.72	Automatischer zeitgesteuerter Export ausgewählter Datensätze & automatischer anschließender Transfer (Vorschlag: FTP)	A	3.10
1.73	OPTION: Geografische Präsentation von Projektdaten – hier wird die geografische Präsentation von Projekten/ihren Projektpartnern in einer interaktiven Karte gezeigt. Geografische Koordinaten müssen für jedes Projekt/jeden Projektpartner bzw. Projektziele/-ergebnisse (z.B. Investitionen) gespeichert werden, um diese auf einer Karte anzeigen zu können.	A	3.3
1.74	Komfortfunktionen bei Mouse-Over: Das System soll Informationen über ein Feld (z.B. wie Berechnung durchgeführt wurde, was eingegeben werden soll, etc) anzeigen. Der Inhalt dieser Informationsboxen muss von ETZ Programmen konfiguriert werden können.	A	3.5.5
2	**Kommunikationportal**	**Relevant**	
2.1	Das Kommunikationsportal muss den sicheren Austausch von Informationen zwischen den betroffenen Stellen ermöglichen (z.B. Austausch zwischen Begünstigten und relevanten Behörden oder zwischen verschiedenen behördlichen Stellen).	A	3.6
2.2	Das System muss den Austausch von Text Nachrichten zwischen Teilnehmern einer Betriebsinstanz des Systems unterstützen.	A	3.6
2.7	Versendung automatischer durch Events ausgelöster Nachrichten muss möglich sein.	A	3.6
2.9	Die Nachrichten müssen Projekten zugeordnet werden können.	A	3.6
2.15	Der zeitgesteuerte und durch Ereignisse (event triggered) ausgelöster Versand von Nachrichten soll möglich sein.	A	3.6
3	**Antragsformularmodul** **Dieses Modul umfasst die Eingabe von Informationen durch potenzielle Projektträger im Rahmen des Einbringens eines Projektantrags bzw. eines Förderungsansuchens. Daten aus dem Antragsformular werden in verschiedenen anderen Modulen wiederverwendet.**	**Relevant**	
3.7	Das Nutzerprofil muss über einen Link, der über Email versendet wird aktiviert werden.	A	3.5.4
3.11	Falls sub-user Schreibrechte haben, muss ein gleichzeitiges Editieren durch mehrere Nutzer unmöglich sein um Datenverlust zu vermeiden. Eine Warnmeldung muss bei gleichzeitiger Bearbeitung angezeigt werden.	A	3.3
3.17	Die Anzahl der Anhänge und ihr Charakter (Verpflichtend ja/nein) muss konfigurierbar sein.	A	3.5.1, 3.5.6
3.18	Dier maximale Anzahl/Dateigröße muss konfigurierbar sein, um den für unstrukturierte Dateien vorgesehen Speicherplatz zu limitieren.	A	3.4, 3.5.6
3.26	Die eingereichten Antragsformulare müssen als .pdf exportierbar und druckbar sein.	A	3.5.1

3.29	Das System kann bei Einreichung des Antragsformulars eine Checksumme einer Version des Antragsformulars berechnen und speichern. Das System muss aber gewährleisten, dass die Checksumme nur bei Änderungen geändert wird, nicht wenn das Antragsformular z.B. angesehen wird.	A	3.5.1
4	**Bewertungs- und Auswahlmodul** **Dieses Modul unterstützt die Bewertung von Projektanträgen mit Hilfe von Standard Bewertungskriterien.**	**Relevant**	
4.20	Login und Passwort der Antragssteller müssen nach der Projektbeurteilung inaktiviert werden. Lead Partner erhalten ein neues Login und Passwort. Auch hier muss es dem Lead Partner möglich sein sub-user Rechte zu vergeben.	A	3.5.4
7	**Projektpartner-Fortschrittsbericht/ FLC Modul** **Dieses Modul ermöglicht den Projektpartnern die Eingabe von periodisch notwendigen Projektfortschrittsberichten und Zahlungsanträgen und die Übermittlung derselben an die Ausgabenkontrolle (FLC).**	**Relevant**	
7.15	Prüflisten und Prüfberichte müssen von Administratoren verwaltet und verändert werden können.	A	3.5.1, 3.5.6
10	**Programmfortschrittsmodul** **Dieses Modul unterstützt die Fortschrittsüberwachung verschiedener Aspekte der ETZ Programme basierend auf Daten, die auf Projektebene gesammelt werden.**	**Relevant**	
10.14	Es muss möglich sein Projektfortschrittsberichte automatische zu erstellen basierend auf Konfigurationen. Der Bericht muss editierbar sein und exportierbar sein. Der Bericht inkludiert auch Statistiken und Aggregationen von Projektdaten.	A	3.5.6, 3.8
11	**Unregelmäßigkeiten- und Rückforderungsmodul** **Dieses Modul unterstützt das Überwachen und Senden von Berichten über Unregelmäßigkeiten, Betrügereien und daraus folgenden Rückforderungen. Es beinhaltet auch Berichte über Rückforderungen, rückgeforderte Beträge und uneinbringbare Beträge.**	**Relevant**	
11.5	Es muss möglich sein einen editierbaren, sortierbaren und exportierbaren Report basierend auf konfigurierbaren Kriterien (Zeit, Projekt, etc.) zu generieren.	A	3.5.6, 3.8
13	**Audit Modul (SLC)** **Hier wird der Prüfbehörde (Audit Authority, AA) die Möglichkeit geboten, die für ihre Arbeit notwendige Informationen zu sammeln, das Resultat des Audits zu kommunizieren und bei Bedarf den Informationsaustausch mit anderen Behörden der ETZ Programme und Projektträgern zu gewährleisten, sowie Informationen über gefundene Unregelmäßigkeiten zu speichern.**	**Relevant**	
13.3	Das System muss an dieser Stelle die automatische Erzeugung eines vordefinierten Berichtes unterstützen.	A	3.5.6, 3.8

9.2 Contract

Absatz Kapitel	**Vertrag** **Architektur-relevante Anforderung aus dem Vertrag (3692 02077_e-MS_ETZ_Vertrag_AU2.pdf).**	Relevant	Chapter
V22	Die Software muss modular aufgebaut sein und sicherstellen, dass die ETZ Programme die formalen juristischen Anforderungen Schritt für Schritt erfüllen können. Außerdem soll den ETZ Programmen so die Flexibilität geboten werden einige Module zu verwenden und andere nicht. Die Rechte am Source Code müssen so gestaltet sein, dass es den verschiedenen ETZ Programmen freigestellt ist, Änderungen am Code selbst durchführen zu können. Details dazu sind diesem Dokument zu entnehmen.	A	3.9, 3.12
V25.1	die Anforderungen entstehend aus CPR Art. 122 (3) und 125 (d) (Regulation (EU) 1303/2013) und möglichen anderen Regulatorien an die Übertragung und Speicherung der Daten erfüllen; im Besonderen sind die folgenden Regulatorien (aktuell noch im Entwurf Status) zu berücksichtigen	A	3.9, 3.12
V25.2	Flexibilität und Konfigurationsfähigkeit aufweisen, um ETZ Programmen die Möglichkeit zu geben, die Software nach ihren Bedürfnissen anpassen zu können. Als Beispiel kann hier flexible Workflowgestaltung genannt werden, sowie Konfigurationen, bspw. die Beschriftung verschiedener Felder, die optionale Nutzung bestimmter Felder, die Nutzung verschiedener Sprachen, verschiedene Währungen, etc.	A	3.9, 3.12
V25.3	eine ausgefeilte User-Verwaltung aufweisen, mit der Möglichkeit zur Vergabe einzelner Userrechte, sogar auf der Ebene einzelner Input-Felder	A	3.5.4
V25.4	sichere Authentifizierung mit Username und Passwort ermöglichen. Es sollte ETZ Programmen möglich sein, zusätzlich fortgeschrittenere Sicherheitsmechanismen wie z.B. elektronische Signatur mit Bürgerkarte ohne tiefe Eingriffe in das System zu implementieren	A	3.11
V27	Da zum derzeitigen Zeitpunkt noch nicht alle Anforderungen an das System fixiert werden können (z.B. aufgrund einer neuen Version der EU Rechtsgrundlage) muss die Möglichkeit bestehen, zusätzliche Felder zu ergänzen. Den das Monitoring System nutzenden ETZ Programmen soll es möglich sein, zusätzliche	A	3.9

	Felder zu den Eingabeformularen auf einfache Weise hinzuzufügen, möglicherweise ohne zu programmieren		
V28	ETZ Programme arbeiten in verschiedenen Sprachen. Daher muss die Möglichkeit zur Übersetzung des User Interfaces in alle europäische Sprachen bestehen. Die Software muss für die Verwendung in allen europäischen Sprachen, Alphabeten und Zeichen geeignet sein. Auch die Generierung möglicher Berichte muss in jeder dieser Sprachen und mit allen erwähnten Zeichen möglich sein	A	3.3
V29	Manche ETZ Programme benötigen den Inhalt der Eingabeformulare in mehreren Sprachen (bis zu drei Sprachen) („Multilanguage Content“). Darum muss es möglich sein, jedes Textfeld in den Eingabeformularen mehrfach auszufüllen – jedes Mal in einer anderen Sprache. Es muss auch möglich sein, Berichte in mehreren Sprachen zu erstellen. Es muss jedoch auch die Möglichkeit geben, diese Funktionalität auf eine Sprache zu limitieren.	A	3.3
V31	Der Auftraggeber wird ein zentrales Test und Referenzsystem in Wien betreiben.	A	3.5.6
V34	Insgesamt ist von Datenmengen im Umfang von ca. 17 TB pro Instanz für den gesamten Software Lebenszyklus auszugehen	A	3.4
VC12.2	Anpassungen des Systems durch den Auftraggeber	A	3.9

9.3 Price Sheet

V	Preisblatt Architektur-relevante Anforderung aus dem Preisblatt	Relevant	Chapter
PB1	50 Concurrent User	A	3.9
PB2	100 Concurrent User	A	3.3

9.4 Solution Concept

Kapitel	Preisblatt Architektur-relevante Konzepte aus dem Lösungskonzept	Relevant	Chapter
K1.3	Responsive Design	A	3.3
K3.5	Anpassung GUI-Design (CSS, Logos,…)	A	3.3

9.5 Initial Kick-Off

Protokoll	Preisblatt Architektur-relevante Konzepte aus dem Projekt-Kickoff vom 17.3.2014	Relevant	Chapter
A17.03.01	Mehrere Fenster Öffnenbar	A	3.3
A17.03.02	300-500 concurrent User	A	3.3
A17.03.03	Lieferung in Englisch+Deutsch	A	3.5.6
A17.03.04	Entry-Point auf zumindest Application-Form muss gegeben sein	A	3.3
A21.03.01	Interact hat die Anforderungen an die Anpassbarkeit und Flexibilität genauer definiert – folgendes muß flexibel sein: - Userrechte auf Feldebene - Workflow bei den Payment Claims - Zusätzliche Textfelder zu den Anträgen - Optionale Blöcke ein/ausschalten - Error Checks und Validation Rules - Dashboard und Reporting - Berichte (Dokumentenvorlagen)	A	3.5.1
A27.03.01	Generischer Ansatz nur für zusätzliche Felder notwendig. Attribute der Application Form, Budgets & Co. werden im Datenmodell modelliert.	A	3.5.1

10 Annex

10.1 Code Conventions

CodeConventions.zip

10.2 DTO Pattern

Core J2EE Patterns - Transfer Object.zip

10.3 OWASP Top 10 - 2013

OWASP Top 10 - 2013.zip

10.4 QM-Guidelines

These documents are part of the PL.O.T QM-System. Project development has to comply with these guidelines as far as there are no contradicting confirmations with the customer.

QM-Guidelines.zip