



### Project Identification

<b>Project number</b>	AAL-2013-6-091
<b>Duration</b>	1 <sup>st</sup> June 2014 – 30 <sup>th</sup> November 2016
<b>Coordinator</b>	Univ. Prof. Dr. Manfred Tscheligi
<b>Coordinator Organization</b>	AIT Austrian Institute of Technology GmbH, Austria
<b>Website</b>	<a href="http://www.pearl-project.eu">www.pearl-project.eu</a>



## Platform for Ergonomic and motivating, ICT-based Age-friendly woRkpLaces (PEARL)

### Document Identification

<b>Deliverable ID:</b>	D-4.4.2 Decision Support System and Rule Engines
<b>Release number/date</b>	V1.0 06.06.2016
<b>Checked and released by</b>	Anton Katov/AAU

### Key Information from "Description of Work"

<b>Deliverable Description</b>	This deliverable describes the final prototype implementation of the programmable interfaces and configuration management tools.
<b>Dissemination Level</b>	PU = Public
<b>Deliverable Type</b>	P = Prototype, R = Report
<b>Original due date</b>	Project Month 24 / 01.June.2016

### Authorship & Reviewer Information

<b>Editor</b>	Anton Katov/AAU
<b>Partners contributing</b>	AAU, SiLO, SENSAP, AIT, RRD
<b>Reviewed by</b>	Kostas Perakis/SiLo

## Abbreviations

<i>Abbrev.</i>	<i>Description</i>
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
KIE	Knowledge Is Everything (Drools core)
MVC	Model-View-Controller
REST	Representational State Transfer
UI	User Interface

## Table of Contents

<i>Abbreviations</i>	II
<i>Table of Contents</i>	III
<i>Executive Summary</i>	1
1 <i>About this Document</i>	2
1.1 Role of the deliverable	2
1.2 Relationship to other PEARL deliverables	2
1.3 Summary of changes over the previous version	2
2 <i>Requirements and System Overview</i>	3
2.1 Requirements of PEARL Project	3
2.2 Architecture Concept and Principles of Operation	3
3 <i>Underlying Technologies and Tools</i>	6
3.1 Case-Based Reasoning and myCBR Engine	6
3.2 Rule-Based Systems and Drools	9
4 <i>Modelling of Users and Tasks in the Workspace</i>	11
4.1 User Profile Model	11
4.2 Configuration Plan Model	12
4.3 Example instantiations and case base building	14
4.4 Knowledge refinement process	14
4.5 Similarity Measures and myCBR Workbench	15
4.6 Adaptation rules	19
5 <i>Components Implementation and Interfaces</i>	22
5.1 Basic Architecture and Principle of Operation	22
5.2 REST API Definition	26
5.3 Database Access	27
6 <i>Implementation and Deployment Details</i>	29
6.1 System specifications	29
6.2 Dependencies and packaging	29
6.3 Installation	30
7 <i>Conclusions</i>	31
<i>References</i>	32
<i>Appendix A Example case instantiations</i>	33
A.1. User 1 – AIT	33
A.2. User 2 –AIT	36
A.3. User 3 - RRD	39

A.4. User 4 – RRD

42

## 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. D4.4.2 Decision Support System and Rule Engines aims to build upon the summary provided in the first version of the deliverable, describing in complete details the outcome of T4.5 Decision Support System and Workflow Engines in terms of implementation logic and underlying technologies of the final PEARL Decision Support System prototype.

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. The two main types of knowledge bases that were used by the DSS were discussed – the case base, used by the CBR matchmaker for similarity-based retrieval, and the rule base, rule-based engine for case adaptation. Finally, the implementation and deployment details were discussed.

# 1 About this Document

## 1.1 Role of the deliverable

The role of D4.4.2 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 final prototype of 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 for the CBR-based matchmaker, as well as the process of rule declaration and the produced rule base for the needs of the Drools adaptation rule engine. The document is to be used as a functional overview of the final prototype of the DSS module and to 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:

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

## 1.3 Summary of changes over the previous version

Deliverable 4.4.2 can be viewed as an updated and extended version of the first version of the document. The main reason for following such an approach is to provide a consolidated document that fully describes all the aspects of the final prototype of PEARL DSS. Several major editions and extensions are made as compared to deliverable 4.4.1. The major development efforts are directed towards implementing and incorporating the Drools adaptation rule-based engine into the existing system and defining a set of rules that will define the necessary adaptation actions with regards to the input parameters. The rules definition is described in section 4.6, while the implementation technique is discussed in section 5.1. The description of the implementation, packaging, installation and deployment of the holistic DSS service is also extended in chapter 6 to provide more detailed guidance for future use. Another major improvement over the first version of the prototype is the redefinition of the similarity measures and the extension of the case base to ensure better performance and accuracy, discussed in the relevant sections in chapter 4.

## 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 the PEARL Business Intelligence Layer and undertakes the tasks of mapping the user profile, workspace and task characteristics to a set of configurations parameters, adjusted to the needs of the particular user. The module is 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 are 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:

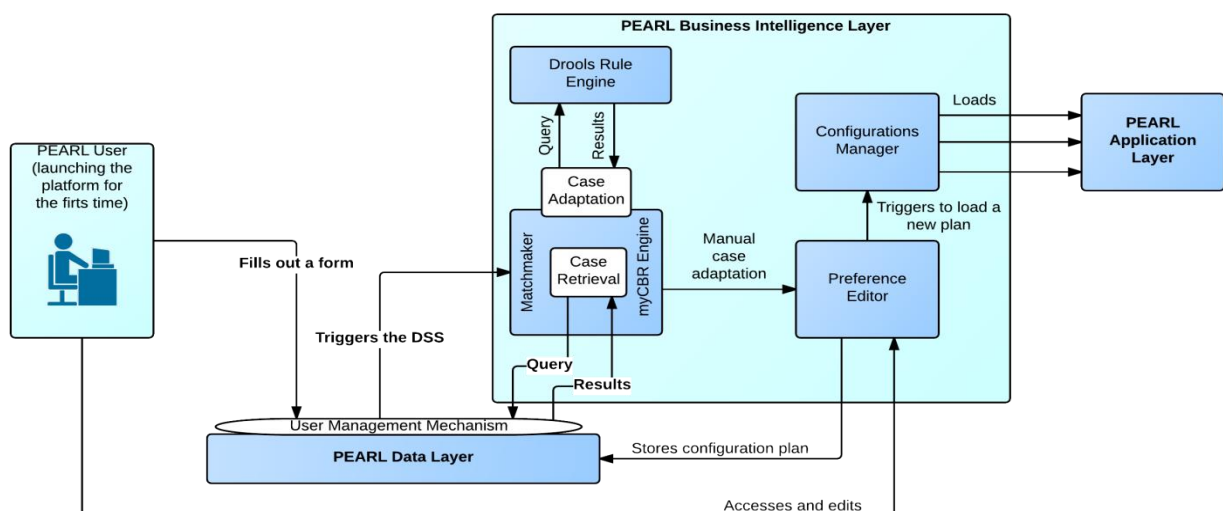


Figure 1. PEARL DSS Architecture

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 appropriate knowledge base, which defines a set of adaptation rules was developed and incorporated the PEARL DSS system.

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.

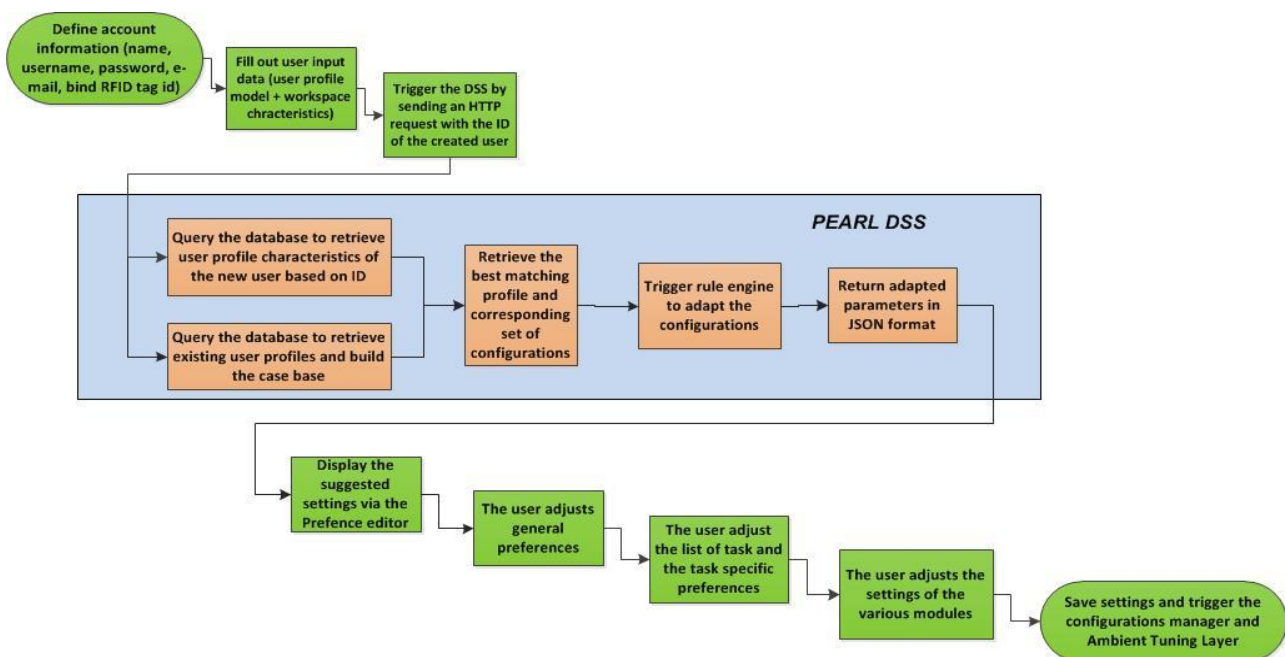


Figure 2. New User Registration Data Flow

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.

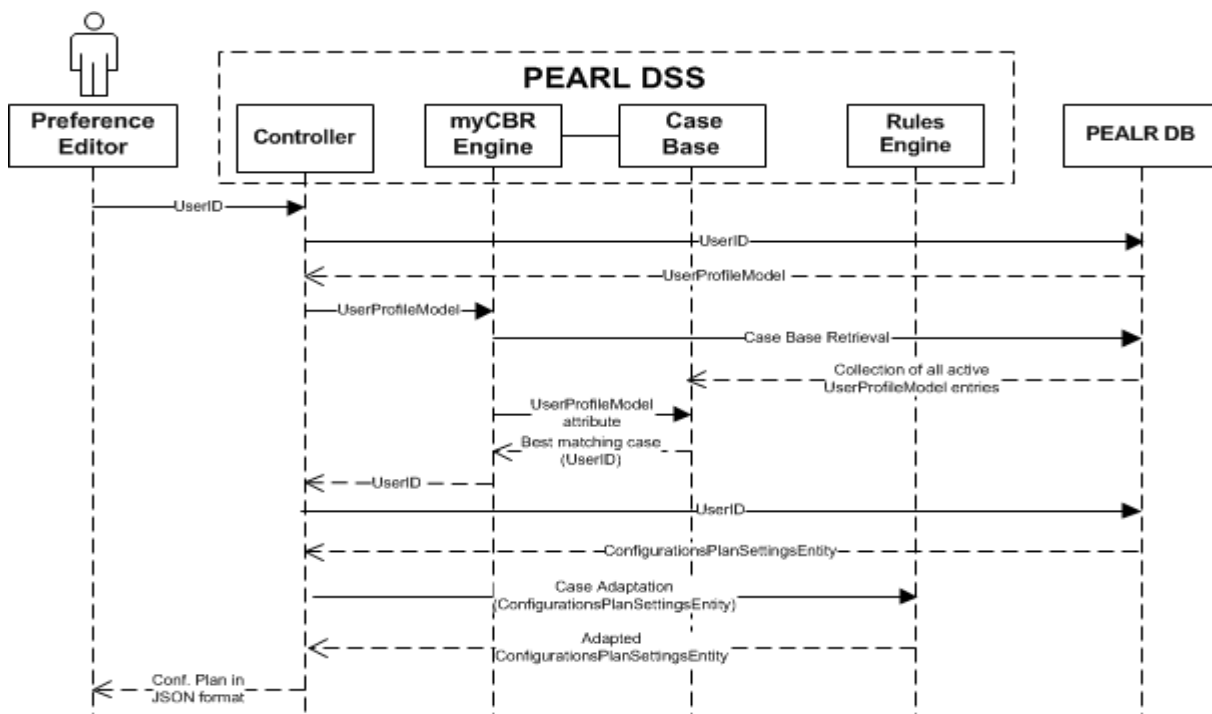


Figure 3. DSS-based configurations retrieval

### 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, as well as a 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 had implemented a manual adaptation; whereas a semi-automatic rule-based adaptation process was fully integrated in the final prototype and will be described in details in sections 3.2 and 4.6. 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.

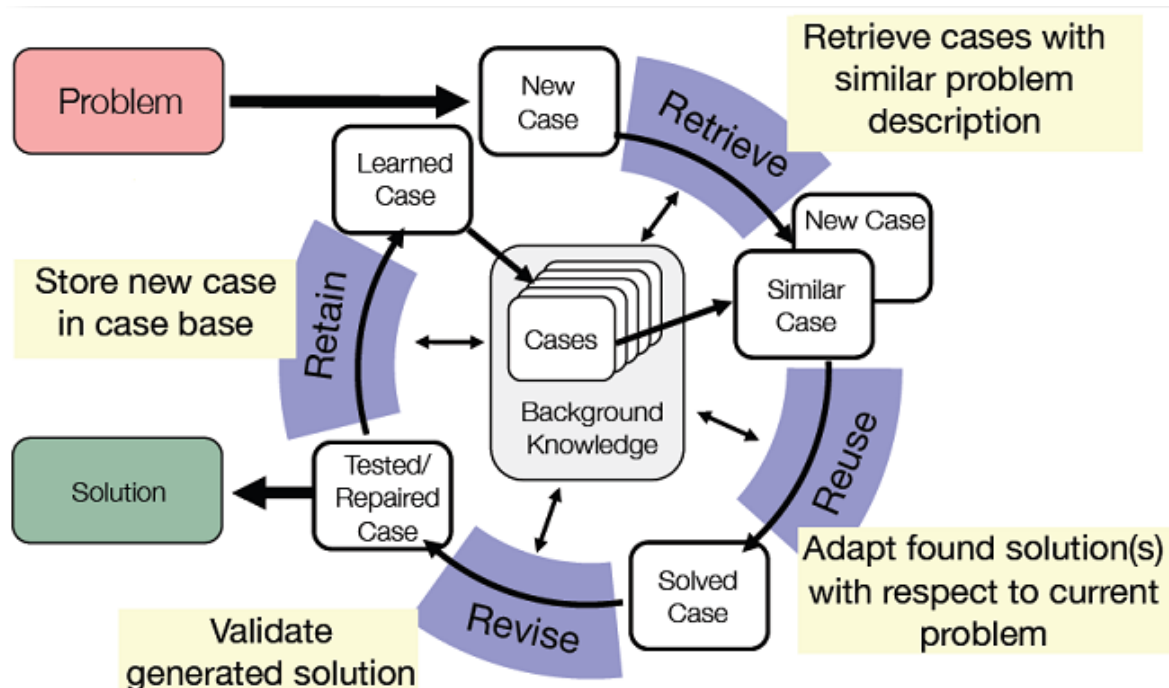


Figure 4. CBR reasoning cycle, [1]

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 was developed and 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 was implemented in the final prototype of PEARL DSS. The implemented rule-based systems takes as an input the configuration plan parameters produced by the CBR matchmaker and returns an adapted solution, based on a set of rules that account for the differences between the new and the already existing users. The rule base was developed with the use of the Drools native language, while the knowledge session core engine was based on the Drools Expert core module.

### 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 [10]. 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 [11]: 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 was used for the needs of PEARL.

Rule-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 them, 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 have integrated a rule-based system solely for the needs of adaptation of separate configuration parameters. The process of explicit definition of the adaptation rule will be described in section 4.6 below. While the software development techniques, used for the incorporation of the Drools rule engine into the first PEARL DSS prototype will be discussed in section 5.1.

### **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

## 4 Modelling of Users and Tasks in the Workspace

In the following section, the data structures that have been used for the implementation of the CBR matchmaker and the PEARL rule engine are described. Two main entities can be identified – the case base of the CBR-based DSS module and the knowledge base of the DSS rule-base adaptation module.

The case base of the CBR-based DSS module of the PEARL platform constitutes of a set of problem-solution pairs, where the problem part is defined by the available user characteristics, as specified by the User Profile Model, and the retrieved solution provides the Configuration Plan Model entity, available for this particular user. The User Profile Model contains information about the user's demographic, cognitive, physical and professional characteristics that is used by the CBR matchmaker for the 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 the user is involved in. The construction of the relevant models is described in *D4.2 User, Tasks and Workspace Databases, Ontologies and Knowledge Bases*. Here, we focus on the incorporation in the CBR engine and the corresponding similarity measures used in the process of case retrieval.

The knowledge base of the Drools rule engine contains a set of rules that ensures the provision of a fully customized configuration plan model even in cases of closely matching user profile, retrieved by the CBR matchmaker. We have operated under the assumption that even closely matching profiles can still differ in some key parameters and therefore some case adaptation logic is needed. The rules have been defined in the Drools native language and are described in section 4.6 below.

### 4.1 User Profile Model

We will start by defining the data models that constitute the problem solution pairs of the CBR case base, namely the *UserProfileModel* (the problem) and the *ConfigurationsSettingsEntity* model (the solution). The User Profile Model concept includes a range of demographic, cognitive, physical and professional characteristics of 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
<b>abilityToAdapt</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>abilityToLearnIndependently</b>	String	Symbol	good, medium, bad	Similarity Table
<b>anxiousnessICT</b>	String	Symbol	very_high, high, medium, low, very_low	Similarity Table
<b>arthritisInHands</b>	String	Symbol	none, medium, severe	Similarity Table
<b>attentionAbility</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>backPain</b>	String	Symbol	none, medium, severe	Similarity Table

<b>colourPerception</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>communicationSkill</b>	String	Symbol	none, listening, reading, speaking, writing	Similarity Table
<b>entrepreneurialSkills</b>	String	Symbol	good, medium, bad	Similarity Table
<b>fieldOfVision</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>flexibility</b>	String	Symbol	high, low, medium	Similarity Table
<b>gender</b>	String	Symbol	Male, female	Similarity Table
<b>handEyeCoordination</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>hearingAbility</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>height</b>	Integer	Integer	$50 \leq \text{height} \leq 250$	Polynomial similarity difference function
<b>knowledgeNavigationSkills</b>	String	Symbol	good, medium, bad	Similarity Table
<b>languageProduction</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>languageReception</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>literacyICT</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>longTermMemory</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>lungProblems</b>	String	Symbol	none, medium, severe	Similarity Table
<b>problemSolvingSkills</b>	String	Symbol	good, medium, bad	Similarity Table
<b>processingSpeed</b>	String	Symbol	good, medium, bad	Similarity Table
<b>setConfigurations</b>	Integer	Integer	0,1	-
<b>shouldersPain</b>	String	Symbol	none, medium, severe	Similarity Table
<b>socialSkills</b>	String	Symbol	good, medium, bad	Similarity Table
<b>teamworkSkills</b>	String	Symbol	good, medium, bad	Similarity Table
<b>testSite</b>	String	Symbol	Netherlands, Romania, Switzerland	Similarity Table
<b>understandingSigns</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>visualSensitivity</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>weight</b>	Integer	Integer	$30 \leq \text{weight} \leq 300$	Polynomial similarity difference function
<b>workingMemory</b>	String	Symbol	very_good, good, medium, bad, very_bad	Similarity Table
<b>yearBirth</b>	Integer	Integer	$1940 \leq \text{yearBirth} \leq 2000$	Smooth-step similarity difference function
<b>yearsEmployed</b>	Float	Float	$0 \leq \text{yearsEmployed} \leq 50.0$	Polynomial similarity difference function

## 4.2 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. There is only one small change as compared to the first prototype version of the DSS: the light temperature can now be set separately for the room light and desk light, thus providing greater configurations flexibility. The change is indicated in table 2 below.

**Table 2.** Configuration plan model attributes

Name	Data Type	Available Values
<b>General Preferences</b>		
<i>PC Configuration</i>		
<b>fontSize</b>	String	Small, Medium, Large
<b>soundLevel</b>	String	Mute, Low, Medium, High
<b>screenResolution</b>	String	Low, Medium, High
<i>Ambient Configurations</i>		
<b>roomLightIntensity</b>	String	Off, Low, Medium, High
<b>deskLightIntensity</b>	String	Off, Low, Medium, High
<b>roomLightTemperature</b>	String	Warm, Medium, Cold
<b>deskLightTemperature</b>	String	Warm, Medium, Cold
<i>Login Task</i>		
<b>defaultTask</b>	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
<b>Task Modelling</b>		
<i>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</i>		
<b>preferredSoftware</b>	String	<i>The available software product name, related to each of the tasks.</i>
<b>availabilityIndicator</b>	String	Blue, Green, Orange, Red
<b>taskWorkspace</b>	Integer	0,1
<b>fontSize</b>	String	Small, Medium, Large
<b>soundLevel</b>	String	Mute, Low, Medium, High
<b>screenResolution</b>	String	Low, Medium, High
<b>roomLightIntensity</b>	String	Off, Low, Medium, High
<b>deskLightIntensity</b>	String	Off, Low, Medium, High
<b>roomLightTemperature</b>	String	Warm, Medium, Cold
<b>deskLightTemperature</b>	String	Warm, Medium, Cold
<b>Physical Wellbeing Module Settings</b>		
<b>setExerciseNotifications</b>	String	yes, no
<b>exerciseGoal</b>	Integer	the integer value in minutes
<b>shortenSittingPeriods</b>	String	yes, no
<b>maxSittingDuration</b>	Integer	the integer value in minutes
<b>setRelaxingExercises</b>	String	yes, no
<b>setAgendaItems</b>	String	yes, no
<b>coveredPeriod</b>	String	working_hours, work_week, always
<b>typeOfExercises</b>	String	Individual_exercises, team_exercises, mix
<b>eLearning Module Settings</b>		
<b>selectedCourses</b>	Array	<i>List of available courses</i>
<b>setElearningNotifications</b>	String	yes, no
<b>sessionsPerWeek</b>	Integer	Integer value between 0 and 5
<b>Cognitive Games Module Settings</b>		
<b>gameCategory</b>	String	attentionGames, memoryGames, reasoningGames, logicGames, orientationGames,

		languageGames, constructiveGames
<b>gameDifficulty</b>	String	easy, normal, hard
<b>Task and Time Management Module Settings</b>		
<b>taskCategories</b>	Array	<i>List of selected tasks</i>
<b>agendaCategories</b>	Array	List of agenda categories

### 4.3 Example instantiations and case base building

After building the vocabulary in terms of the User Profile Model concept and its attributes and defining the appropriate similarity measures (described in section 4.5 below), 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. The initial examples are presented in Appendix A. The first version of the PEARL DSS, being a proof of concept prototype, was operating only on the basis of this example instantiations.

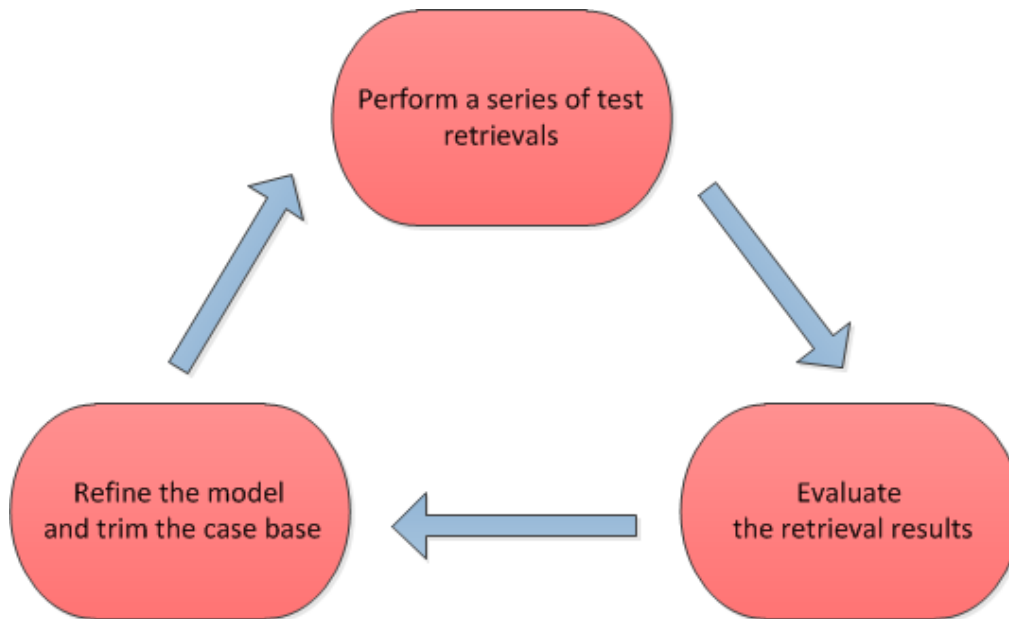
Naturally, in order to improve the system's accuracy the generation of more substantial case base was needed. Here, it is important to note that the PEARL DSS matchmaker is designed to dynamically create the case base by retrieving the most recent user information stored in the database. This gives two 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.

On the basis of the above mentioned functionality, the case base was extended in two major iterations. The initial case base extension was realized within the consortium by asking the partners to test the platform's UI and functionalities and create complete user profiles, including both user profile preferences and the corresponding configurations settings. These profiles automatically became part of the CBR matchmaker's case base. It is important to note that a user model – configurations settings pairs were included in the case base only after the user has successfully completed the user profile section, the general settings section and the corresponding sections for the separate tasks. The iteration of the case base extension was realized during the platforms initial lab trials. Further information about the number of participants and the lab trial conditions can be found in the corresponding report of the performed lab trials (D2.5.1 Report on First Lab trials).

After the completion of the lab trials and examination and refinement of the extended case base was performed and the similarity measures definition was redefined in order to improve the case retrieval process. The process is described in section 4.4 below.

### 4.4 Knowledge refinement process

In order to improve the performance of the PEARL CBR matchmaker module and enhance the accuracy of the retrieved results the CBR knowledge model had to undergo a process of refinement, presented in figure 5 below.



**Figure 5.** CBR knowledge refinement process, [6]

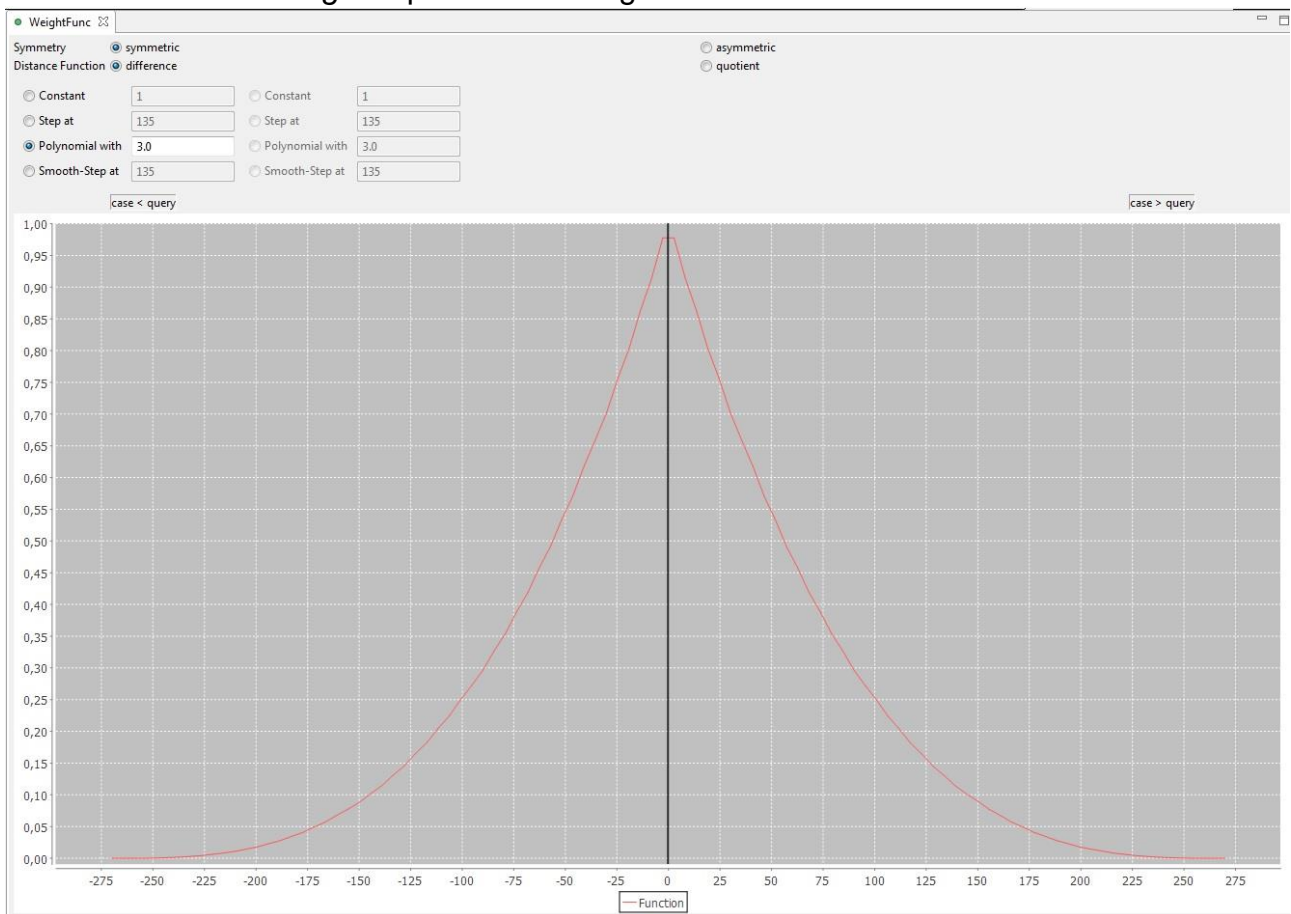
In order to test the matchmaker's performance and accuracy the case base was extracted from the PEARL database and imported to myCBR Workbench tool. Twenty test profiles were created and entered in the system, varying the age, the physical and mental characteristics. The retrieval results were evaluated and the knowledge model was refined in several steps. The first step was to reexamine the UserProfile concept and look for attributes that might be considered irrelevant to the retrieved solution. Although it was decided to keep the original model's attribute unchanged, certain value ranges were reduced. Manipulating the case base, which was built during the lab trials, was the next step in the knowledge refinement process. We have looked for poorly defined or incomplete user profiles and we either removed, or edited them in order to offer tangible solutions. The last step was to redefine the knowledge model's similarity measures in order to produce more accurate case retrieval. The main changes that were implemented were related to the global similarity function and will be described in section 4.5 below.

#### 4.5 Similarity Measures and myCBR Workbench

The similarity measures were initially defined for the first version of the PEARL DSS and were later reexamined after the PEARL lab trials. Since the local similarity measures were fairly straightforward, the main modifications focused on the weighted global similarity function that is described towards the end of this section.

We start by defining the general principle of the similarity-based retrieval applied in the PEARL project and we describe the local similarity functions that have been utilized. When a new user enters his/her 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. 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 6.



**Figure 6.** Polynomial similarity decrease function for the *weight* attribute

As shown on figure 6, 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 7.

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 7.** 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. Similar tables were defined for symbolic attributes used in the current project.

The global similarity function for the User Profile Model concept was 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, following a heuristic approach. The weights were readjusted after the completion of the PEARL lab trials following the steps described in section 4.4. The interrelation between the input user profile model parameters and the output configuration plan parameters was examined heuristically in order to determine a set of parameters that should carry a greater weight. Given the fact that the workspace configuration settings that are controlled by the PEARL platform are mainly related to the physical capabilities of the end user, the weight for the parameters such as visual acuity and sensitivity, colour perception, field of vision and hearing ability were increased. The validity of the new measures was further tested with a set of test user profiles as indicated in section 4.4. The resulting values are presented in table 3 below.

**Table 3.** Global similarity weighted function attributes

Attribute	Discriminant	Weight
abilitytoadapt	true	1.0
abilitytolearnin...	true	1.0
anxiousness	true	2.0
arthritisinhands	true	1.0
attentionability	true	1.0
backpain	true	1.0
colourpercepti...	true	4.0
communicatio...	true	1.0
entrepreneurial...	true	1.0
fieldofvision	true	4.0
flexibility	true	1.0
gender	true	3.0
handeyecoordi...	true	2.0
hearingability	true	4.0
height	true	1.0
knowledgenavi...	true	1.0
languageprodu...	true	1.0
languagerecep...	true	1.0
literacy	true	2.0
longtermmem...	true	1.0
lungproblems	true	1.0
problemsolvin...	true	1.0
processingspeed	true	2.0
setConfigurati...	true	1.0
shoulderspain	true	1.0
socialskills	true	1.0
teamworkskills	true	1.0
testsite	true	1.0
understanding...	true	1.0
user_id	true	1.0
visualsensitivity	true	4.0
weight	true	1.0
workingmemory	true	1.0
yearbirth	true	2.0
yearsemployed	true	1.0

## 4.6 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 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.

For the implementation of the adaptation logic and the definition of the formalized rules Drools business rules management system has been used. The rules can be defined as pieces of knowledge that can be expressed as WHEN-THEN structures. The rules were defined in the Drools rule language and have been stored as .drl files that are further retrieved and utilized by the Drools Expert core engine. The general syntax of a rule has been described in details in [8] and presented in figure 8 below:

```
rule
  "name"

  attributes

  when

  LHS

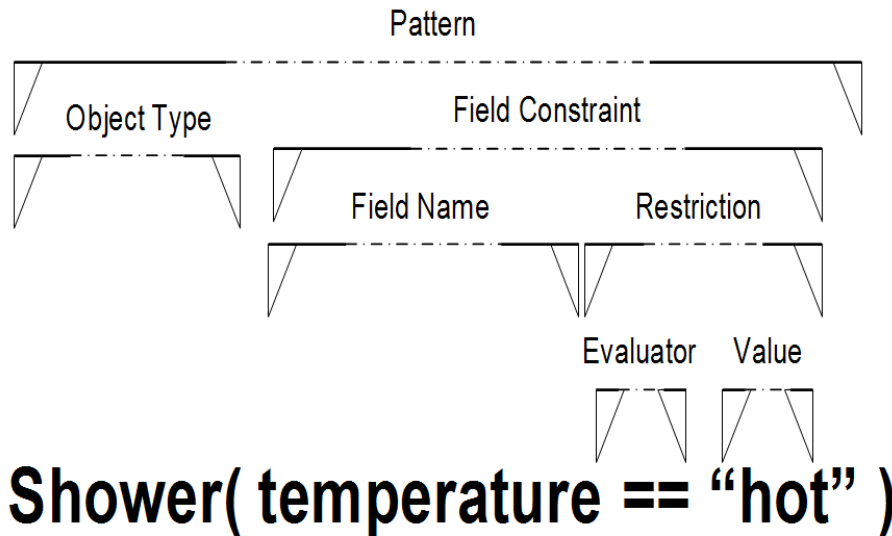
  then

  RHS

end
```

**Figure 8.** General syntax of Drools rules

Each rule consists of a rule name, optional attributes, condition and consequence. The rule name is a string value that describes the main purpose of the given rule and has to be unique in the given rule package. The attributes define the behavior parameters of the rule, such as dialect, salience, duration, etc. The left-hand side (LHS) is the conditional parts of the rule. It starts with *when* keyword and consists of zero or more conditional elements. If no condition elements are defined, the rule will be defined as always true and will be always activated, when a new sessions is created. The conditional elements operate on one or more patterns that will be matched to the facts inserted in the working memory. Each pattern defines zero or more constraints and can be bound to a pattern binding variable. The constraints are expressions that return true or false. The general pattern structure, matched to a specific example is presented on figure 9 below.



**Figure 9.** Drools rule pattern, [9]

The right-hand side (RHS) defines the consequence (or the action part) of the rule and contains the code that will be executed if the “WHEN” part is true. It is considered bad practice to include conditional statements in the RHS part. If there is such need then rule can probably be redefined and broken into multiple rules. The main purpose of the actions part is to modify working memory data by insert, delete or modify statements.

For the needs of PEARL a small set of rules has been defined as proof of concept of the DSS adaptation module design. Considering the subset of the workspace configurations settings that can be modified by the PEARL platform, the rules focus mainly on adjusting these parameters, based on the perceptual ability characteristics, indicated by the users. Four example rules, defined and implemented for the needs of PEARL, are presented in the following code snippet:



```

package dss.ruleengine

import dss.ruleengine.UserProfileModel;
import dss.ruleengine.ConfigurationSettingsEntity;
import dss.ruleengine.MapTaskWorkspaceSettings;

dialect "java"
global dss.ruleengine.ConfigurationSettingsEntity mySettings;

rule "Adjust for bad visual acuity and sensitivity"
  when
    inputProfile : UserProfileModel
  (visualAcuity=="bad" || visualAcuity=="very_bad")
  then
    mySettings.updateSingleParameter(0,"deskLightTemperature", "warm");
    mySettings.updateSingleParameter(0,"roomLightTemperature", "warm");
    mySettings.updateSingleParameter(0,"deskLight", "high");
    mySettings.updateSingleParameter(0,"fontSize", "large");
    mySettings.updateSingleParameter(0,"screenResolution", "medium");
  end

rule "Adjust for bad colour perception"
  when
    inputProfile : UserProfileModel
  (colourPerception=="bad" || colourPerception=="very_bad")
  then
    mySettings.updateSingleParameter(0,"deskLightTemperature", "cold");
    mySettings.updateSingleParameter(0,"roomLightTemperature", "cold");
    mySettings.updateSingleParameter(0,"roomLight", "high");
  end

rule "Adjust for bad field of vision"
  when
    inputProfile : UserProfileModel
  (fieldOfVision=="bad" || fieldOfVision=="very_bad")
  then
    mySettings.updateSingleParameter(0,"deskLight", "off");
    mySettings.updateSingleParameter(0,"roomLight", "high");
    mySettings.updateSingleParameter(0,"screenResolution", "medium");
  end

rule "Adjust for bad hearing ability"
  when
    inputProfile : UserProfileModel
  (hearingAbility=="bad" || hearingAbility=="very_bad")
  then
    mySettings.updateSingleParameter(0,"soundLevel", "high");
  end

```

The rules adjust different subsets of workspace configuration parameters and write the newly defined values to a global variable, which contains the configuration plan returned by the CBR-based matchmaker. The examples rules, presented above, are fired only when the user has indicated poor (*bad*, *very\_bad*) visual acuity and sensitivity, field of vision, colour perception or hearing ability.

## 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. Representational State Transfer (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 10 and the data flow is described in the form of a sequence diagram in figure 11.

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>
```

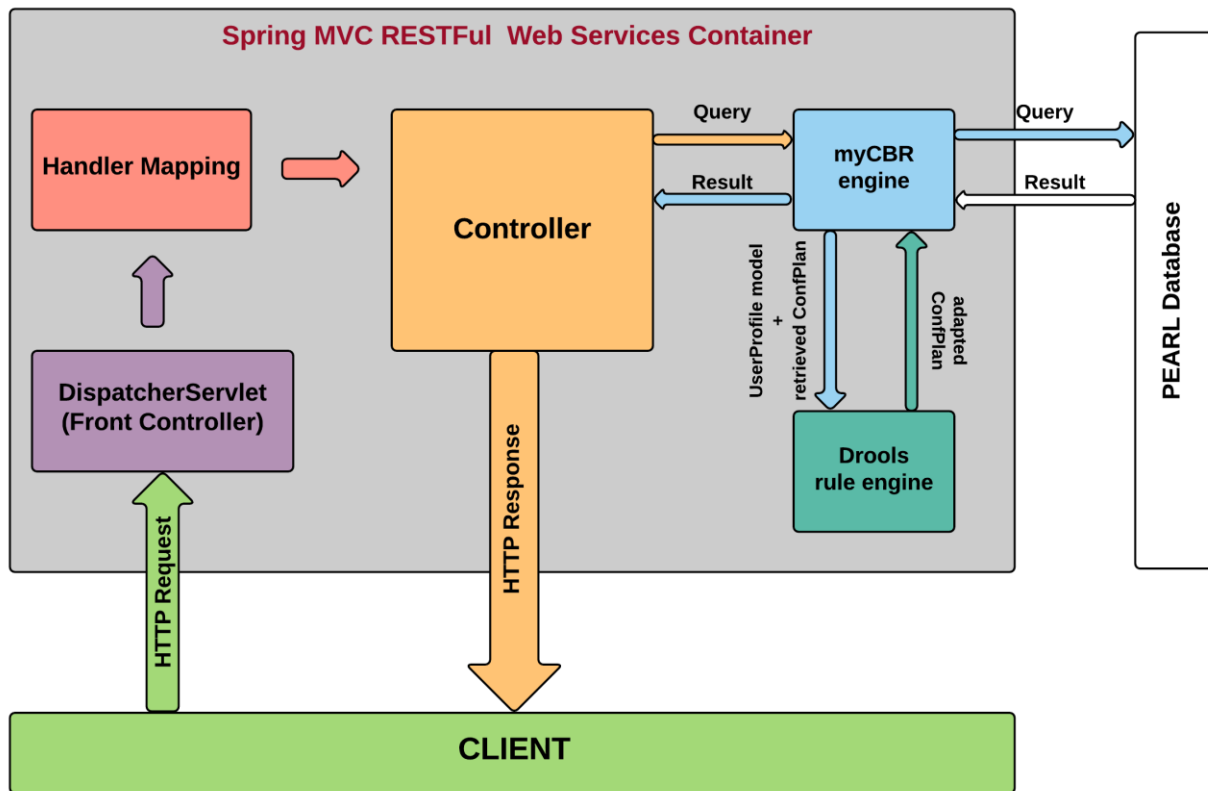


Figure 10. PEARL DSS architecture and principle of operation

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. The controller then retrieves the adapted configuration plan from the database. The best matching profile and the corresponding configurations settings entity are then passed as parameters to the

PEARL rule engine, which instantiates a new knowledge session and invokes the relevant rules from the knowledge base in order to adapt the best matching configuration plan to fit the personal needs of the new user. Finally, the controller 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);
        }
    }
}
```

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. **Drools rule engine** – PEARL DSS adaptation rule engine is implemented with the use of the Drools business logic integration platform. The general Drools dependencies were added to the project from the Maven repository. Drools core is defined as Knowledge Is Everything (KIE) and is instantiated in the *PearlRuleEngine* class and invoked by the controller. As every Drools based project, PEARL rule engine module consists of rules, facts and knowledge session. Drools rules are defined in Drools native language and are stored in the

*ConfSettings.drl* file. The rules definition was already described in details in *section 4.5 Adaptation Rules*. The facts are the data on which the rules will act upon. In the current scenario the facts are passed as input to the *PearlRuleEngine* class as input parameters in the form of an instance of the *UserProfileModel* class. The knowledge session is the core component that is responsible for firing all rules. In the current project it is bootstrapped as follows:

```
KieServices ks = KieServices.Factory.get();
KieContainer kContainer = ks.getKieClasspathContainer();
KieSession kSession = kContainer.newKieSession("ksession-rules");

...

kSession.insert(userProfile);
kSession.setGlobal("mySettings", settings);
kSession.fireAllRules();
```

The *KieSession* represents a running instance of the Rule Engine with a specific configuration and set of rules and is created from a *KieBase*. It also defines the evaluation algorithm that determines how to match the rules against the current set of domain objects. It is important to note that the *KieSession* and *KieBase* are defined declaratively in the *kmodule.xml* file, created in the *resources/META-INF* folder. The definition used in the current project is presented in the following code snippet:

```
<?xml version="1.0" encoding="UTF-8"?>
<kmodule xmlns="http://jboss.org/kie/6.0.0/kmodule">
  <kbase name="rules" packages="rules">
    <ksession name="ksession-rules"/>
  </kbase>
</kmodule>
```

Here a single *KieBase* is defined, which will compile all rules stored in the *rules* package, under the *resources* folder. This would be triggered by creating a *KieContainer*. Finally, *KieServices* is the interface, which provides access to all the Kie building and runtime facilities.

Once the knowledge session is defined, the facts that the rules engine will operate on has to be inserted. In the example above an instance of the *UserProfileModel* class is created and inserted to the session. Additionally, a global variable that holds the configuration plan retrieved by the CBR matchmaker is defined. This allows the rule engine to directly modify certain configuration parameters, based on the predefined rules. Finally, all rules are fired by calling *kSession.fireAllRules()*.

- 7. HTTP Response** – the controller returns the response directly to the client.

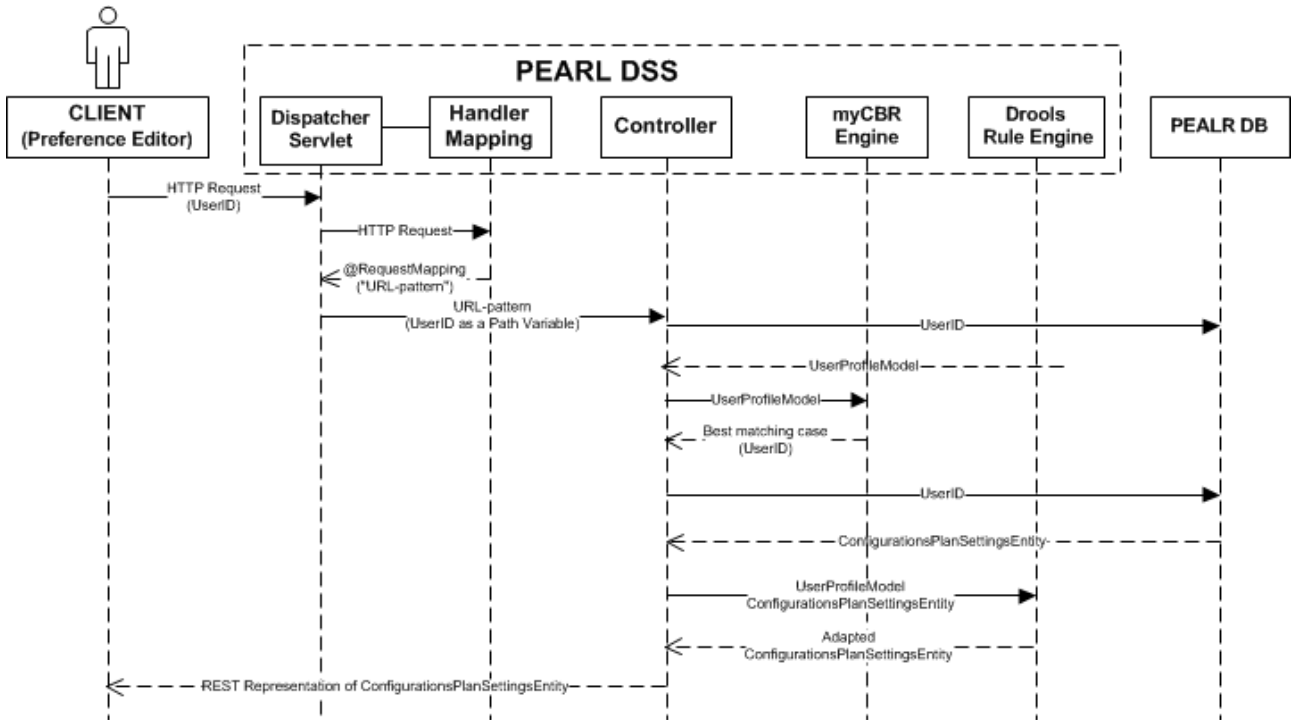


Figure 11. PEARL DSS REST service data flow sequence diagram

## 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

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

		<pre> "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": [] } </pre>	
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 through 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 and Deployment Details

### 6.1 System specifications

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
- Java Platform, Standard Edition: 1.8.0\_25
- listening port: 8081
- public endpoint: pearl-atl.euprojects.net

### 6.2 Dependencies and packaging

Dependency management and packaging in the current project was handled via Maven Integration Tool for Eclipse v1.7.0. The libraries and plug-ins that were used during the implementation of the prototype are presented in table 4 below.

Library / Plug-in	Version
<b>Dependencies</b>	
org.springframework.spring-context	4.1.7.RELEASE
org.springframework.spring-webmvc	4.1.7.RELEASE
org.aspectj.aspectjrt	1.7.3
org.slf4j.slf4j-api	1.7.12
org.slf4j.jcl-over-slf4j	1.7.12
org.slf4j.slf4j-log4j12	1.7.12
log4j	1.2.17
javax.inject	1
javax.servlet.servlet-api	2.5
javax.servlet.jsp.jsp-api	2.1
javax.servlet.jstl	1.2
com.fasterxml.jackson.core.jackson-core	2.6
com.fasterxml.jackson.core.jackson-annotations	2.6
com.fasterxml.jackson.core.jackson-databind	2.6
org.springframework.hateoas.spring-hateoas	0.17.0.RELEASE
com.fasterxml.classmate	1.2.0
org.hibernate.hibernate-validator	5.2.1.Final
org.jboss.logging.jboss-logging	3.3.0.Final
javax.validation.validation-api	1.1.0.Final
Mysql.mysql-connector-java	5.1.6
org.springframework.spring-jdbc	4.1.7.RELEASE
de.dfki.mycbr.core	3.1
de.dfki.mycbr.io	3.1
de.dfki.mycbr.util	3.1
org.kie.kie-api	6.4.0
org.drools.drools-core	6.4.0
org.drools.drools-decisiontables	6.4.0
<b>Plug-ins</b>	
maven-eclipse-plugin	2.9
org.apache.maven.plugins.maven-compiler-plugin	3.3

maven-war-plugin	2.6
org.eclipse.m2e. lifecycle-mapping	1.0.0

Spring Tool Suite v3.7.1.RELEASE was used in the development process and the packaging was realized with the use of *maven-war-plugin* via the IDE's user interface completing the following steps:

1. Maven clean
2. Maven generate-sources
3. Maven build... (setting *package* as a goal)

### 6.3 Installation

The installation process corresponds to the standard process for .war deployment on a Wildfly server. Below WILDFLY\_HOME indicates the root directory of the WildFly distribution. The required steps are as follows:

1. Copy the generated .war file to **\$WILDFLY\_HOME/standalone/deployment**
2. Execute the following command:  
**\$WILDFLY\_HOME/bin/jboss-cli.sh --connect --command="deploy --force [PATH\_TO\_WAR]"**
3. Test the deployment:  
[http://pearl-atl.euprojects.net/pearl\\_bil/](http://pearl-atl.euprojects.net/pearl_bil/)

## 7 Conclusions

Deliverable 4.4.2 *Decision Support System and Rule Engines* provided a comprehensive overview of the underlying artificial intelligence techniques, the architecture design, the principle of operation and the implementation logic of the final PEARL DSS module prototype. The DSS module can be divided into two major logical entities – the CBR-based matchmaker and the Drools-based adaptation rule engine.

The matchmaker was responsible for retrieving a matching user profile and its corresponding configurations parameters, based on a set of input user characteristics. CBR technique was selected as the basis of the matchmaker's implementation 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. The case base was initially built manually by creating generic user profiles, based on the input interviews during requirements elicitation process. The case base was further extended during the PEARL lab trials. The appropriate similarity measures model was defined initially heuristically and later, after the extension of the case base, was readjusted using a series of test retrievals, combined with systematic evaluation of the retrieval results.

The Drools-based adaptation rule engine is responsible for fine-tuning the best-matching solution, retrieved by the matchmaker, based on a set of predefined rules. Due to the ability of the platform to control a set of features, mainly related to the visual and audio characteristics of the workspace, the rules covered some obvious interrelations between the user physical characteristics and the corresponding workspace settings as a proof of concept.

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 that takes as an input the ID of the newly registered user and returns the customized configuration plan in JSON format. The implementation techniques and the details of the produced web service were described in details in order to provide a guide for future improvement or debugging.

## References

- [1] Maja Pantic, Introduction to Machine Learning and Case-Based Reasoning, Course Introduction, Imperial 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 Artificial 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](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.
- [8] Reference manual Drools 6.4.0.Final, "Chapter 8. Rule Language Reference," available at <http://docs.jboss.org/drools/release/6.4.0.Final/drools-docs/html/ch08.html#d0e7340>, last accessed 29/08/2016.
- [9] M. Proctor, "Drools – Business Logic Integration Platform," FOSDEM X, 2010.
- [10] C. Grosan and A. Abraham, "Rule-Based Expert Systems," in Intelligent Systems, Springer Berlin Heidelberg, 2011, pp. 149–185.
- [11] E. Turban and J. Aronson, Decision Support Systems and Intelligent Systems, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1997.

## Appendix A Example case instantiations

### A.1. User 1 – AIT

#### User Input Data

##### *a. User profile*

#### Demographics

- yearbirth(1958)
- gender (female)
- language (Romanian)

#### Physical Characteristics

- height (165)
- weight (55)
- backpain (none)
- shoulderspain (none)
- arthritisinhands (none)
- lungproblems (medium)
- flexibility (medium)

#### Occupation and competences

- testsite (Romania)
- jobposition (Country Manager)
- yearsemployed (15)

#### Underlying Skills

- literacy (medium)
- anxiousness (low)

#### Perceptual

- visualsensitivity (medium)
- colourperception (good)
- fieldofvision (good)
- hearingability (very\_good)

#### Cognitive

- languagereception (very\_good)
- languageproduction (very\_good)
- understandingsigns (good)
- attentionability (good)

#### Information Processing

- processingspeed (very\_good)
- workingmemory (medium)
- longtermmemory (very\_good)

- handeyecoordination (good)

### **b. Workspace Modeling**

#### Room

- size (big)
- colleagues (12)
- noiseLevel (loud)
- Artificial Light
  - ceilingLight (1)
  - deskLight (0)
- numberOfDesks (13)
- numberOfChairs (13)

#### Hardware

- availabilityIndicator (0)
- Internet (1)
  - internetSpeed (medium)
- External Monitor (1)
  - touchScreen (0)
  - screenSize (normal)
- speakers (0)
- rfidReader (0)
- microphone (1)
- webcam (1)
- deskPhone (1)

### **Configuration settings**

#### **a. General Preferences (general)**

##### **PC Configuration**

- fontSize: Medium
- soundLevel: Mute
- screenResolution: High

##### **Ambient Configuration**

- roomLight (if available): High
- deskLight (if available): Off
- roomLightTemperature: Cold
- deskLightTemperature: Cold

##### **Log in task**

- defaultTask (email)

### **b. Task Modeling**

#### email (1)

- preferredSoftware (MS\_Outlook)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

calendar (1)

- preferredSoftware (Outlook)
- availabilityIndicator (Available)
- taskWorkspace (0)

pcsearching (1)

- preferredSoftware (Windows\_Search)
- availabilityIndicator (InBetween)
- taskWorkspace (0)

websearching (1)

- preferredSoftware (Google\_Search)
- availabilityIndicator (Available)
- taskWorkspace (0)

textediting (1)

- preferredSoftware (MS\_Word)
- availabilityIndicator (InBetween)
- taskWorkspace (0)

spreadsheetediting (1)

- preferredSoftware (MS\_Excel)
- availabilityIndicator (InBetween)
- taskWorkspace (0)

presentationediting (1)

- preferredSoftware (MS\_Powerpoint)
- availabilityIndicator (InBetween)
- taskWorkspace (0)

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

- Conference Call (1)
  - preferredSoftware (MS\_Skype)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

webbrowsing (1)

- preferredSoftware (Firefox)
- Availability indicator (Available)
- taskWorkspace (0)

## A.2. User 2 –AIT

### 1. User Input Data

#### *User Modeling*

##### Demographics

- yearbirth (1957)
- gender (male)
- language (Romanian)

##### Physical Characteristics

- height (170)
- weight (65)
- backpain (medium)
- shoulderspain (none)
- arthritisinhands (none)
- lungproblems (medium)
- flexibility (medium)

##### Occupation and competences

- testsite (Romania)
- jobposition (Sales Assistant)
- yearsemployed (?)

##### Underlying Skills

- illiteracy (bad)
- anxiousness (high)

##### Perceptual

- visualsensitivity (good)
- colourperception (good)
- fieldofvision (good)
- hearingability (good)

##### Cognitive

- languagereception (very\_good)
- languageproduction (good)
- understanding\_abstract\_signs (good)
- attentionability (good)

##### Information Processing

- processingspeed (good)
- workingmemory (medium)
- longtermmemory (good)
- handeyecoordination (medium)



### ***a. Workspace Modeling***

#### Room

- size (small)
- colleagues (1)
- noiseLevel (normal)
- Artificial Light
  - ceilingLight (1)
  - deskLight (1)
- numberOfDesks (3)
- numberOfChairs (9)

#### Hardware

- availabilityIndicator (0)
- Internet (1)
  - internetSpeed (medium)
- External Monitor (1)
  - touchScreen (0)
  - screenSize (normal)
- speakers (0)
- rfidReader (0)
- microphone (0)
- webcam (0)
- deskPhone (1)

## **2. Configuration settings**

### ***a. General Preferences***

#### **PC Configuration**

- fontSize: Medium
- soundLevel: Mute
- screenResolution: Medium

#### **Ambient Configuration**

- roomLight: Low
- deskLight: Medium
- roomLightTemperature: Warm
- deskLightTemperature: Warm

#### **Log in task**

- defaultTask (not set/null)

### ***b. Task Modeling***

#### email (1)

- preferredSoftware(MS\_Outlook)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

#### calendar (1)

- preferredSoftware(MS\_Outlook)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

pcsearching (1)

- preferredSoftware(Windows\_Search)
- availabilityIndicator (InBetween)
- taskWorkspace (0)

webseraching (1)

- preferredSoftware(Google\_Search)
- availabilityIndicator (Available)
- taskWorkspace (0)

testediting (1)

- preferredSoftware(MS\_Word)
- availabilityIndicator (Available)
- taskWorkspace (1)
  - **PC Configuration**
    - soundLevel: Mute
  - **Ambient Configuration**
    - deskLight (if available): High

spreadsheetediting (1)

- preferredSoftware(MS\_Excel)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (1)
  - **PC Configuration**
    - soundLevel: Mute
  - **Ambient Configuration**
    - deskLight (if available): High

meeting (1)

- preferredSoftware(MS\_Skype)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (1)
  - **PC Configuration**
    - soundLevel: High
  - **Ambient Configuration**
    - roomLight (if available): High

webbrowsing (1)

- preferredSoftware(Firefox)
- availabilityIndicator (Available)
- taskWorkspace (0)

## A.3. User 3 - RRD

### 1. User Input Data

#### a. *User Modeling*

Demographics

- yearbirth (1954)
- gender (female)
- language (Dutch)

Physical Characteristics

- height (170)
- weight (70)
- backpain (none)
- shoulderspain (none)
- arthritisinhands (none)
- lungproblems (none)
- flexibility (none)

Occupation and competences

- testsite (Netherlands)
- jobposition (Management assistant)
- yearsemployed (7)

Underlying Skills

- literacy (bad)
- anxiousness (medium)

List of eLearning related skills & competences<sup>1</sup>

- communocationskill (none)
- abilitytolearnindependenty (good)
- socialskills - ethics, positive attitude, responsibility (good)
- teamworkskills - collaborative learning, networking (good)
- abilitytoadapt (medium)
- problemsolvingskills (medium)
- knowledgenavigationskills (medium)

Perceptual

- visualsensitivity (medium)
- colourperception (good)

---

<sup>1</sup> 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.

- fieldofvision (medium)
- hearingability (very good)

### Cognitive

- languagereception (good)
- languageproduction (good)
- understandingsigns (medium)
- attentionability (good)

### Information Processing

- processingspeed (medium)
- workingmemory (good)
- longtermmemory (good)
- handeyecoordination (good)

#### ***b. Workspace Modeling***

- size (medium)
- colleagues (1)
- noiseLevel (normal)
- Artificial Light
  - ceilingLight (1)
  - deskLight (0)
- numberOfDesks (2)
- numberOfChairs (2)

## **2. Configuration settings**

### ***a. General Preferences***

#### **PC Configuration**

- fontSize: Large
- soundLevel: Mute
- screenResolution: Medium

#### **Ambient Configuration**

- roomLight: Medium
- deskLight: Off
- roomLightTemperature: medium
- deskLightTemperature: medium

#### **Log in task**

- defaultTask (email)

#### ***b. Task Modeling***

#### **email (1)**

- preferredSoftware (MS\_Outlook)
- availabilityIndicator (InBetween)
- taskWorkspace (1)

➤ **PC Configuration**

- fontSize: **Large**
- soundLevel: **Mute**
- screenResolution: **Medium**

➤ **Ambient Configuration**

- roomLight: **Medium**
- deskLight: **Off**
- roomLightTemperature: **Medium**

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

- preferredSoftware (**Outlook**)
- availabilityIndicator (**InBetween**)
- taskWorkspace (**0**)

pcsearching (1)

- preferredSoftware (**Windows\_Search**)
- availabilityIndicator (**InBetween**)
- taskWorkspace (**0**)

websearching (1)

- preferredSoftware (**Google\_Search**)
- availabilityIndicator (**Available**)
- taskWorkspace (**0**)

textediting (1)

- preferredSoftware (**MS\_Word**)
- availabilityIndicator (**DoNotDisturb**)
- taskWorkspace (**0**)

spreadsheetsediting(1)

- preferredSoftware (**MS\_Excel**)
- availabilityIndicator (**Available**)
- taskWorkspace (**0**)

meeting (1)

- preferredSoftware (**MS\_Skype**)
- availabilityIndicator (**DoNotDisturb**)
- taskWorkspace (**0**)

webbrowsing(1)

- preferredSoftware (**Firefox**)
- availabilityIndicator (**Available**)
- taskWorkspace (**0**)

## A.4. User 4 – RRD

### 1. User Input Data a. *User Modeling*

#### Demographics

- yearbirth (1963)
- gender (Male)
- language (Dutch)

#### Physical Characteristics

- height (189)
- weight (78)
- backpain (none)
- shoulderspain (none)
- arthritisinhands (none)
- lungproblems (none)
- flexibility (none)

#### Occupation and competences

- testsite (Netherlands)
- jobposition (Senior researcher)
- yearsemployed (11)

#### Underlying Skills

- literacy (very\_good)
- anxiousness (very\_low)

List of eLearning related skills & competences<sup>2</sup>

- communicationskill (none)
- abilitytolearnindependency (good)
- socialskills - ethics, positive attitude, responsibility (good)
- teamworkskills - collaborative learning, networking (medium)
- abilitytoadapt (good)
- problemsolvingskills(good)
- knowledgenavigationskills (good)

#### Perceptual

- visualsensitivity (very\_bad)
- colourperception (medium)
- fieldofvision (very\_bad)
- hearingability (good)

---

<sup>2</sup> 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

- languagereception (**very\_good**)
- languageproduction (**very\_good**)
- understandingsigns (**good**)
- attentionability (**good**)

### Information Processing

- processingspeed (**good**)
- workingmemory (**good**)
- longtermmemory (**good**)
- handeyecoordination (**medium**)

#### **a. Workspace Modeling**

### Room

- size (**small**)
- colleagues (**0**)
- noiseLevel (**low**)
- Artificial Light
  - ceilingLight (**1**)
  - deskLight (**0**)
- numberOfDesks (**1**)
- numberOfChairs (**2**)

### Hardware

- availabilityIndicator (**0**)
- Internet (**1**)
  - internetSpeed (**fast**)
- External Monitor (**1**)
  - touchScreen (**0**)
  - screenSize (**normal**)
- speakers (**1**)
- RFIDReader (**1**)
- microphone (**1**)
- webcam (**1**)
- deskPhone (**1**)

## **2. Configuration settings**

### **c. General Preferences**

#### **PC Configuration**

- fontSize: **Large**
- soundLevel: **Mute**
- screenResolution: **High**

#### **Ambient Configuration**

- roomLight: **High**
- deskLight : **Off**
- roomLightTemperature: **cold**
- deskLightTemperature: **cold**

**Log in task**

- defaultTask (email)

**d. Task Modeling**email (1)

- preferredSoftware (MS\_Outlook)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (1)
  - **PC Configuration**
    - fontSize: Large
    - soundLevel: Mute
    - screenResolution: High
  - **Ambient Configuration**
    - roomLight: High
    - deskLight: Off
    - roomLightTemperature: cold

calendar (1)

- preferredSoftware (MS\_Outlook)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

pcsearching (1)

- preferredSoftware (Windows\_Search)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

websearching (1)

- preferredSoftware (Google\_Search)
- availabilityIndicator (Available)
- taskWorkspace (0)

textediting (1)

- preferredSoftware (MS\_Word)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

spreadsheetsediting (1)

- preferredSoftware (MS\_Excel)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)

presentationediting (1)

- preferredSoftware (MS\_Powerpoint)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (0)



meeting (1)

- Conference Call (1)
  - preferredSoftware (MS\_Skype)
- availabilityIndicator (DoNotDisturb)
- taskWorkspace (1)
  - **PC Configuration**
    - fontSize: Large
    - soundLevel: Medium
    - screenResolution: High
  - **Ambient Configuration**
    - roomLight: Medium
    - deskLight: Off
    - roomLightTemperature: cold

webbrowsing (1)

- preferredSoftware (Firefox)
- availabilityIndicator (Available)
- taskWorkspace (0)