Project Identification	
Project number	AAL-2013-6-091
Duration	1st June 2014 – 30th November 2016
Coordinator	Univ. Prof. Dr. Manfred Tscheligi
Coordinator Organization	AIT Austrian Institute of Technology GmbH, Austria
Website	www.pearl-project.eu

pearl **P**latform for **E**rgonomic and motivating, ICT-based **A**ge-friendly wo**R**kp**L**aces

Document Identification	
Deliverable ID:	D-4.4.1 Decision Support System and Rule Engines
Release number/date	V1.0 02.12.2015
Checked and released by	Anton Katov/AAU

Key Information from "Description of Work"	
Deliverable Description	This deliverable describes the first prototype implementation of the programmable interfaces and configuration management tools.
Dissemination Level	PU = Public
Deliverable Type	P = Prototype, R = Report
Original due date	Project Month 18 / 01.December.2015

Authorship& Reviewer Information	
Editor	Anton Katov/AAU
Partners contributing	AAU, SiLO, SENSAP, AIT, RRD
Reviewed by	Kostas Perakis/SiLo

Abbreviations

Abbrev.	*Description*
BIL	Business Intelligence Layer
CB	Case Base
CBR	Case-Based Reasoning
CSV	Comma-Separated Values
DSS	Decision Support System
HATEOS	Hypermedia as the Engine of Application State
IDE	Integrated Development Environment
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
MVC	Model-View-Controller
REST	Representational State Transfer
UI	User Interface

Table of Contents

Executive Summary

PEARL Decision Support System (DSS) module is one of the major components constituting the PEARL Business Intelligence Layer (BIL). The main purpose of the DSS module is to provide a customized configuration plan to every new platform user during the registration process, thus facilitating the initial platform deployment and improving the user experience. D4.4.1 Decision Support System and Rule Engines aims to describe the outcome of T4.5 Decision Support System and Workflow Engines in terms of implementation logic and underlying technologies of the first PEARL Decision Support System prototype and to outline the directions of future work.

The PEARL DSS module undertakes the task of mapping a set of user profile, workspace and task characteristics to a set of platform configuration settings and adjusting the necessary configuration parameters, based on the specific characteristics of each user. The operation of the PEARL DSS is based on the Case-Based Reasoning (CBR) paradigm in combination with a Rule-based engine for case adaptation. PEARL DSS module consists of two major submodules – the CBR-based Matchmaker and the case adaptation Rule Engine. The Matchmaker is responsible for comparing the profile characteristics to the case base of existing user(s), calculating the best match and retrieving the corresponding configuration settings from the PEARL database. The Rule engine then undertakes the retrieved configurations and adjusts them, based on the profile differences between the new user and the retrieved user.

PEARL DSS has been implemented as a RESTful web service, based on the Spring MVC Framework. The web service is initially invoked by an HTTP request containing the new user ID sent by a client application and returns the customized configuration plan in JSON format.

1 About this Document

1.1 Role of the deliverable

The role of D4.4.1 Decision Support System and Rule Engines is to provide an overview of the principle of operation, the underlying technologies and the implementation logic of the PEARL DSS module. Furthermore, the deliverable summarizes the parameters of the utilized data models, their available values and the process of CBR-project customization and similarity measures definition. The document will be used as a functional overview of the DSS module and will provide guidelines for the development and implementation of additional module features in the future.

1.2 Relationship to other PEARL deliverables

The deliverable is related to the following PEARL deliverables:

Deliv:	*Relation*
D4.1	System Architecture Specification and Implementation
D4.2	User, Tasks and Workspace Databases, Ontologies and Knowledge Bases

2 Requirements and System Overview

The current chapter outlines the requirements and the main functionalities of the DSS module, presents an overview of the main principles of operation and provides a short description of the underlying components and their interactions with the relevant PEARL platform modules.

2.1 Requirements of PEARL Project

The DSS is an integral component of PEARL Business Intelligence Layer and undertakes the tasks of mapping the user profile, workspace and task characteristics to a set of configurations parameters, adjusted for the needs of the particular user. The module will be launched during the initial user registration phase and aims to simplify the initial setup process and to improve the interaction with the platform by providing a customized configuration plan for each new user. In order to provide maximum flexibility during the initial platform configuration phase a semi-automated operation model will be utilized, where the platform will suggest a set of customized configuration settings and the user will be allowed to either accept them or further modify them, based on his/her preferences.
The DSS module implementation, which will be described in the following sections, is based on a Case-Based Reasoning (CBR) engine that incorporates a rule-based system module for case adaptation.

2.2 Architecture Concept and Principles of Operation

Providing a set of customized configuration parameters, based on a predefined list of user characteristics can be classified as a complex highly unstructured problem due to the large number of possible input and output parameters, the lack of formalized rules defining their interdependence and the ambiguity of the possible solutions. In order to tackle the lack of explicit domain model and to provide a flexible solution that will allow adding new configuration features in the future we have chosen to utilize a CBR-based approach in combination with rule-based system module for case adaptation. The characteristics and benefits of CBR and rule-based systems will be discussed in the following chapter. The principle of operation of the proposed DSS is illustrated on the following diagram:

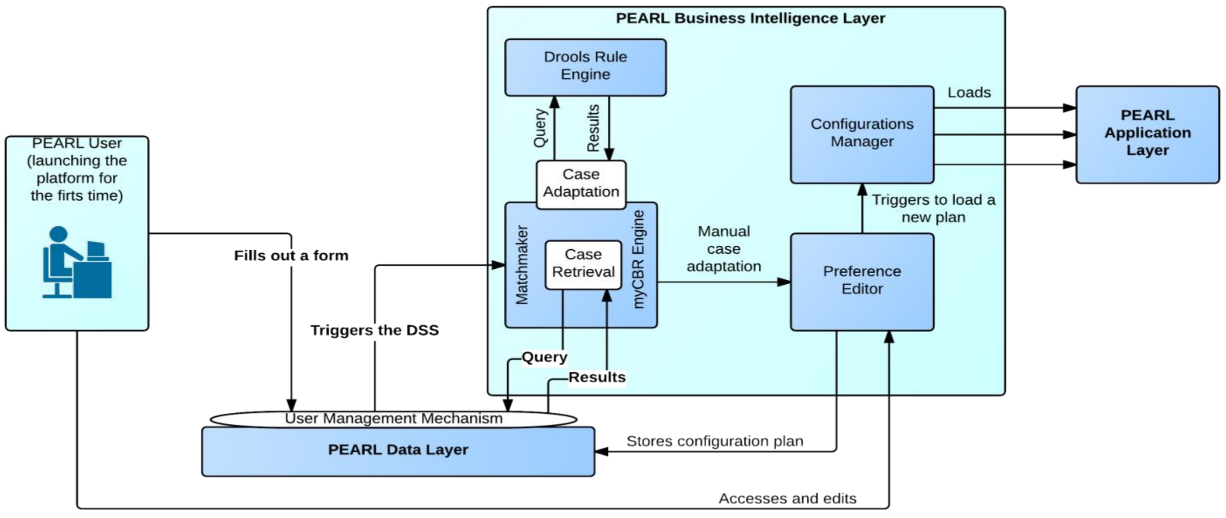


Figure 1. PEARL DSS Architecture

PEARL DSS module is a fundamental part of the Business Intelligence Layer that will constitute the core intelligence of the PEARL system, as defined in *D4.1 System Architecture Specification and Implementation*. The module is utilized during the process of new user registration and returns a customized set of configuration parameters, based on the personal characteristics submitted by the user. It consists of two main submodules – the matchmaker and the rule engine. The matchmaker is powered by myCBR and is responsible for matching a set of user preferences to a similar existing profile and retrieving the relevant configuration plan. In the process of selection, a set of similarity functions provided by myCBR engine are employed. For further details in the similarity measures please refer to *Chapter 4 – Modelling of Users and Tasks in the Workspace.* The PEARL Rule Engine is powered by Drools and incorporates the case adaptation logic required whenever a retrieved solution does not satisfy fully the initial user requirements and needs to be altered.
The PEARL DSS module interacts directly with two main components - the PEARL Data Layer, which host all relevant user profile information and configuration settings, and the Preference Editor, which offers an adapted UI for manual modification of the configuration settings, and implements the required datasource for data retrieval and storage. For the needs of configurations retrieval, a RESTful web service has been developed that takes as an input the ID of the newly created user and returns the adapted configuration settings in JSON format. In order to better understand the principle of operation of the DSS and the integration logic within the PEARL platform, the data flow during the registration process is presented in figure 2.

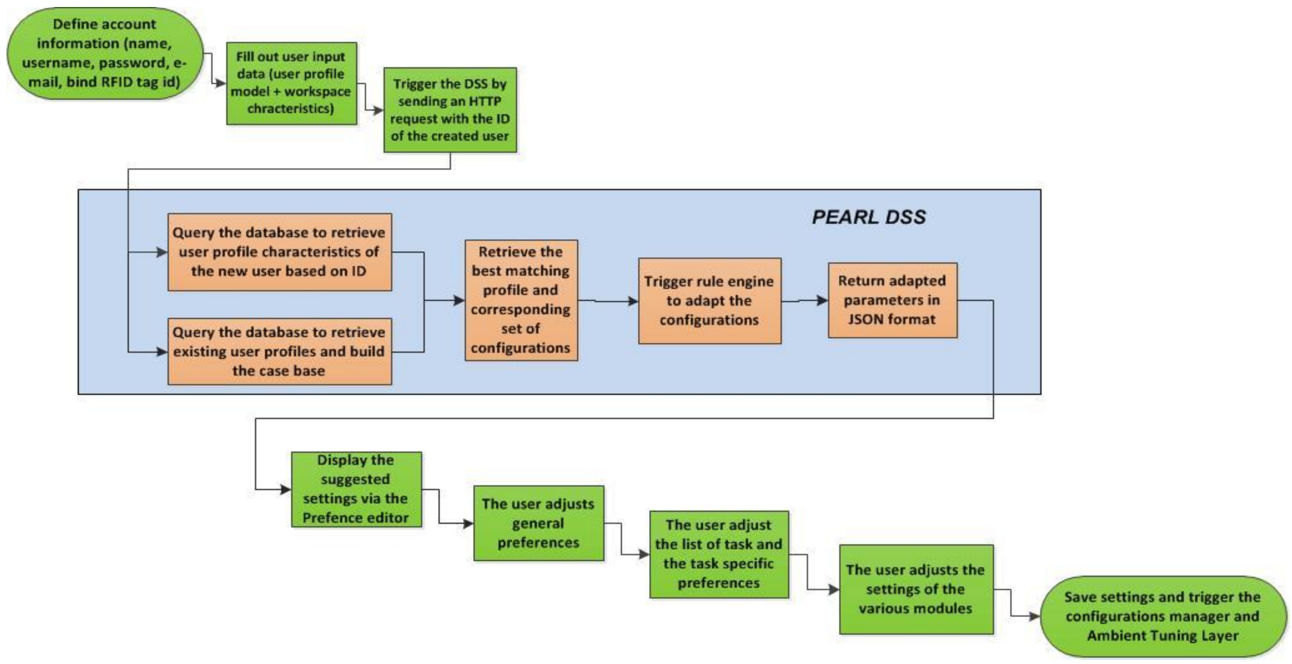

Define account information (name, username, password, e-mail, bind RFID tag id)
Fill out user input data (user profile model + workspace chracteristics)
Trigger the DSS by sending an HTTP request with the ID of the created user
PEARL DSS
Query the database to retrieve user profile characteristics of the new user based on ID
Query the database to retrieve existing user profiles and build the case base
Retrieve the best matching profile and corresponding set of configurations
Trigger rule engine to adapt the configurations
Return adapted parameters in JSON format
Display the suggested settings via the Prefence editor
The user adjusts general preferences
The user adjust the list of task and the task specific preferences
The user adjusts the settings of the various modules
Save settings and trigger the configurations manager and Ambient Tuning Layer

Figure 2. New User Registration Data Flow

The designated steps can be further described as follows:

1. **Initial data input** – the new user is prompted to fill out a number of forms defining his account credentials and a set of personal characteristics.
2. **DSS trigger** – the input data are stored in the PEARL database and the DSS is triggered by sending an HTTP GET request with a path variable containing the ID of the newly created user.

3. **Problem retrieval** – the user profile characteristics of the newly created user are retrieved by the DSS by executing a database query based on the user ID. The resulting data set constitutes the input problem definition for the CBR engine.
4. **Case base construction**– the CBR case base is constructed by retrieving from the PEARL database all existing user profiles that have a full set of configuration settings already defined. An additional parameter, *"setConfigurations",* that indicates the validity of a given database entry is introduced.
5. **Case retrieval** – the best matching user profile is selected from the case base, based on predefined similarity functions, provided by myCBR engine. The corresponding configuration settings are then retrieved from the database.
6. **Case adaptation** – the retrieved configurations data set is then adapted by the rule engine based on predefined rules.
7. **Adapted solution** – the adapted configurations are then passed to the Preference editor in JSON format and displayed to the user.
8. **Solution revision** – the user is then prompted to either accept or modify the suggested configurations via the Preference Editor UI.
9. **Case retention** – when the configurations are adapted/accepted by the user, the resulting set of parameters is stored to the database and the *setConfigurations* variable is changed, thus indicating that the new case can be included in the case base in the future.
10. **Configurations deployment** – the configurations manager and the ambient tuning layer trigger manager are activated in order to deploy the new configurations.

The data flow up until the point of adapted solution retrieval is presented in the sequence diagram on figure 3.

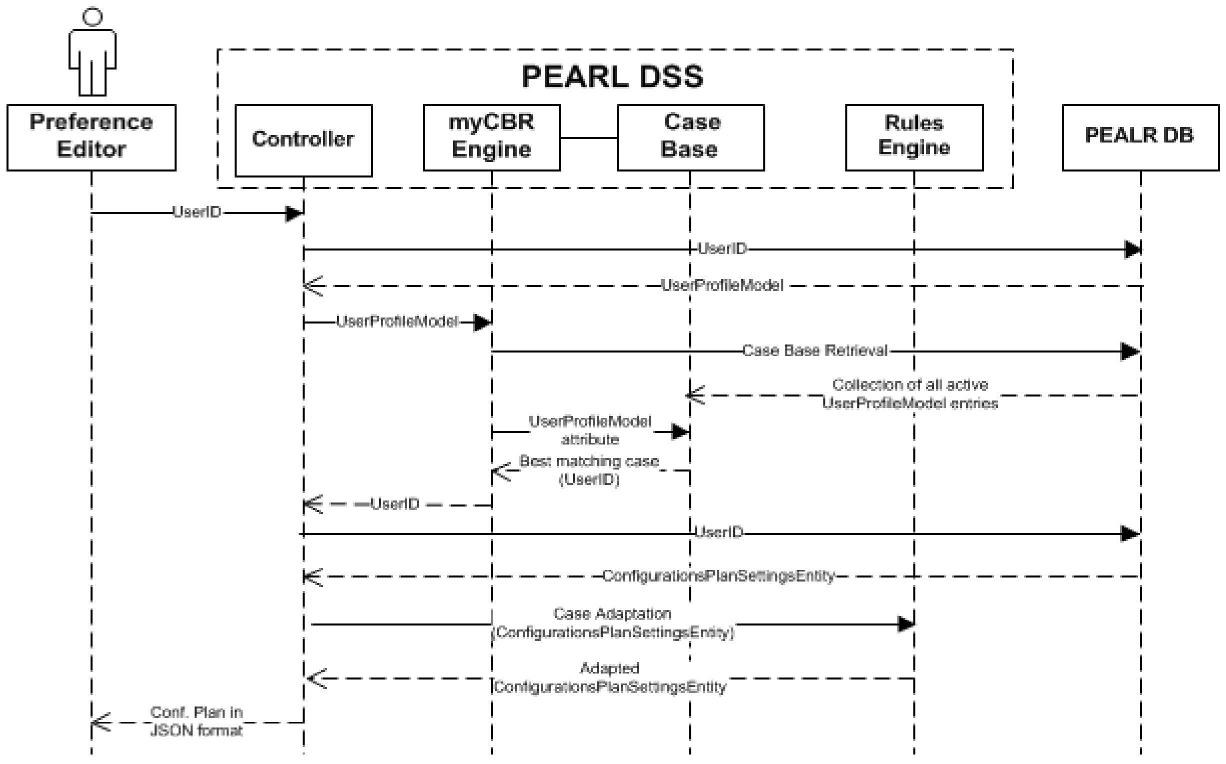

PEARL DSS
Preference Editor
Controller
myCBR Engine
Case Base
Rules Engine
PEALR DB
UserID
UserID
UserProfileModel
UserProfileModel
Case Base Retrieval
Collection of all active UserProfileModel entries
UserProfileModel attribute
Best matching case (UserID)
UserID
UserID
ConfigurationsPlanSettingsEntity
Case Adaptation (ConfigurationsPlanSettingsEntity)
Adapted ConfigurationsPlanSettingsEntity
Conf. Plan in JSON format

Figure 3. DSS-based configurations retrieval

3 Underlying Technologies and Tools

The current chapter aims to further elaborate on the description of the artificial intelligence techniques – case-based reasoning and rule-based expert systems - that lay the foundation of the PEARL DSS logic. In order to take advantage of the state-of-the-art developments in the aforementioned fields for the needs of the PEARL project, we have integrated myCBR open-source similarity-based retrieval tool for the needs of the DSS matchmaker and will further integrate Drools Rule Engine for the process of rule-based case adaptation. The main characteristics and functionalities of myCBR and Drools will also be outlined in the following sections.

3.1 Case-Based Reasoning and myCBR Engine

The PEARL DSS matchmaker submodule's main function is to select an existing user profile that closely matches the characteristics of the new user and to retrieve its corresponding configuration settings. We have chosen the CBR techniques as an appropriate solution because of its ability to operate in situations when the domain might have undefined parameters, a weak or unknown causal model or when its formalization requires a huge amount of rules.

3.1.1 CBR fundamentals

In [1] CBR is defined as reasoning by remembering, i.e. extracting and adapting already existing past solutions of a problem in the process of solving new problems. The concept of CBR is based on four general assumptions about the surrounding world, defined in [2]:

- **Regularity** – the same actions under the same conditions produce equal or similar results
- **Typicality** – experiences tend to repeat
- **Consistency** – small changes in the required solutions require small changes in the input parameters
- **Adaptability** – when events repeat, the differences tend to be small

The fundamental component of every CBR system is the case base (CB), which stores the collections of all previously known or experienced problem situations in the form of cases. Each case defines contextually a known past experience in terms of a problem definition and the respective solution or the expected outcome [1]. In the context of PEARL we have defined the case to be represented by a problem-solution pair that consists of the following main entities, which have been extensively defined in *D4.2 User, Tasks and Workspaces Databases, Ontologies and* Knowledge Bases and will be summarized in Chapter 4 – Modeling of Users and Tasks in the Workspace of the current document:

- **User Profile Model** – contains a set of demographic, cognitive, physical and professional characteristics of the user. The user profile model attributes constitute the input parameters to the CBR matchmaker, or the problem.

- **Configuration Plan Model** – defines a list of available tasks (email, text-editing, meetings, etc.) and various groups of reconfigurable workspace properties (light intensity, preferred software, font size, etc.), which correspond to the different tasks and can be adapted by the users. The configuration plan model attributes are uniquely identified for each user profile model and constitute the solution retrieved by the DSS matchmaker.

The general process of problem solving of a CBR system follows four distinct stages (figure 4) [3]. In the case retrieval phase the new problem is defined and assessed and the CB is searched for a matching or similar case in order to obtain a relevant solution. As mentioned before, in the context of PEARL, the problem will be defined by the user profile model parameters, entered by the new user. During the retrieval phase the most similar existing profile will be selected and its corresponding configuration plan model will be returned as a solution. Once an initial solution is selected, it is adapted during the following adaptation phase in order to fit the specific requirements of the newly presented problem. The adaptation might be manually done by the user or might be based on a set of rules that account for the indicated differences and perform the necessary changes to the original solution. In the first version of the PEARL DSS module we have implemented a manual adaptation; however, we plan to implement a semi-automatic rule-based adaptation process in the final prototype. The applicability of the resulting solution is then verified in the evaluation phase. Depending on the specific requirements and characteristics of the CBR system, the solution may be evaluated before or after it has been applied. If the verification is successful, the newly formed problem-solution pair can be stored in the CB in the final phase of the CBR cycle. In case the result from the evaluation is unsatisfactory, the solution might be adapted further or additional solutions might be retrieved. In the context of PEARL the solution will be instantly applied by deploying the selected configuration settings and the user evaluation of the proposed solution will take place during the ongoing usage of the PEARL platform. The user will be provided with a user friendly interface to change the configuration settings (i.e. adapt the solution) at any point. Since the CB is dynamically created every time a new request is sent to the DSS, the user adaptations will be automatically applied when the new case is used in the future iterations of the case retrieval process.

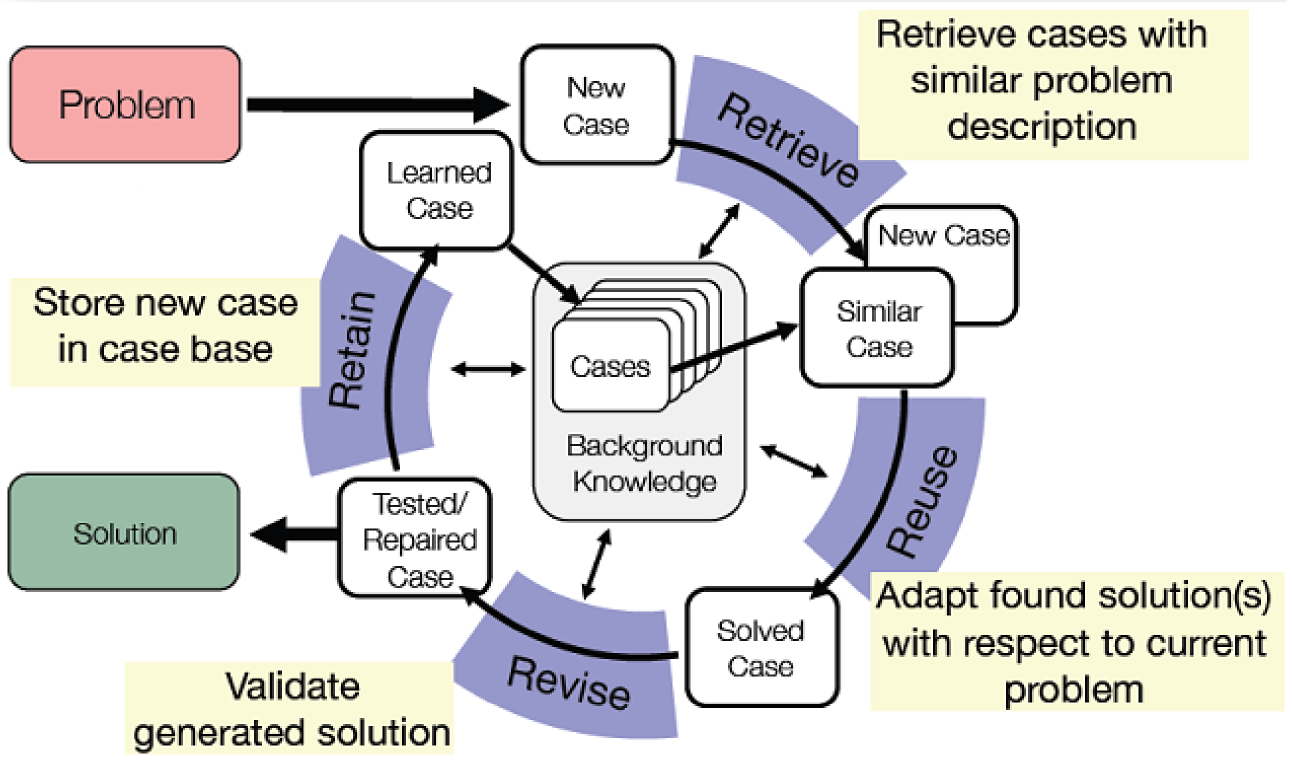

Problem
New Case
Retrieve
Retrieve cases with similar problem description
Learned Case
New Case
Similar Case
Store new case in case base
Retain
Cases
Background Knowledge
Reuse
Solution
Tested/ Repaired Case
Solved Case
Adapt found solution(s) with respect to current problem
Revise
Validate generated solution

Figure 4. CBR reasoning cycle, [1]

In addition to the cases which represent the specific problem situations, a CBR system may include also general knowledge about the specific problem domain, for which the system has been designed. In [4] three types of problem domain knowledge are identified – vocabulary, adaptation knowledge and similarity measures. The vocabulary describes the feature parameters that define each case and are used to retrieve relevant cases. The vocabulary of the PEARL matchmaker will comprise of the user profile and configuration plan model parameters. The adaptation knowledge contains information about the influence of each parameter in the form of explicit rules. In the final prototype of the PEARL DSS a rule-based expert system for case adaptation will be implemented. And finally, the similarity measures encode the similarity model that is utilized. A widely used model is based on the local-global principle, defined in [5], which proposes decomposition of the similarity function into local similarity function that is used for the evaluation of separate individual case attributes, and global similarity function that combines the local similarities and is used to compare case on a higher level. A similar approach that will be described in greater details in the following chapters has been used in the current implementation.

3.1.2 myCBR Similarity-Based Retrieval Tool

In order to accelerate the development process and to take advantage of the state-of-the-art similarity-based retrieval techniques, myCBR tool has been used as the basis for the PEARL DSS matchmaker implementation. myCBR is an open-source similarity-based retrieval tool and software development kit. myCBR is a free software that is distributed under the terms of the GNU Lesser General Public License, as published by the Free Software foundation – either version 3 or any later version. The license allows the developers and companies to use and integrate the software into their products without being required to release the source code of their own components.
myCBR is Java-based and aims to facilitates the development and integration of CBR systems. The tool offers a variety of features [6], such as:

- GUIs for modelling knowledge-intensive similarity measures
- Similarity-based retrieval functionality
- Export of domain model (including similarity measures) in XML
- Extension to structured object-oriented case representations, including taxonomy editors
- Powerful textual similarity modelling
- Scriptable similarity measures using Jython
- Prototyping via CSV
- Improved scalability
- Simple data model (applications can easily be build on top)
- Fast retrieval results
- Rapid loading of large case bases

The process of defining the similarity measures for the various parameters and producing the relevant domain model via the myCBR features will be described in further details in *Chapter 4*.

3.2 Rule-Based Systems and Drools

In order enable the provision of fully customized configuration plans for every new user rule-based semi-automatic case adaptation process will be implemented in the final prototype of PEARL DSS. The implemented rule-based systems will take as an input the configuration plan parameters produced by the CBR matchmaker and will return an adapted solution, based on a set of rules that account for the differences between the new and the already existing users.

3.2.1 Rule-Based Expert Systems Fundamentals

Rule-based systems are the systems that use production rules for knowledge representation. They are claimed to be the simplest form of artificial intelligence [16]. Rules are a set of if-then statements that specify how to act or what conclusion to offer based on a given set of input data. Rules in reality are not independent, but rather can strongly confide in each other. There are two type of rules identified in [7]: knowledge rules that states the facts and their relationships and inference rules that define how two find a solution in the presence of a set of facts. The knowledge rules are stored in the knowledge base, whereas the inference rules are part of the inference engine.

The operation of a rule-based system starts with a knowledge base, which contains the available knowledge in the form of IF-THEN rules and a working memory that might be empty or might contain some input data. The system examines all rule conditions and selects a set off all rules that are satisfied based on the working memory – a conflict set. A rule from the conflict set is then triggered and the actions specified in the THEN clause are performed. The specific rule is chosen based on a conflict resolution strategy. The triggered action might modify the working memory, the knowledge base or execute another action, specified by the rule. The rules are being fired until the termination criterion is met, i.e. if a solution has been found and if it has been determined that the solution is non-existent. When a rule is selected as a part of the conflict set, it is placed on the blackboard (or agenda) for execution. Once the rule is fired it is removed from the agenda and the agenda is updated. A special algorithm is used to reduce the number of comparisons between the facts and the knowledge base. A popular algorithm that performs this action is the Rete algorithm. The Rete algorithm forms a rooted graph or inference tree. Each node of the graph represents the IF-part of a matched rule and stores information about the facts that has been used. When a fact is changed, it propagates through the Rete graph changing the information stored at every node. Thus, the number of comparisons needed when a fact is changed is significantly reduced as the new fact is only matched against the relevant rules. Rete or its enhanced versions have been vastly used in a variety of commercially available rule-based engines, such as Drools for example [17], which will utilized for the needs of PEARL.

Ruled-based expert systems target narrow problem domains and deal with qualitative rather than quantitative problems. As it has been already mentioned above, providing a set of customized configuration parameters, based on a predefined list of user characteristics in the context PEARL, can be characterized as a complex highly unstructured problem. Due to the high number of input and output parameters and the ambiguous interrelations between then, the explicit modelling of the problem domain by a set of rules will be unfeasible within the scope of the project. Therefore, for the needs of PEARL we have implemented a case-based reasoning approach for the selection of the initial solution and we will integrate a rule-based system solely for the needs of adaptation of separate configuration parameters.

3.2.2 Drools Rule Engine

The open source Drools Rule engine has been selected for the needs of PEARL due to its popularity, extensive documentation and support. Drools is released under the Apache Software License 2.0, which allows its reuse for commercial purposes.
Drools Expert is a business rules engine that is part of the Drools BRMS solution, that provides a core Business Rules Engine, a web authoring and rules management application (Drools Workbench), an Eclipse IDE plugin for core development, complex event processing features (Drools Fusion), process/workflow integration for rule orchestration (jBPM), optimization of automated planning (OptaPlanner) and a centralized repository for Drools Knowledge Bases (Drools Guvnor). Some of the key features of Drools Business Rules Engine are summarized below:

- PHREAK Algorithm – pattern matching algorithm based on RETE
- Forward and Backward Chaining Inference
- JSR-94 Compliance – specification for Java Rules Engine API
- Stateless Knowledge Sessions – not utilizing inference (validation, calculation, routing and filtering
- Stateful Knowledge Sessions – allow iterative changes over time (monitoring, diagnostics, logistics, compliance)
- Execution Control – schedules the execution of a set of rules in a deterministic order; conflict resolution
- Truth Maintenance with Logical Objects – allow fact that were asserted to be automatically retracted when the asserting conditions are no longer true
- Decision Tables in Spreadsheets – management of rules in a spreadsheet format
- Logging

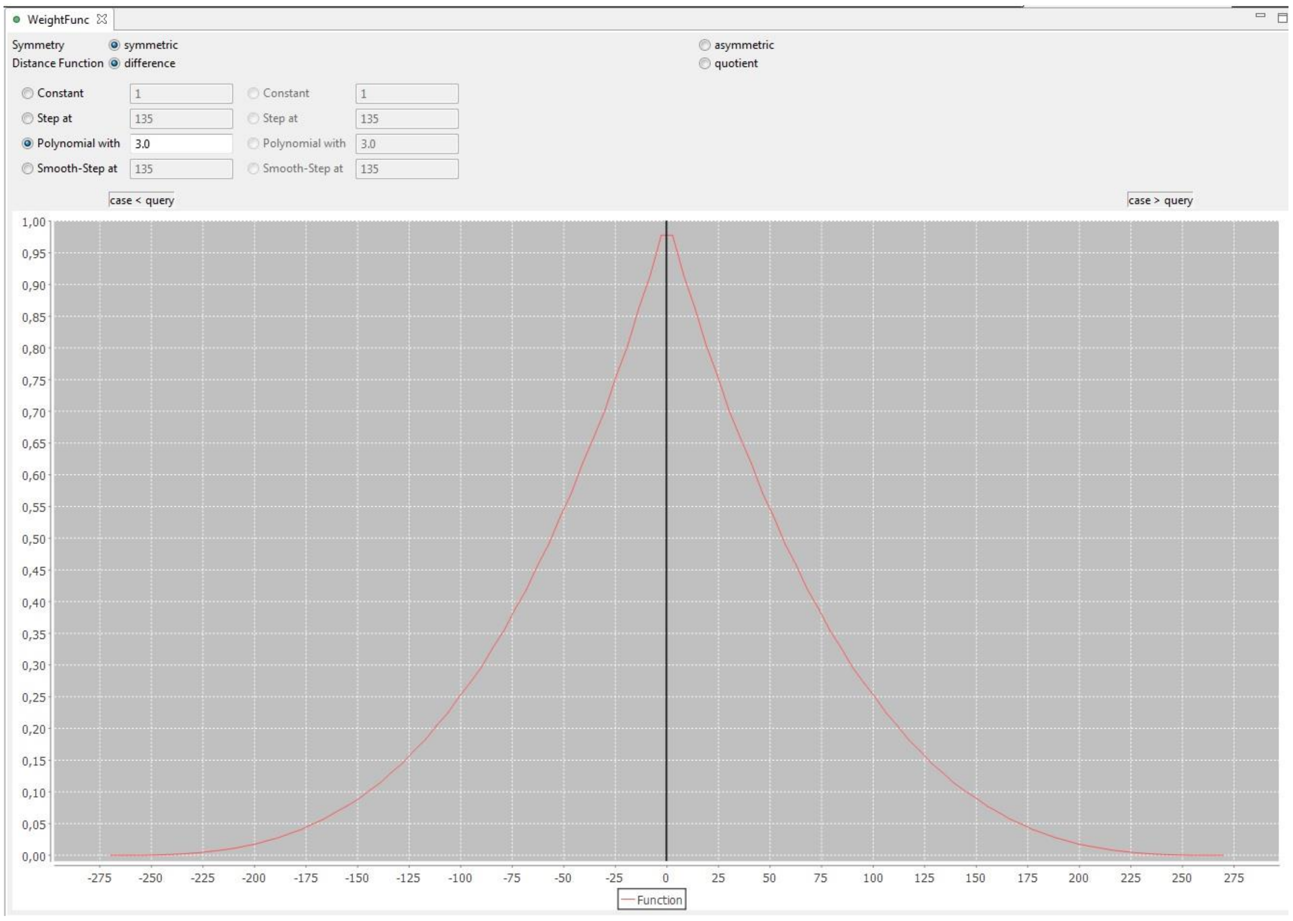
The integration of Drools engine is planned for the final prototype of the PEARL DSS module and therefore it will be described in details in the second version of this deliverable.

4 Modelling of Users and Tasks in the Workspace

The case base of the CBR-based DSS module of the PEARL platform will constitute of a set of problem-solution pairs, where the problem part will be defined by the available user characteristics, as specified by the User Profile Model, and the retrieved solution will provide the Configuration Plan Model entity, available for this particular user. The User Profile Model contains information about user's demographic, cognitive, physical and professional characteristics that will be used for selection and customization of the PEARL platform features. The Configuration Plan Model defines not only the general characteristics of the workspace environment and the list of user's common daily tasks, but also allows the PEARL end user to choose different workspace setups for the different tasks he is involved in. The process of selecting the appropriate parameters and constructing the relevant models will be described in *D4.2 User, Tasks and Workspace Databases, Ontologies and Knowledge Bases*. Here we will focus on the technical representation of available parameters, their incorporation in the CBR engine and the corresponding similarity measures used in the process of case retrieval.

4.1 Similarity Measures and myCBR Workbench

When a new user enters his user profile details in correspondence with the predefined User Profile Model, the PEARL DSS performs similarity-based comparison of the features in order to select the best matching user profile out of the existing case base. For the needs of PEARL, this process has been implemented with the use of the open-source similarity-based retrieval tool myCBR. The system computes the similarity and retrieves the nearest neighbor from the case base. The local-global principle has been applied here by first calculating separately the similarity for each of the individual case attributes, or the local similarity measures, and then applying weighted sum average function to obtain the global similarity measure. The myCBR Workbench tool, which provides convenient GUIs for modeling knowledge-intensive similarity measures, has been used to define the User Profile Model representation and to define the appropriate similarity functions. For more detailed information please refer to [7]. In the context of myCBR the User Profile Model entity is referred to as the concept and all underlying parameters (such as gender, height, colour perception ability, etc.) - as the attributes. The attributes are defined by a data type (Double, Integer, String, Date, Float, Interval, Boolean or Symbol) and available values or value ranges. Detailed description of the attributes definition will be presented in the following chapter. For the needs of PEARL three main data types - Symbol, Float and Integer - have been used. Depending on the data type, myCBR offers a range of different similarity functions for calculating the local similarity measures. For numerical attributes distance functions with predefined similarity behavior, such as constant, single step or polynomial similarity decrease are available. An example of the distance function definition for the numerical attribute *weight* is presented on figure 5.



WeightFunc

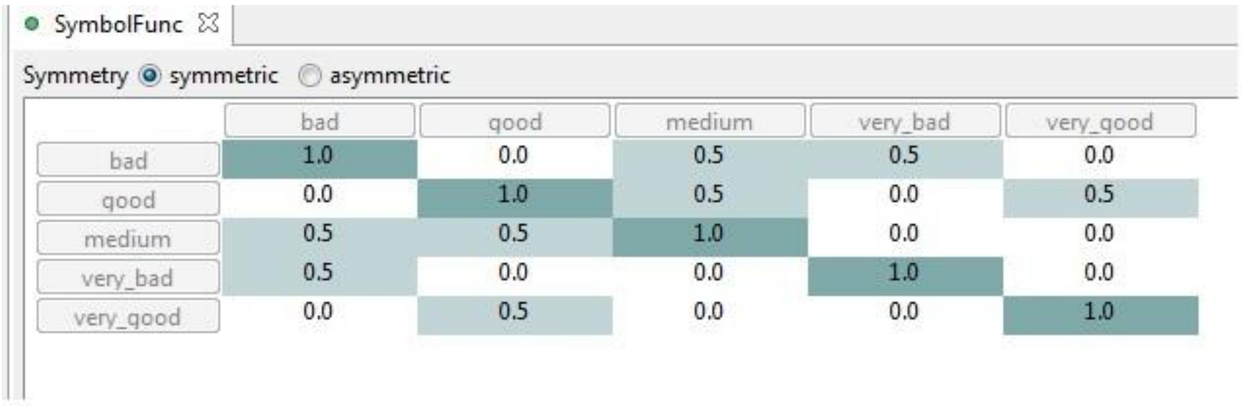
Symmetry [✓ symmetric] asymmetric

Distance Function [✓ difference] quotient

case < query		case > query	
Constant	1	Constant	1
Step at	135	Step at	135
[✓ Polynomial with]	3.0	Polynomial with	3.0
Smooth-Step at	135	Smooth-Step at	135

Figure 5. Polynomial similarity decrease function for the *weight* attribute

As shown on figure 5, myCBR offers a convenient UI for modeling the similarity measures, which allows you to fine tune the applicable parameters. In the case of the *weight* attribute presented above, a symmetric polynomial similarity decrease function has been selected. The majority of the attributes of the User Profile Model, however, has been defined as symbolic attributes undertaking a predefined set of values. For the needs of PEARL the similarity measures for symbolic attributes have been defined using a similarity table. As an example, the similarity table for the colorPerception attribute has been demonstrated in figure 6.



SymbolFunc

Symmetry [✓ symmetric] asymmetric

	bad	good	medium	very_bad	very_good
bad	1.0	0.0	0.5	0.5	0.0
good	0.0	1.0	0.5	0.0	0.5
medium	0.5	0.5	1.0	0.0	0.0
very_bad	0.5	0.0	0.0	1.0	0.0
very_good	0.0	0.5	0.0	0.0	1.0

Figure 6. Similarity table for the colorPerception attribute

It can be noted that all possible values of the symbol attribute *colorPerception* are predefined and all pairwise combinations together with their similarities are explicitly defined.
The global similarity function for the User Profile Model concept is also defined with the help of myCBR Workbench. A weighed sum function has been chosen for the needs of the current project and the appropriate weights have been assigned.

4.2 User Profile Model

The User Profile Model concept includes a range of demographic, cognitive, physical and professional characteristics of the each user that are used in the process of retrieval and customization of the configuration plan for each new user. The User Profile Model attributes define the problem part of each case that is used by the CBR engine to retrieve a matching or similar solution. The User Profile Model concept and the corresponding attribute values and similarity measures were defined with the help of myCBR Workbench UI and are listed below. In the process of implementation the User Profile Model entity is represented by the *UserProfileModel* class.

Table 1. User profile model attributes

Name	Data type	Attribute Type	Available values	Similarity Function
abilityToAdapt	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
abilityToLearnIndependently	String	Symbol	good, medium, bad	Similarity Table
anxiousnessICT	String	Symbol	very_high, high, medium, low, very_low	Similarity Table
arthritisInHands	String	Symbol	none, medium, severe	Similarity Table
attentionAbility	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
backPain	String	Symbol	none, medium, severe	Similarity Table
colourPerception	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
communicationSkill	String	Symbol	none, listening, reading, speaking, writing	Similarity Table
entrepreneurialSkills	String	Symbol	good, medium, bad	Similarity Table
fieldOfVision	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
flexibility	String	Symbol	high, low, medium	Similarity Table
gender	String	Symbol	Male, female	Similarity Table
handEyeCoordination	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
hearingAbility	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
height	Integer	Integer	$50 \leq height \leq 250$	Polynomial similarity difference function
knowledgeNavigationSkills	String	Symbol	good, medium, bad	Similarity Table
languageProduction	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
languageReception	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
literacyICT	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
longTermMemory	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
lungProblems	String	Symbol	none, medium, severe	Similarity Table
problemSolvingSkills	String	Symbol	good, medium, bad	Similarity Table
processingSpeed	String	Symbol	good, medium, bad	Similarity Table
setConfigurations	Integer	Integer	0,1	-

shouldersPain	String	Symbol	none, medium, severe	Similarity Table
socialSkills	String	Symbol	good, medium, bad	Similarity Table
teamworkSkills	String	Symbol	good, medium, bad	Similarity Table
testSite	String	Symbol	Netherlands, Romania, Switzerland	Similarity Table
understandingSigns	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
visualSensitivity	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
weight	Integer	Integer	30 ≤ weight ≤ 300	Polynomial similarity difference function
workingMemory	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
yearBirth	Integer	Integer	1940 ≤ yearBirth ≤ 2000	Smooth-step similarity difference function
yearsEmployed	Float	Float	0 ≤ yearsEmployed ≤ 50.0	Polynomial similarity difference function

4.3 Configuration Plan Model

The configuration plan model incorporates information about each user's most common daily tasks, the corresponding ambient environment configuration settings and a number of predefined settings of the modules constituting the PEARL Application Layer. There is a unique configuration plan entity corresponding to each registered user, which is retrieved as a solution by the PEARL DSS. The configuration plan model is designed in such a way so as to allow the user to select specific ambient configuration setups for each of the common daily tasks that can be selected via the PEARL Task Switcher. For example, the user will be able to select specific font size, screen resolution, light intensity, availability indicator value, etc. for e-mail related tasks, whereas for the times when s/he is involved in multimedia related activities, s/he might choose to have better screen resolution and smaller font size. The user will also have the option to skip the step of assigning any specific configurations and use the general configuration settings values. The Configuration Plan Model attributes are presented in the table below.

Table 2. Configuration plan model attributes

Name	Data Type	Available Values
General Preferences		
PC Configuration		
fontSize	String	Small, Medium, Large
soundLevel	String	Mute, Low, Medium, High
screenResolution	String	Low, Medium, High
Ambient Configurations		
roomLightIntensity	String	Off, Low, Medium, High
deskLightIntensity	String	Off, Low, Medium, High
lightTemperature	String	Warm, Medium, Cold
Login Task		
defaultTask	String	general, email, calendar, text-editing, spreadsheet-editing, presentation-editing, meeting, web-browsing, researching, task management, organization specific tasks, quick notes, travelling, eLearning, multimedia, cognitive games, searching
Task Modelling		
For each of the available tasks: **email, calendar, text-editing, spreadsheet-editing, presentation-editing, meeting, web-browsing, researching, task management, organization specific tasks, quick notes, travelling, eLearning, multimedia, cognitive games, searching**		

preferredSoftware	String	*The available software product name, related to each of the tasks.*
availabilityIndicator	String	Blue, Green, Orange, Red
taskWorkspace	Integer	0,1
fontSize	String	Small, Medium, Large
soundLevel	String	Mute, Low, Medium, High
screenResolution	String	Low, Medium, High
roomLightIntensity	String	Off, Low, Medium, High
deskLightIntensity	String	Off, Low, Medium, High
lightTemperature	String	Warm, Medium, Cold
Physical Wellbeing Module Settings		
setExerciseNotifications	String	yes, no
exerciseGoal	Integer	the integer value in minutes
shortenSittingPeriods	String	yes, no
maxSittingDuration	Integer	the integer value in minutes
setRelaxingExercises	String	yes, no
setAgendaItems	String	yes, no
coverredPeriod	String	working_hours, work_week, always
typeOfExercises	String	Individual_exercises, team_exercises, mix
eLearning Module Settings		
selectedCourses	Array	*List of available courses*
setElearningNotifications	String	yes, no
sessionsPerWeek	Integer	Integer value between 0 and 5
Cognitive Games Module Settings		
gameCategory	String	attentionGames, memoryGames, reasoningGames, logicGames, orientationGames, languageGames, constructiveGames
gameDifficulty	String	easy, normal, hard
Task and Time Management Module Settings		
taskCategories	Array	*List of selected tasks*
agendaCategories	Array	List of agenda categories

4.4 Example instantiations

After building the vocabulary in terms of the User Profile Model concept and its attributes and defining the appropriate similarity measures, the next step in the CBR system design is the process of knowledge acquisition and building the case base. In order to obtain the initial case instantiations each of the partners involved in user requirements collection and analysis process was asked to generate two real life problem-solution pairs (user profile model characteristics and the corresponding configuration plan settings), based on the interviews conducted with potential end users. The resulting examples were stored in the database and were used in the development stage for testing the case retrieval functionality and refining the similarity measures and as a proof of concept during the project review demonstration. One of them is presented in Appendix A.

It is important to note that the PEARL DSS matchmaker is designed to dynamically create the case base by retrieving the most recent user information stored in the database. This gives two major advantages. First, the case base will be automatically extended with each new successfully registered user. Second, the existing solutions will be constantly validated and refined by the users as long as they are actively using the PEARL platform. In this manner the initial case base will be extended during the first user trials by collecting the participants' user profile information and their corresponding configuration preferences. At this point the similarity measures definition will be refined in order to improve the case retrieval process.

4.5 Adaptation rules

Due to the limited amount of available cases and the inherent ambiguity of the available solutions (two similar users might have different personal preferences) the need of implementing appropriate case adaptation logic has been identified. We have chosen to incorporate a rule-based expert system that will allows us to define a set of formalized rules for case adaptation, based on the differences between the User Profiles Model parameter of the new case and the case retrieved by the PEARL matchmaker. The implementation of the adaptation logic and the definition of the formalized rules have been planned for the second and final iteration of the PEARL DSS module.

5 Components Implementation and Interfaces

The PEARL DSS module is one of the major components that the PEARL Business Intelligence Layer is comprised of. The main purpose of the module is to produce a customized configuration plan for each new user during the initial registration process in order to expedite the initial platform setup and to improve the overall user experience. Essentially, the module was implemented as a RESTful web service that takes as an input the ID of the newly registered user and returns the customized configuration plan in JSON format. PEARL DSS interacts directly only with two other PEARL components – the Preference Editor and the PEARL Data Layer. The Preference Editor provides a user friendly interface that allows the user to define, store and manipulate his/her profile characteristics and configuration settings. The module is responsible for triggering the DSS when the user saves his User Profile Model parameters and then display the resulting configuration plan settings for the user to adjust and save them. The PEARL Data Layer hosts all relevant user profile data in a MySQL database. During the process of configuration plan customization the DSS module performs several database queries to retrieve the new User Profile Model, to dynamically build the case base and to retrieve a relevant configuration plan settings for customization.

5.1 Basic Architecture and Principle of Operation

In order to comply with the requirements of the other PEARL platform modules, the PEARL DSS was designed and implemented as a RESTful web service. REST () architectural style allows web services to be designed to serve specific resources based on client request. Spring annotation-based Model-View-Controller (MVC) framework was chosen for the implementation of the PEARL DSS REST web service due to its extensive documentation and library support. The basic architecture and principle of operation are illustrated in figure 7 and the data flow is described in the form of a sequence diagram in figure 8.

The process goes through the following steps:

1. **Client Request** - PEARL DSS web service is initially invoked by an HTTP request containing the new user ID sent by a client application, in this case the PEARL Preference Editor. The details of the exposed API will be discussed in the following section.
2. **Front Controller** – the http request is intercepted by the Dispatcher Servlet, defined in the *web.xml* file:

```
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>
```

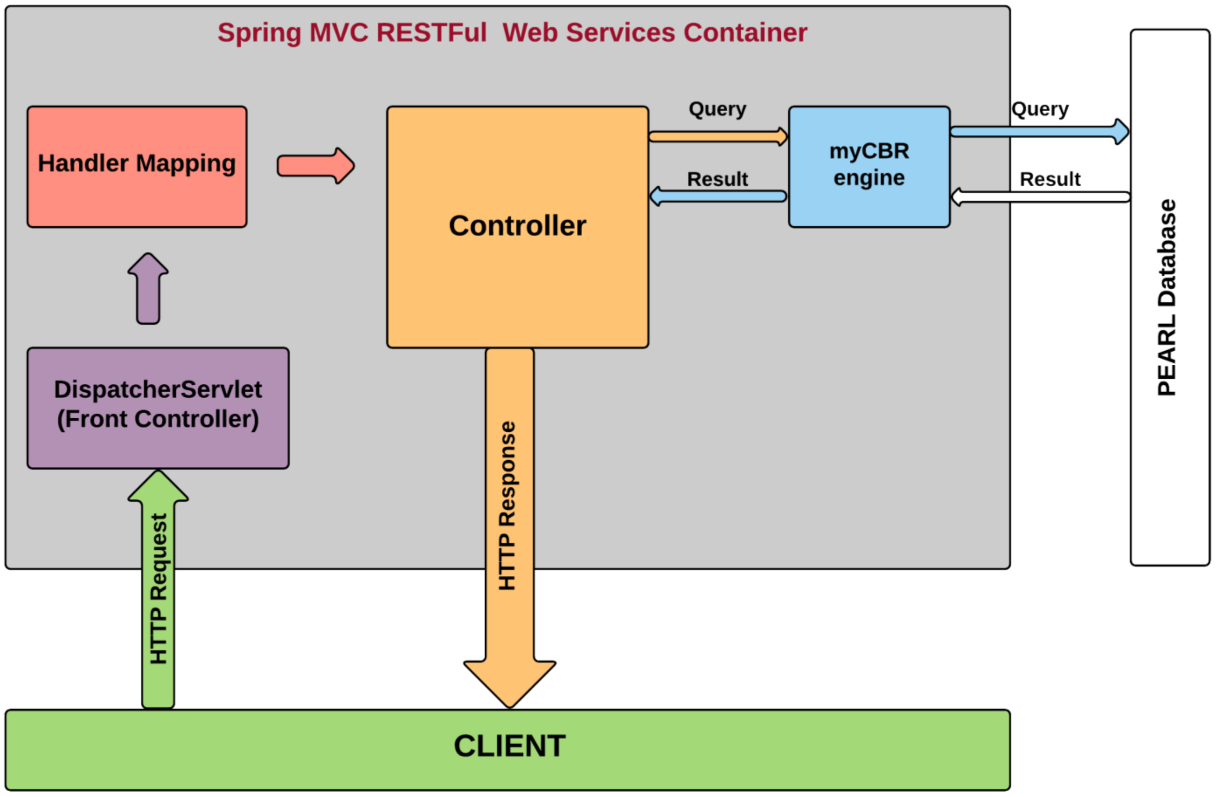


Figure 7. PEARL DSS architecture and principle of operation

Spring framework **DispatcherServlet** class has been used, which dispatches the HTTP requests to registered handlers and provides mapping and exception handling capabilities. From the code snippet above we can see that the URL mapping pattern and the path to the application context file have been defined.

3. **Handler Mappings** – this section is defined in the application context file and defines how the Dispatcher Servlet should identify the appropriate controller to handle the incoming HTTP requests. In the current project, an annotation driven approach has been implemented by defining the specific URL or URL pattern for every controller handle method via the **@RequestMapping("URL-pattern")** annotation.

4. **Controller** – the controller is the core abstraction in the Spring MVC framework. The controller is a java object that implements a set of classes that are invoked to handle HTTP requests. The controller classes are defined by annotating them with **@Controller** annotation. The main controller class responsible for invoking the myCBR engine, retrieving the best configuration plan and returning a REST representation of the *ConfigurationSettingsEntity* class is presented in the code snippet below. It can be noticed that the user_id parameter is annotated with **@PathVariable**, which allows it to be initialized with the value of the corresponding path segment. First, the controller retrieves the new user profile parameters directly from the database via the *findByUserProfileId* interface. Then, it initializes the myCBR engine instance, performs a query and returns the id of the best matching user profile model from the case base. Finally, the controller retrieves the best matching configuration plan from the database and converts the configuration plan object to a REST representation using *com.fasterxml.jackson.core* library and

Spring HATEOS resource assembler. The returned response contains the assembled resource with the configuration plan parameters in JSON format and the appropriate HTTP status code.

```
@Controller
@RequestMapping("/rest")
public class ConfigurationsEntityRetrievalController {
…
    @RequestMapping(value="/retrieveconf/{user_id}",method = RequestMethod.GET)
    public ResponseEntity<ConfSetEntityResource> retrieveGenPrefByUserId(
            @PathVariable Integer user_id, SessionStatus status) {
      newUser = userProfileDao.findByUserProfileId(user_id);
      if(engine == null) {loadengine();}
             else{ /*do nothing, engine is already loaded */ }
      matching_user_id = solveQuery(newUser);
      ConfigurationSettingsEntity entry;
      entry = service.retrieveConfSetByMatchingUserId(matching_user_id);
      if(entry != null)
        {
             entry.setUser_id(user_id);
             ConfSetEntityResource res = new
ConfSetEntityResourceAsm().toResource(entry);
             matching_user_id=null;
             status.setComplete();
             return new ResponseEntity<ConfSetEntityResource>(res, HttpStatus.OK);
        } else {
             matching_user_id=null;
             status.setComplete();
             return new
ResponseEntity<ConfSetEntityResource>(HttpStatus.NOT_FOUND);
        }
}
```

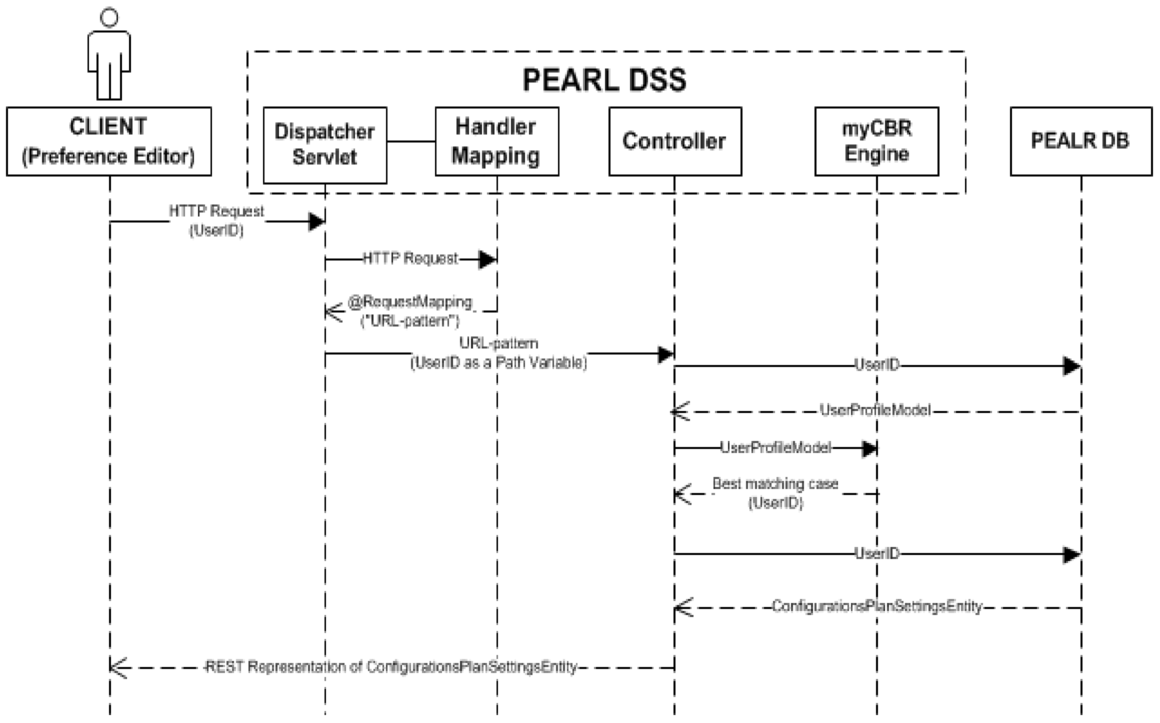

PEARL DSS
CLIENT (Preference Editor)
Dispatcher Servlet
Handler Mapping
Controller
myCBR Engine
PEALR DB
HTTP Request (UserID)
HTTP Request
@RequestMapping ("URL-pattern")
URL-pattern (UserID as a Path Variable)
UserID
UserProfileModel
UserProfileModel
Best matching case (UserID)
UserID
ConfigurationsPlanSettingsEntity
REST Representation of ConfigurationsPlanSettingsEntity

Figure 8. PEARL DSS REST service data flow sequence diagram

5. **myCBR engine** – the engine uses the myCBR open source similarity retrieval tool core libraries, including methods for database access and case base construction and similarity-based case retrieval. The engine is instantiated and the *solveQuery* method for case retrieval is invoked by the Controller class. The case base is imported by the *DatabaseImporter* class directly for the PEARL database. *DatabaseImporter* class relies on an xml configuration file that stores the database connection string, defines the CBR concept of interest and maps existing database tables to CBR project attributes. The *solveQuery* method takes as an input an instance of the *UserProfileModel* class (defined in section 4.2 User Profile Model) and converts the class variables to CBR concepts attributes. The method invokes the appropriate case retrieval method, prepares and executes the case base query and retrieves the id of the best matching case.
6. **HTTP Response** – the controller returns the response directly to the client.

5.2 REST API Definition

A detailed description of the parameters of the exposed web service is presented in the table below. The web service is invoked by the Preference Editor when the user stores its user profile information during the initial registration process.

Table 3. REST API definition

Method	**GET** *{root}/pearl_bil/rest/retrieveconf/{user_id}*		
Headers	Content-Type		application/json
API definition and request query parameters			
Service Name	**Input**	**Output**	**Info**
retrieveConfPlanByUserId	Integer *user_id*	ConfSetEntityResource *res*	The output will be a success/error message/code and JSON representation of the configuration plan. *user_id* is the ID of the newly created user profile. *res* is the REST representation of the *ConfigurationSettingsEntity* class, created using Spring HATEOS resource assembler.
Response representations			
Response code	**Response message**	**Response body**	
200	OK	{ "user_id": Integer, "tasksSettings": [{ "taskSettingsMap": { "taskName": String, "soundLevel": String, "screenResolution": String, "roomLightIntensity": String, "deskLightIntensity": String, "lightTemperature": String, "defaultTask": String, "preferredSoftware": String, "availabilityIndicator": String, "taskWorkspace": Integer } },	

		{ "taskSettingsMap": { "taskName": String, "soundLevel": "low", "screenResolution": String, "roomLightIntensity": String, "deskLightIntensity": String, "lightTemperature": String, "preferredSoftware": String, "availabilityIndicator": String, "taskWorkspace": Integer } }, { "taskSettingsMap": { "taskName": String, "availabilityIndicator": String, "preferredSoftware": String } }], "links": [] }
404	NOT_FOUND	-

5.3 Database Access

The PEARL DSS module retrieves the relevant user profile and configuration plan information directly from the PEARL Database. The database connectivity is realized via the Spring Framework Java Database Connectivity (JDBC) integration with the use of the *JdbcTemplate* utility class*.* Spring JDBC Framework provides automation of a number of low-level tasks, such as opening the connection, preparation and execution of the SQL statements, process exceptions, transaction handling and closing the connection. *JdbcTemplate* class executes SQL queries, update statements and stored procedure calls, performs iteration over ResultSets and extraction of returned parameter values. In the current implementation a *DataSource* was configured in the Spring configuration file and the shared DataSource bean was dependency-injected in the implemented Data Access Object (DAO) classes. The following code snippet presets the definition of the *DataSource* bean.

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="com.mysql.jdbc.Driver" />
      <property name="url" value="jdbc:mysql://localhost:3306/pearl" />
      <property name="username" value="secret" />
      <property name="password" value="secret" />
</bean>
```

The interaction with the database has been implemented via a number of DAO classes, which expose their functionality thought the corresponding interfaces. The following database interaction methods have been implemented:

1. **User Profile Model Retrieval** – returns *UserProfileModel* object, based on *user_id.*
2. **General Preferences Retrieval** - returns *GeneralPreferences* object, based on *user_id*. The *GeneralPreferences* object contains information about the *user_id*, the default task selected by the user to be launched upon login and the default ambient environment configuration parameters.

3. **User Task Retrieval** – returns *UserTasks* object, based on *user_id*. The *UserTasks* object contains the *user_id* and a Java List collection with the common daily tasks, selected by the user.
4. **Task-Workspace Characteristics Retrieval** – returns Java HashMap collection that stores all configuration parameters, related to a given task. The retrieval is based on *user_id* and *tasks_id.*
5. **Database Importer** – import the case base from PEARL MySQL database.

6 Implementation Deployment Details

The PEARL DSS module has been packaged as a .war file and deployed on a virtual machine hosted in the premises of SingularLogic (SiLo). For the needs of PEARL, a WildFly application server has been deployed and managed by SiLO. WildFly is an open-source software that implements the Java Platform Enterprise Edition (Java EE) specification. The application server specifications are as follows:

- version: wildfly-9.0.1.Final
- listening port: 8081
- public endpoint: pearl-atl.euprojects.net

7 Further Work

The current deliverable provided an overview of the architecture design, the principle of operation, CBR project customization process and the implantation logic of the PEARL DSS module. The document described in details the prototype implementation resulting from T4.5 Decision Support System and Workflow Engines. The prototype was successfully implemented and integrated with the relevant PEARL platform components and is fully operational. Our future work will follow two major directions. First, the operation of the first prototype will be extensively tested and the performance of the case retrieval functionality will be analysed; the purpose of this is to optimize the existing similarity measures in order to improve the success rate of the case retrieval. Furthermore, the case base will be significantly extended during the first user trials, which will naturally improve the overall performance of the module. The second direction of work that we plan to follow is defining the case adaptation rules and integrating Drools rule engine in the existing prototype. As it was previously stated, we have identified the need of implementing appropriate case adaptation logic due to the limited amount of available cases and the inherent ambiguity of the available solutions. Therefore, our main efforts will be focused on developing a functional rule-based case adaptation system.

References

[1] Maja Pantic, Introduction to Machine Learning and Case-Based Reasoning, Course Introduction, Inmperial College London, 2011

[2] Kolodner, Janet. Case-Based Reasoning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[3] A. Aamodt, E. Plaza (1994); Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, pp. 39-59.

[4] M.M. Richter, “On the notion of similarity in case-based reasoning”, Mathematical and Statistical Methods in Artificial Intelligence, G. della Riccia, R. Kruse, R. Viertl, (Eds.), pp. 171-184. Heidelberg, Germany: Springer-Verlag, 1995

[5] M. Richter. Foundations of similarity and utility. In Proceedings of the 20th International Florida ArtiØcial Intelligence Research Society Conference (FLAIRS 2007). AAAI Press, 2007.

[6] Documentation: myCBR 3.1, available at http://mycbr-project.net/downloads/myCBR_3_tutorial_slides.pdf

[7] C. S. S. Kerstin Bach, “Knowledge Modeling with the Open Source Tool myCBR,” CEUR Workshop Proceedings, vol. 1289, 2014.

Appendix A Example case instantiation

A.1. User 4 – RRD

1. User Input Data

a. User Modeling

Demographics

- Year of birth (1963)
- Sex (Male)
- Preferred language (Dutch)

Physical Characteristics

- Height (189)
- Weight (78)
- Lower back pain (none)
- Neck/shoulders pain (none)
- Arthritis in hands (none)
- Lung problems (none)
- Flexibility (none)

Occupation and competences

- Test site (NL)
- Job position (Senior researcher)
- Years employed (11)

Underlying Skills

- ict_literacy (very good)
- ict_anxiousness (very low)

List of eLearning related skills & competences[1]

- Communication skill that needs greatest improvement (none)
- ability to learn independenty (excellent)
- social skills - ethics, positive attitude, responsibility (excellent)
- teamwork skills - collaborative learning, networking (can be improved)
- ability to adapt to changing circumstances (excellent)
- problem solving and logical thinking (excellent)
- knowledge navigation (excellent)

Perceptual

- Visual_acuity_and_sensitivity (very bad)
- Colour_perception (Medium)
- Field_of_vision (very bad)
- Hearing_ability (good)

[1] References: Gilbert.J, 2005: Catching the Knowledge Wave: the Knowledge Society and the Future of Education, Wellington, NZ & Conference Board of Canada, 1991: Employability skills profiles.

Cognitive

- Language_reception (very good)
- Language_production (very good)
- Understanding_abstract_signs (good)
- Attention (good)

Information Processing

- processing_speed (good)
- working_memory (good)
- long_term_memory (good)
- hand_eye_coordination (medium)

a. Workspace Modeling

Room

- Size (small)
- Present colleagues (0)
- Noise level (silent)
- Artificial Light
  - Ceiling (1)
  - Table Light (0)
- Number of desks (1)
- Number of chairs (2)

Hardware

- Availability Indicator (0)
- Internet (1)
  - Bandwidth (fast)
- External Monitor (1)
  - Touch (0)
  - Screen size (normal)
- Speakers (1)
  - Type (loudspeakers)
- RFID reader (1)
- Mouse (1)
- Touchpad (1)
- Microphone (1)
- Webcam (1)
- Desk phone (1)

2. Configuration settings

a. General Preferences

PC Configuration

- Font Size: (extra) Large
- Sound Level: Mute
- Screen resolution: High

Ambient Configuration

- Room light intensity (if available): High

- Desk light intensity (if available): Off
- LightTemperature: cold

Log in task

- Choose a default task to lanunch at startup (check e-mail)

b. Task Modeling

E-Mail (1)

- Preferred e-mail client (Outlook)
- Check emails (1)
- Prioritize emails (1)
- Sort emails (1)
- Respond to email (1)
- Write new email (1)
- Archive emails (1)
- Availability indicator (2)
- Add specific workspace configuration (1)

  - **PC Configuration**
    - Font Size: extra large
    - Sound Level: Mute
    - Screen resolution: High

  - **Ambient Configuration**
    - Room light intensity (if available): High
    - Desk light intensity (if available): Off
    - LightTemperature: cold

Calendar (1) (Table: Tasks | Table: UserTasks)

- Preferred Calendar App (Outlook Calendar)
- Open schedule (1)
- Create new appointment (1)
- Edit existing appointment (1)
- Availability indicator (2)
- Add specific workspace configuration (0)

Searching and archiving documents (1)

- Preferred Search App (Windows Search)
- Search file using Windows search (1)
- Open Windows Explorer to archiving/manage documents (1)
- Availability indicator (2)
- Add specific workspace configuration (0)

Text-Editor (Word) (1)

- Preferred text editor (Microsoft Word)
- Open and read text document (1)
- Edit text document (1)
- Create new text document (1)
- Availability indicator (2)
- Add specific workspace configuration (0)

Spreadsheet (Excel) (1)

- Preferred spreadsheet editor (Microsoft Excel)
- Open and read spreadsheet (1)
- Edit spreadsheet (1)
- Create new spreadsheet (1)
- Availability indicator (2)
- Add specific workspace configuration (0)

Presentation (Powerpoint) (1) (Table: Tasks | Table: UserTasks)

- Preferred presentation editor (Microsoft Powerpoint)
- Open and read presentation (1)
- Edit presentation (1)
- Create new presentation (1)
- Availability indicator (2)
- Add specific workspace configuration (0)

Meeting (1) (Table: Tasks | Table: UserTasks)

- Conference Call (1)
  - Preferred communications messenger (Skype)
  - Face-to-face meeting (1)
- Availability indicator (2)
- Add specific workspace configuration (1)

  - **PC Configuration**
    - Font Size: extra large
    - Sound Level: Medium
    - Screen resolution: High

  - **Ambient Configuration**
    - Room light intensity (if available): Medium
    - Desk light intensity (if available): Off
    - LightTemperature: cold

Browser (1) (Table: Tasks | Table: UserTasks)

- Preferred browser (Firefox)
- New tab (1)
- Search in search engine (e.g. Google) (1)
- Bookmark site (1)
- Edit bookmarks (1)
- Availability indicator (2)

Researching (1)

- Preferred reader (Abobe Reader)
- Reading articles (1)
- Literature search (1)
- Availability indicator (2)
- Add specific workspace configuration (0)

Task management (1)

- Preferred task management app (Outlook ToDo bar)
- View pending tasks (1)
- Add new task (1)
- Edit/complete existing task (1)
- Availability indicator (2)

Organization specific tasks (1)

- Open company's sharepoint/intranet portal (1)
- Open company-specific time registration portal/program (1)
- Availability indicator (2)

Traveling (1)

- To other location within the company (1)
- To other location outside the company (1)

c. *Physical Wellbeing module settings*

Physical activity

- Do you like PEARL to help you to be sufficient physically active? [YES]
- We recommend you to be at least 30 min. active during working hours each day. Please set your goal: [30]

Sitting duration

- Do you like PEARL to help you shortening your sitting periods? [NO]
- We recommend you to sit shorter than 45 consecutive min. Please set your max. sitting duration: [-]

Wellbeing

- Do you like PEARL to suggest relaxing exercises? [NO]

Planning physical activity

- PEARL can help you plan your physical exercises by proposing agenda items in empty time slots in your agenda to optimize your physical pattern. Would you like to try this? [NO]

(Work)week feedback

- When during the week would you like to be supported by PEARL on your physical wellbeing? [radio button: * During workings hours (9u – 17u)]

Social component

- PEARL can provide exercises that you can do by yourself or with others (e.g. colleagues). Which one do you prefer? Please provide me with: [radio button: a mix of individual exercises]