MY-AHA

Contract # 689592

My-AHA

Deliverable 6.11

DSS Platform II

Editor:	USI
Deliverable nature:	DEM
Dissemination level: (Confidentiality)	PU
Contractual delivery date:	M24
Actual delivery date:	M25
Suggested readers:	Risk model creators, Software developers and architectures
Version:	1
Total number of pages:	31
Keywords:	DSS, Risk model, Risk factor, UML, Middleware

Abstract

This deliverable describes the software package Decision Support System (DSS) developed by R as the implementation of the risk models described in the WP3. The present document contains the software model description and the description of the classes and modulus and their functionalities respectively. **This deliverable is the updated version of the deliverable D6.3 (DSS platform I).**

[End of abstract]

Disclaimer

This document contains material, which is the copyright of certain MY-AHA consortium parties, and may not be reproduced or copied without permission.

In case of Public (PU):
All MY-AHA consortium parties have agreed to full publication of this document.

In case of Restricted to Programme (PP):
All MY-AHA consortium parties have agreed to make this document available on request to other framework programme participants.

In case of Restricted to Group (RE):
The information contained in this document is the proprietary confidential information of the MY-AHA consortium and may not be disclosed except in accordance with the consortium agreement. However, all MY-AHA consortium parties have agreed to make this document available to the whole group.

In case of Consortium confidential (CO):
The information contained in this document is the proprietary confidential information of the MY-AHA consortium and may not be disclosed except in accordance with the consortium agreement.

The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the MY-AHA consortium as a whole, nor a certain party of the MY-AHA consortium warrant that the information contained in this document is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using this information.

[Full project title] MY-AHA – my Active and Healthy **A**geing

[Short project title] MY-AHA

[Number and title of work-package] WP6 Data Fusion, Analytics and other Services

[Document title] DSS Platform II

[Editor: Name, Partner] Amir Tabatabaei, USI

[Work-package leader: Name, Partner] Marcin Grzegorzek, USI

Copyright notice

Executive summary

This document gives the technical specifications of the Decision Support System (DSS) used as the computational engine of the risk model specified in the WP3 in the deliverable D3.1.

In this document, the structure of the risk model including the classes and their names, specifications, attributes and operations will be described. The responsibilities and connections between the classes will be shortly described as well. The data types to be used in the attributes and methods are mentioned for further development and consistency with other building blocks for the sake of integration. The current structure is based on the available risk model described in D3.1 and further modifications would be reflected in the updates and the revisions of this report. The current demonstrator shows the calculation of the personalized risk model for a domain of example and generates the sample graph.

Also a Unified Modelling Language (UML) diagram of the software model will be depicted wherein the class model structures, the connection and responsibilities will be observed. Depicting such diagram will visualize the structure of the systems in an efficient way required for further analysis and extension.

It is worth to mention that the current DSS structure is an extendable platform flexible enough to be adapted further with the later development of the risk models and required data analysis. Such a capability has been provided in the current platform.

This deliverable is the updated version of the deliverable D6.3 (DSS platform I, M12). Some part of this document is appearing in the deliverable D6.9 due to strong synergy between them. Also the demonstrator is closely linked with the tasks represented in the deliverables D6.12 and D6.13. So the readers are encouraged to refer to the latter documents when reading the current deliverable.

List of authors

Company	Author
USI	Amir Tabatabaei, Philip Gouverneur
FhP	Perdo Madureira

Table of Contents

1 Introduction

In this document, the update on the Decision Support System (DSS) is given. The DSS is the computational engine of engaged risk model of My-AHA which is connected to the Middleware and communicate with it for fetching the user data from it and feeding the computation results and tailored plans to the Middleware. Currently the DSS is able to assess the risk adhering the risk models developed in WP3 based on the user's data in all domains and outcomes. General structure of the DSS is given in the following graph:

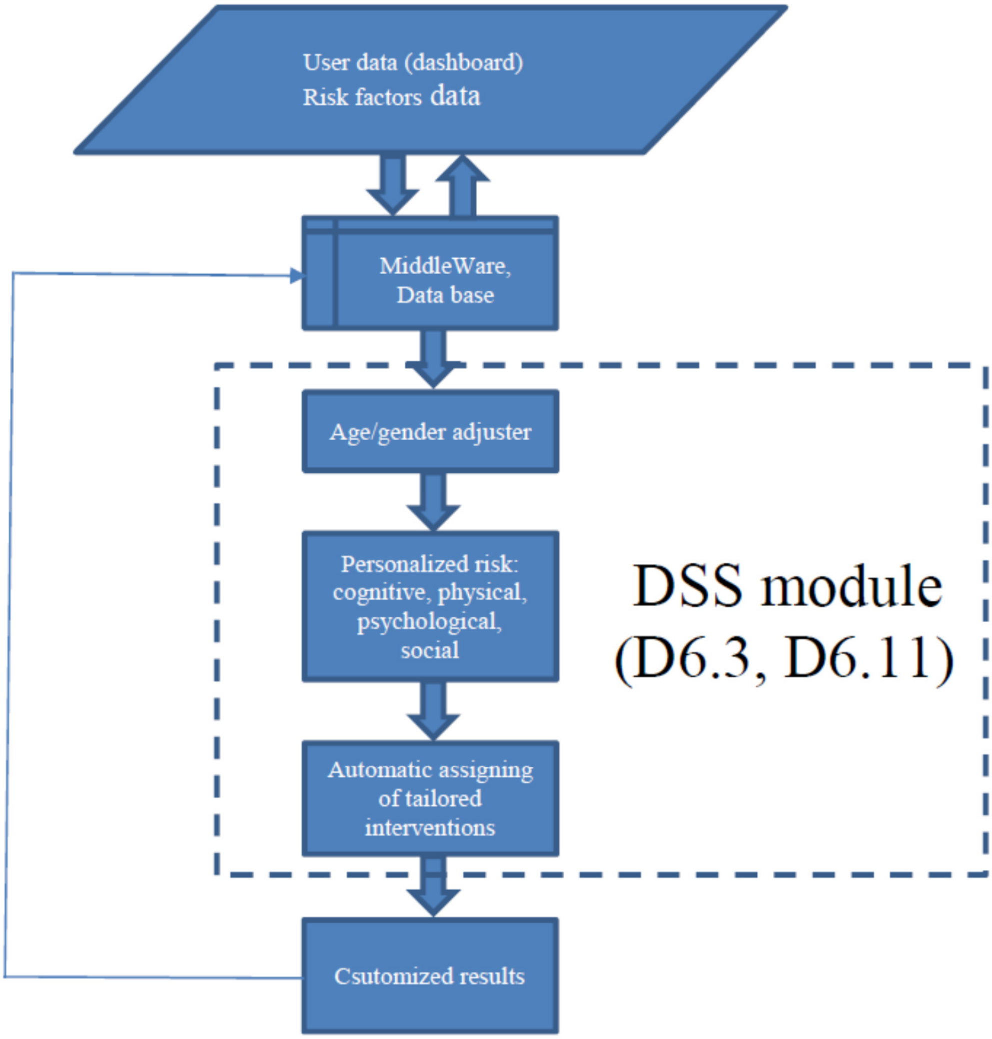


Figure 1. The general structure of the DSS in the My-AHA system

The data acquired by the primary and secondary users are read from the Middleware. The users' data is stored in the Middleware. The DSS calculates the risk values in six outcomes and assigns the customized interventions based on the assessed risks and/or other factors involved in other domains than risk domains. Here in this document the specification of the DSS is given. As per decision in the PMB meeting in M24 in Naples the assigning of the intervention is carried out by the PIs from the beginning of the RCT period based on the available list of the interventions. So this capability of the DSS will be deployed and used later on during the RCT and after that. Also it is worth noting that the current deliverable is closely linked with D6.9, D6.12 and D6.14 wherein the corresponding updates are considered as well.

2 Communication of the DSS with the Middleware (Update)

In the previous (initial) scheme, the communication between the DSS and MW has been based on the triggering the MW within a constant time window (default interval is 2 seconds) looking for any possible updates in the risk factors (in general measurements) and also user information. This method results in almost real time update in the risk assessment profile however it imposes un-necessary communicational load to the system. To tackle with this problem, a Java TopicListener interface has been developed by FhP and customized by USI to trigger the DSS only when it is needed according to any possible update (changes) in the measurements in real time. The following figure depicts the DSS-MW communication pipeline.

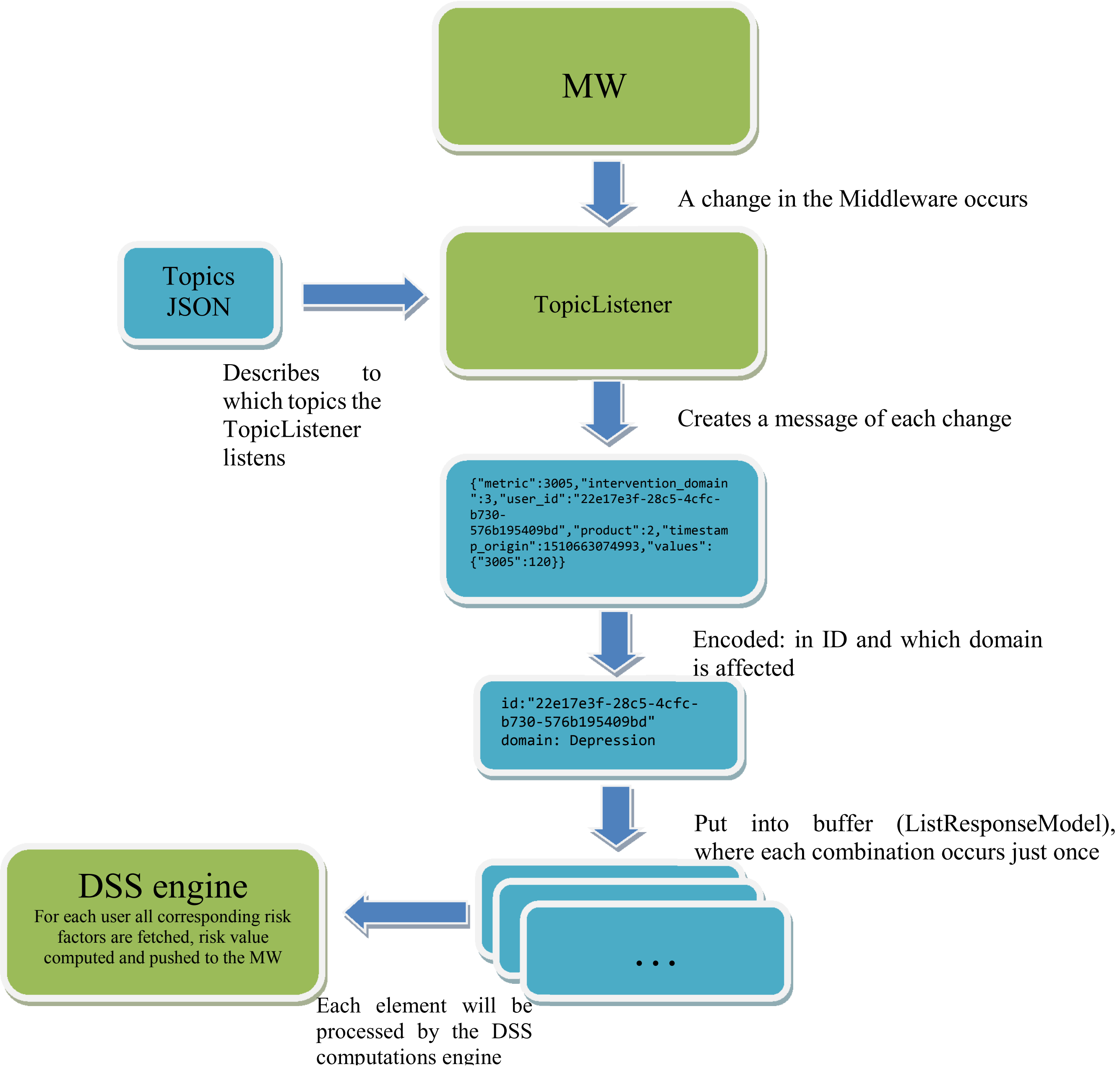

MW
A change in the Middleware occurs
Topics JSON
TopicListener
Describes to which topics the TopicListener listens
Creates a message of each change
{"metric":3005,"intervention_domain":3,"user_id":"22e17e3f-28c5-4cfc-b730-576b195409bd","product":2,"timestamp_origin":1510663074993,"values":{"3005":120}}
Encoded: in ID and which domain is affected
id:"22e17e3f-28c5-4cfc-b730-576b195409bd"
domain: Depression
Put into buffer (ListResponseModel), where each combination occurs just once
...
DSS engine
For each user all corresponding risk factors are fetched, risk value computed and pushed to the MW
Each element will be processed by the DSS computations engine

Figure 2. Updated DSS-MW communication scheme

The Java TopicListener uses a List of topics, which will be listened from the Middleware. Basically it includes all risk factors involved in the risk calculation in the risk model and the DSS as a measurements. The data is saved as JSON structure as follows:

```
[
  "body_weight",
  "height",
  "blood_pressure",
  "cholesterol",
  "diabetes",
  "depressive_symptoms",
  "self_perceived_health",
  "mmse",
  "living_condition",
  "history_of_falls",
  "cognitive_impairment",
  "prior_depression",
  "urinary_incontinence",
  "alcohol_problems",
  "smoking",
  "functional_disability",
  "men_early_retirement",
  "women_early_retirement",
  "colorectal_cancer_screening",
  "life_partner",
  "living_area",
  "household_size",
  "owning_pet",
  "life_enjoyment",
  "need_personal_care",
  "friends_relatives_contact",
  "poor_self_related_health",
  "comorbidities",
  "history_of_stroke",
  "hypotension",
  "gait_problems",
  "years_of_school"
]
```

Using this list, every risk factor in the Middleware is observed and a change will be called by a function. The function in the listener returns a message, like `{"metric":3005,"intervention_domain":3,"user_id":"22e17e3f-28c5-4cfc-b730-576b195409bd","product":2,"timestamp_origin":1510663074993,"values":  {"3005":120}}`. It contains all necessary information of a change in the Middleware. The necessary parameters for the DSS are the metric and the user id of the new value. With this information at first the domain of the metric is determined. Then a buffer with the domain and the user id is set up and filled. Each combination of values can exist only once, in this way unnecessary computation is avoided. For each element the DSS engine is called, which calculates the new risk value per domain/outcome. For this the needed values are fetched from the Middleware first. Then the risk values are computed and pushed back to the Middleware. Therefore the methods for reading and writing the Middleware are used, which are explained at the end.

In the following, the updated module in R used to update a risk value in an outcome is described. First, an admin authentication is performed and the administration token is retrieved. This method is described in detail later on. Next, the personal data of the user, which is the same for all domains is retrieved as well. Finally, the updated risk for each domain is calculated and pushed back to the Middleware. Furthermore, the server response for a logging file is returned. Since the code must be available always, we are using a try catch block and print some error message, if there is a problem. In the following the description for a general domain/outcome is shown for the sake of size and simplicity as the procedure is the same for all other domains and outcomes.

```
updateRisk = function(id, domain){
        tryCatch({

                        #authenticat as admin
                        token <<- authenticateAsAdmin();

                        # Retrieve personal data
                        personaldata = retrievePersonalData(token, id);

                        if (!grepl('Not all needed measurements are available', personaldata))
                                {
                                risk = "Not calculated";

                                stamp = currentTimestamp();

                                if(domain=="Frailty"){
                                        # Retrieve risk factors
                                        measurements = retrieveLastMeasurements
                                        (token, id, c(5002, 3010, 3011, 5003));
                                        risk = calculateFrailtyRisk(personaldata, measurements);
                                        data = sprintf('{"7002": %s}', risk);
                                        serverresponse =
                                        mwAddMeasurement(id, "7", "7", "7002", stamp, data);
                                }
                                else if[ other domains]
                                }
                                else
                                {
                                        risk = 'Wrong domain Name!';
                                        serverresponse = FALSE;
                                }
                        }
                        else{
                                risk = 'Not all needed personal data is available.';
                                serverresponse = FALSE;
                        }

                        return(serverresponse);

                }, error = function(error_condition) {
                        closeAllConnections();
                        sink('Errorfile.txt', append = TRUE);
                        cat("Error:\n")
                        message(error_condition)
                        print(error_condition)
                        return(FALSE)
                }, finally={
                        closeAllConnections();
                })
}
```

In the update process the function ‘retrievePersonalData’ is called. This function collects all personal data, which is the gender, the date of birth, diabetes and the educational years, for one user id as they are not stored in the Middleware as measurements. Therefore an administration token is needed. To the retrieve the personal data, the user profile is fetched from the Middleware. Next the data is put into a data array and given back. If there is no diabetes or years of education, the value ‘Unknown’ is given back. In the following you can find the procedure in detail.

```
retrievePersonalData = function(token, id)
# Retrieves the personal data for a given user id, an administration token is
needed
# Inputs: administration token, user id
# Returns: c(gender, date_of_birth, diabetes, years)
{
      #--------- Use of GetUser Function
      element2 = mwGetUser(token, id)

      #-------- Extract gender
      gender = element2["gender"];
      gender = gender[[1]];

      #-------- Extract date_of_birth
      date_of_birth = element2["date_of_birth"];
      date_of_birth = date_of_birth[[1]];

      #-------- Extract diabetes
      diabetes = element2["diabetes"];
      diabetes = diabetes[[1]];

      #-------- Extract educational years
      years = element2["education_level"];
      years = years[[1]];

      if (length(diabetes)==0)
            diabetes = "Unknown";
      if (length(years)==0)
            years = "Unknown";

      return(c(gender, date_of_birth, diabetes, years))
}
```

In above, the function ‘retrieveLastMeasurements’ returns a list of last measurements for a given list of metrics for a user. Again an administration token for the procedure is required. Therefore, for every element in metrics, a search of the last measurement is done. If there is a value, it is put into the return list; otherwise the string ‘Unknown’ is put into the list. At the end of the procedure, the return list is given back.

```
retrieveLastMeasurements = function(token, id, metrics)
# Function to retrieve the last measurements of different metrics
# Inputs: token of the user, id of the user, list of metrics to retrieve the
measurements
# Returns: list of measurements
{
      measurements <- vector();
      for(i in metrics){

            metric = sprintf('[%s]', i);

            response = mwGetLastMeasurement(token, id, metric , 1);

            if (!grepl('no measurement', response[1]))
            {
                  newvalue = response[,"values"];
                  newvalue = newvalue[1,1];
                  measurements = append(measurements, newvalue)
            }
            else{
                  measurements = append(measurements,"Unknown")
            }
      }
      return(measurements)
}
```

The details of Middleware specification including the methods for reading and writing the data from/to the Middleware are given in deliverable D4.6. The first step towards getting access to the measurement is authentication. The authentication can be done in two different levels including user and admin level and the token access can be exploited. The following part shows the aforementioned step and is a duplicate from D6.4 for the ease of the readers.

1. AUTHENTICATION

The authentication link and request are assigned as follows.

```
authenticationURL <- "https://myaha-mw.ddns.net:8443/auth/realms/myaha/protocol/openid-connect/token"
response <- POST(authenticationURL, body = user , encode = c("form"))
response <- POST(authenticationURL, body = admin , encode = c("form"))
```

Where the credential of the user or admin must be granted from Middleware side as example:

```
user <- list(username= "newusr",
                                    password          = "123",
                                    client_id         = "dashboard",
                                    client_secret  = "79313b56-e6c6-4377-855e-9a50c95f5586",
                                    grant_type        = "password")
admin <- list(username = "rui.neves",
                                    password          = "123",
                                    client_id         = "dashboard",
                                    client_secret  = "79313b56-e6c6-4377-855e-9a50c95f5586",
                                        grant_type                  = "password")
```

The possible successful reply regarding to above credential would be as follows:

```
> response
Response [https://myaha-mw.ddns.net:8443/auth/realms/myaha/protocol/openid-connect/token]
  Date: 2017-01-06 19:55
  Status: 200
  Content-Type: application/json
  Size: 3.64 kB
```

The token can be extracted with the following request:

```
element = jsonlite::fromJSON(content(responsea, "text", type = "parse_query"))
```

and a possible successful reply would be:

```
> element
$access_token
[1] "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJNenJfZmI5OVJSQXd1Vi1ZaUY1el9PUW9oMV9YZ
HdJeGFBY1RXM00zcTlBIn0.eyJqdGkiOiJhMWU1ZDI3Yy1hNDI0LTRkNjgtYWM2ZS04YzM2YTZmNzNmNDgiL
CJleHAiOjE0ODM3MzQzMjcsIm5iZiI6MCwiaWF0IjoxNDgzNzMyNTI3LCJpc3MiOiJodHRwczovL215YWhhLW1
3LmRkbnMubmV0Ojg0NDMvYXV0aC9yZWFsbXMvbXlhaGEiLCJhdWQiOiJkYXNoYm9hcmQiLCJzdWIiOiJlM2
M3Mzg3MC0wMjk0LTQ5MGYtOGE2NC0wMDkyN2ZjYjkxODUiLCJ0eXAiOiJCZWFyZXIiLCJhenAiOiJkYXNo
Ym9hcmQiLCJhdXRoX3RpbWUiOjAsInNlc3Npb25fc3RhdGUiOiIwOWM3NGJjNC0wMTI4LTRkMjItODFjNi02Zj
k2MWQ3ODJiNmEiLCJhY3IiOiIxIiwiY2xpZW50X3Nlc3Npb24iOiJmYmIwNjRmNy1jMDI0LTQ3ZjMtODQ2Yi1j
ODlhODZiOTMwNGIiLCJhbGxvd2VkLW9yaWdpbnMiOlsiKiJdLCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsidW1
hX2F1dGhvcml6YXRpb24iLCJ1c2VyIl19LCJyZXNvdXJjZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtY
W5hZ2UtYWNjb3VudCIsInZpZXctcHJvZmlsZSJdfX0sInJvbGVzIjoiW29mZmxpbmVfYWNjZXNzLCB1bWFfYXV
0aG9yaXphdGlvbiwgdXNlcl0iLCJuYW1lIjoicnIgcnIiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJuZXd1c3IiLCJnaX
Zlbl9uYW1lIjoicnIiLCJmYW1pbHlfbmFtZSI6InJyIn0.YKbF7Duug5lZOoN4vGENZd6FL0kBYzfGTQHKZZ_nm6a9
mY2FUAH4kYknXcigGmMRol38n2R14IODpvnpj_RMnKIdDxGCNCoj6bIib7nsmyrGMidKatIBh3iczWuVAQx8cNJ
AWOiJe_d8_YaNoZBkweUejNK7cnsW_j5tPFXMXHwDCOjM5qfdLTFEEq9zTn2NnT-CFbWZxJoiSUBUhZ_D2S3
hr0fsT5t8uR7HOmqKesI8i5_WICbd1LXc4kDw5jniz_VNSVINUv2vJHnH6x731W806JgyqmTgRQXBH6GV_VUeK
bl_otqU926GFWRUbAkszMFNIAS3TXF2ELmAusoKTA"

$expires_in
[1] 1800

$refresh_expires_in
[1] 1800

$refresh_token
[1] "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJNenJfZmI5OVJSQXd1Vi1ZaUY1el9PUW9oMV9YZ
HdJeGFBY1RXM00zcTlBIn0.eyJqdGkiOiJjZmJiOWI0ZS1kYTEyLTRlNDEtOTkxZi05MWFiNDA5ZTQ3MmMiLCJ
leHAiOjE0ODM3MzQzMjcsIm5iZiI6MCwiaWF0IjoxNDgzNzMyNTI3LCJpc3MiOiJodHRwczovL215YWhhLW13L
mRkbnMubmV0Ojg0NDMvYXV0aC9yZWFsbXMvbXlhaGEiLCJhdWQiOiJkYXNoYm9hcmQiLCJzdWIiOiJlM2M3
Mzg3MC0wMjk0LTQ5MGYtOGE2NC0wMDkyN2ZjYjkxODUiLCJ0eXAiOiJSZWZyZXNoIiwiYXpwIjoiZGFzaGJ
vYXJkIiwiYXV0aF90aW1lIjowLCJzZXNzaW9uX3N0YXRlIjoiMDljNzRiYzQtMDEyOC00ZDIyLTgxYzYtNmY5Nj
FkNzgyYjZhIiwiY2xpZW50X3Nlc3Npb24iOiJmYmIwNjRmNy1jMDI0LTQ3ZjMtODQ2Yi1jODlhODZiOTMwNGIi
LCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsidW1hX2F1dGhvcml6YXRpb24iLCJ1c2VyIl19LCJyZXNvdXJjZV9h
Y2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsInZpZXctcHJvZmlsZSJdfX19.KkCEdn
i67UBnfScoikX_gQmLsHPz01pGWOSG2VXBrxVqORQnGGUQWI4jp1yqySotqImao14nVYTpagMAI6LNIqJza6rH
mTj8QDTgiHF7oEe8rD-e1BkP8DG3OS7oZPfBseXnMzOAmwlNde7Sg0oVa0M1M6xslPMRTF_2p0MNlQTXRp6W
UwJFw7O8xoX-6J1e_lz-iUjGUBZTa-z_2A_1Stdi2o8-a8EUvYLFD38ipS1PXSt9IulS-q99jW3RNegnRc12EoHghXAZ
KD3df7ubGaCW2XyQy87ozBqDkogAn6DQ6Cpkq8EOHaql0hiNy0rSPmgQKw5O-qz1LGz6M6UBkw"

$token_type
[1] "bearer"
```

```
$id_token
[1] "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJNenJfZmI5OVJSQXd1Vi1ZaUY1el9PUW9oMV9YZ
HdJeGFBY1RXM00zcTlBIn0.eyJqdGkiOiJiYmFjNmU3ZC1iYzhkLTQ1YzctOTQyZS1iNGQ2N2Q3ODc2YTkiLCJle
HAiOjE0ODM3MzQzMjcsIm5iZiI6MCwiaWF0IjoxNDgzNzMyNTI3LCJpc3MiOiJodHRwczovL215YWhhLW13Lm
RkbnMubmV0Ojg0NDMvYXV0aC9yZWFsbXMvbXlhaGEiLCJhdWQiOiJkYXNoYm9hcmQiLCJzdWIiOiJlM2M3
Mzg3MC0wMjk0LTQ5MGYtOGE2NC0wMDkyN2ZjYjkxODUiLCJ0eXAiOiJJRCIsImF6cCI6ImRhc2hib2FyZCIsIm
F1dGhfdGltZSI6MCwic2Vzc2lvbl9zdGF0ZSI6IjA5Yzc0YmM0LTAxMjgtNGQyMi04MWM2LTZmOTYxZDc4MmI
2YSIsImFjciI6IjEiLCJuYW1lIjoicnIgcnIiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJuZXd1c3IiLCJnaXZlbl9uYW1lI
joicnIiLCJmYW1pbHlfbmFtZSI6InJyIn0.XWJR326J20wvOJeHNVrEXggRUfBzVt-cxguN5JgclCgSHAl0PdtkZqXGU
_dMFFGAiBTfBEKBeSG7Mx0yp81_m46VxWc1SiIIqbZ2ko_YqAQcrRoypAHiThduyZlKF3y7IOn3wluDF-jDDpK5
MO62k9qKV1AjhACobqR95PEkkNfJDGZ9z57qgG5yBZsPXOO4BQdrd-g6OuIsFMaghBghVeopBU2oZAgszn2Dx1
DyJ_bKIXM5AX0rkkjXlA8baa21R2yo0VPbYpacw_THkq2ooQCqnG0uJZQxlStXk4tTB2AOrk6KH5Xz7swZkE3DI0
V6YCHXQUjO5x2vDZV2aHnmlw"

$`not-before-policy`
[1] 1482325843

$session_state
[1] "09c74bc4-0128-4d22-81c6-6f961d782b6a"
```

2. GETTING USERS DATA

The user/admin link and request are assigned as follows.

getUserURL <- "https://myaha-mw.ddns.net:8445/api/rest/user/get/"

response <- GET(getUserURL, add_headers(Authorization = token))

where “token” has been extracted as described above (“token = element[[1]]”)

A possible valid response (user) would be as follows.

```
> element
$date_of_birth
[1] "16-11-1993"

$gender
[1] "Male"

$language
[1] "en"

$name
[1] "rr23"

$password
[1] "7ed4f8bb5c843528a5e3b184c09f63e4ff360d4f3c7677157e21cddeb5731a1f"

$surname
[1] "rr"

$user_id
[1] "e3c73870-0294-490f-8a64-00927fcb9185"

$username
[1] "newusr"
```

```
$monitoring
  smart_companion.username smart_companion.password
1             user_spulit
```

3. HANDLING USERS MEASUREMENTS

The users' measurement can be accessed through the following link and request:

```
getMeasurementURL <- https://myaha-mw.ddns.net:8445/api/rest/measurement/get/

response <- GET(getMeasurementURL, add_headers( Authorization = token))
```

As an admin, all of the existing measurements can be observed. A possible response would be in the following format:

```
> class(element)
[1] "data.frame"
> colnames( element )
[1] "_id"                "_rev"               "timestamp_arrival"
 [4] "timestamp_last_update"  "metric"            "intervention_domain_id"
[7] "user_id"             "product"            "timestamp_origin"
[10] "values"
```

As it is observed above, there are 10 fields which are corresponding with the user's measurements. The specification of each field is given in D4.6 in more details and can be observed using "head(element)" command.

3 Computation of the Personalized Risk Value (update)

3.1 Age/Gender adjuster

In this initial step, the prevalence data for each person and for each domain is calculated according to the extracted nonlinear weighted regression models in each domain. Please refer to subsection 3.3.1 of deliverable D6.9 for detailed description.

Remark: There is a major difference between the old version of the adjuster and the current one. The old adjuster model has been assigning the prevalence data versus age ranges in different intervals while in the current model the intervals are replaced by equations (per domain) in which the prevalence is calculated according to each equation. The equations are the result of aforementioned regression model.

3.2 Personalized risk per domain

The personalized risk value is computed based on the users data input fetched from the Middleware and the calculated prevalence within the following sequences. The detailed specification of each step can be read from deliverable D6.14, so to avoid repetition just the steps are mentioned here.

1. Age Gender Adjuster
2. Finding Pool Indices
3. Calculation of the Personal Risk Value
4. Calculation of the Personal Relative Risk
5. Calculation of the Prevalence
6. Converting the Probabilities to the Odd Ratios (OR)
7. Calculating the Posterior Odd Ratios (POR)
8. Calculating the Final Relative Risk Values in Different Domains

4 Risk Models - Inputs and Outputs (Update)

4.1 Description of the classes

The software system model includes different classes with 1 enumeration. In this subsection the description of the classes along with the responsibilities, the attributes and the methods will be given.

1. FrailtyRiskPool
- Brief description: Specific risk pool for the frailty which realizes the class 'RiskPool'.

2. FallRiskPool
- Brief description: Specific risk pool for the risk of fall which realizes the class 'RiskPool'.

3. DementiaRiskPool
- Brief description: Specific risk pool for the risk of dementia which realizes the class 'RiskPool'.

4. DepressionRiskPool
- Brief description: Specific risk pool for the risk of depression which realizes the class 'RiskPool'.

5. LonelinessRiskPool
- Brief description: Specific risk pool for the risk of loneliness which realizes the class 'RiskPool'.

6. SocialIsolationRiskPool
- Brief description: Specific risk pool for the risk of social isolation which realizes the class 'RiskPool'.

7. RisksPool
- Brief description: The given class describes a risk pool. The attribute and methods are described as below.

Attributes	Attributes Description	Visibility
PoolName: char [1..*]	This describes the name of the pool. Possible fields: {	Private

	physicalFrailtyPool, physicalFallsPool, cognitiveDementiaPool, psychologicalDepressionPool, socialLonelinessPool, socialSocialIsolationPool}	
Pool: list	For every pool specific risk factors are given that represent it. These factors are stored in a list. Example: physicalFrailtyRiskNames = c("FrailAge", "Gender", "DepressiveSymptoms", "PoorSelfRatedHealth", "Comorbidities", "CognitiveImpairment")	Private
AgeGenderAdjuster: Adjuster	This attribute is used to represents the prevalence of the outcome according to the person's age and calculated regression model per outcome.	Private

Methods	Methods Description	Visibility
getPoolName(object)	This method returns the name of the pool. object: “RisksPool”	Public
setPoolName(object, value)	This method changes the name of the pool object to value. object: “RisksPool”, value: “charcter”	Public
addRiskFactorToPool(object, value)	This method adds a risk factor value to the object. object: “RisksPool”, value: “RiskFactor”	Public
removeRiskFactorFromPool(object, index)	This method removes a risk factor from the pool. An index of the position in the list must be set. object: “RisksPool”, index: “numeric”	Public
getPool(object)	This method returns the pool/list of risk factors. object: “RisksPool”	Public
findRiskIndices(object, risknames)	This method returns the index of a risk factor.	Public

	object: “RisksPool”, riskNames: “character”	
calcRelativeRisk(object, age, gender, riskNames, listRiskArguments, isPresent, outcomename)	This method calculates the risk of the given pool and returns the value. object: “RisksPool” (e.g., physicalFrailtyPool, physicalFallsPool, cognitiveDementiaPool, psychologicalDepressionPool, socialLonelinessPool, socialSocialIsolationPool) age: “numeric” (age is calculated by subtracting the birth year from current year. gender: “Enumeration” (“Male”, “Female”, “Unknown”) riskNames: “character” (e.g., physicalFrailtyRiskNames, physicalFallsRiskNames, cognitiveDementiaRiskNames, psychologicalDepressionRiskNames, socialLonelinessRiskNames, socialSocialIsolationRiskNames) listRiskArguments: “list” (e.g., physicalFrailtyArguments, physicalFallsArguments, cognitiveDementiaArguments, psychologicalDepressionArguments, socialLonelinessArguments, socialSocialIsolationArguments) isPresent: “list” (e.g., physicalFrailtyIsPresent, physicalFallsIsPresent, cognitiveDementiaIsPresent, psychologicalDepressionIsPresent, socialLonelinessIsPresent, socialSocialIsolationIsPresent) outcomename: “list” (e.g., PhysicalFrailty, PhysicalFalls, CognitiveDementia, PsychologicalDepression, SocialLoneliness,	Public

	SocialSocialIsolation)	

8. RiskFactor

- Brief description: The risk pool consisting of different risk factors, which are explained by class RiskFactor. The attributes and methods are as follows.

Attributes	Attributes Description	Visibility
RiskFactorName: “character”	Name of the risk factor according to the specified names in class ‘Person.R’. Example: EducationYears, LivingCondition	Private
RiskFactorKind: “character”	Specifies the kind of the risk factor. Example: numeric Enumeration	Private
RiskFactorDataType: “Enumeration”	Specifies the data type of the risk factor. Example: SelfPerceivedHealth = new(Class="Enumeration", EValue="Unknown", Enumerations=c("Yes","No","Unknown")),	Private
RiskFactorDataFrame: “data.frame”	A data frame to store different values.	Private
RiskFactorPrevalence="numeric"	Specifies the prevalence of the risk factor.	Private
IntervalsLowerBound: “numeric”	Specifies the lower bound of the risk factor interval. Example: -Inf, 0, 1	Private
IntervalsUpperBound: “numeric”	Specifies the upper bound of the risk factor interval. Example: Inf, 10, 20	Private

Methods	Methods Description	Visibility
getRiskFactorName(object)	This method returns the name of the factor. object: “RiskFactor”	Public
setRiskFactorName(object, value)	This method changes the name of the factor. object: “RiskFactor”, value: “character”	Public
getRiskFactorKind(object)	This method returns the kind of the factor. object: “RiskFactor”	Public

setRiskFactorKind(object, value)	This method changes the kind of the factor. object: “RiskFactor”, value: “character”	Public
getRiskFactorDataFrame(object)	This method returns the data frame of the factor. object: “RiskFactor”	Public
setDataFrame()	This method changes the data frame of the factor.	Public
getRiskFactorCategories(object)	This method returns the categories of the factor. object: “RiskFactor”	Public
getRiskFactorIntervalsLowerBound(object)	This method returns the lower bound of the risk factor interval. object: “RiskFactor”	Public
getRiskFactorIntervalsUpperBound(object)	This method returns the upper bound of the risk factor interval. object: “RiskFactor”	Public
getRiskFactorOddsRatios(object, componentName)	This method returns the odd ratios of the risk factor. object: “RiskFactor” componentName: “character”	Public
findsOddsRatio(object, risk, componentName)	This method returns the specified Odd Ratio (OR). object: “RiskFactor” risk: “character” componentName: “character”	Public
loadRiskFactorSpecifications(object, specificationPath)	This method loads risk factor specifications from the Excel files (local path or relative path or from the Middleware). object: “RiskFactor” specificationPath: “character”	Public

getRiskFactorPrevalence(object)	This method returns the prevalence of the risk factor. object: “RiskFactor”	Public
setRiskFactorPrevalence(object, value)	This method changes the prevalence of the risk factor. object: “RiskFactor” value: “numeric”	Public

9. AgeGenderAdjuster

- Brief description: Describes an age group with a name, an upper and lower bound of the age which calculates prevalence for different gender. The attributes and methods are as follows.

Attributes	Attributes Description	Visibility
AgeGroupNames: char [1..*]	Name of each age group.	Private
AgeGroupLowerBounds: double [1..*]	Age lower bound of the group	Private
AgeGroupUpperBounds: double [1..*]	Age upper bound of the group	Private
FemalePrevalence: double [1..*]	Prevalence for female individuals	Private
GenderFreePrevalence: double [1..*]	Prevalence for gender free individuals	Private
MalePrevalence: double [1..*]	Prevalence for male individuals	Private

Methods	Methods Description
getAgeGroupNames(object): “char”	This method returns the group names of the adjuster. object: “Adjuster”
getAgeGroupLowerBound(object): “double”	This method returns the lower bound of the age group. object: “Adjuster”
getAgeGroupUpperBound(object): “double”	This method returns the upper bound of the age group. object: “Adjuster”
getMalePrevalence(object): “double”	This method returns the male prevalence. object: “Adjuster”
getFemalePrevalence(object): “double”	This method returns the female prevalence.

	object: “Adjuster”
getGenderFreePrevalence(object): “double”	This method returns the gender free prevalence. object: “Adjuster”
getPrevalenceDataFrame(object): “data.frame”	This method returns the data frame of prevalence. object: “Adjuster”
setPrevalenceDataFrame(object, value)	This method changes the data frame of prevalence. object: “Adjuster” value: “data.frame”
computePrevalence(object, age, gender, outcomename): int	This method computes the prevalence for a specific age and gender. object: “Adjuster” age: “numeric” gender: “character” outcomename: “character” Example: prevelancePhysicalFrailty= Adjuster.computePrevalence(physicalFrailtyPool@AgeGenderAdjuster,age, gender, "PhysicalFrailty")

Remark: The attributes and methods coloured in green in class “AgeGenderAdjuster” above are used when the prevalence data of a domain or outcome are loaded from a spread sheet page. Currently the prevalence data are extracted from a nonlinear regression model so they are not used in practice.

10. PhysicalRiskModel

- Brief description: Specific risk model for the risk of physical changes which inherits from the class ‘PersonalizedRiskModel’.

11. CognitiveRiskModel

- Brief description: Specific risk model for the risk of cognitive changes which inherits from the class ‘PersonalizedRiskModel’.

12. PsychologicalRiskModel

- Brief description: Specific risk model for the risk of psychological changes which inherits from the class ‘PersonalizedRiskModel’.

13. SocialRiskModel

- Brief description: Specific risk model for the risk of social changes which inherits from the class ‘PersonalizedRiskModel’.

14. FralityRiskModel

- Brief description: Specific risk model for the risk of frailty which realizes class ‘PhysicalRiskModel’.

15. FallRiskModel

- Brief description: Specific risk model for the risk of falls which realizes class ‘PhysicalRiskModel’.

16. DementiaRiskModel

- Brief description: Specific risk model for the risk of dementia which realizes class ‘CognitiveRiskModel’.

17. DepressionFallModel

- Brief description: Specific risk model for the risk of depression which realizes class ‘PsychologicalRiskModel’.

18. LonelinessRiskModel

- Brief description: Specific risk model for the risk of loneliness which realizes class ‘SocialRiskModel’.

19. SocialIsolationRiskModel

- Brief description: Specific risk model for the risk of social isolation which realizes class ‘SocialRiskModel’.

20. Person

- Brief description: Risk models are associated to persons. This class describes a person with different attributes to explain them. The received information can be modified (get and set methods).

Attributes	Attributes Description	Visibility
ID: “numeric”	A unique ID which is given to each person.	Private
DateOfBirth: “date”	Date of the birth of the person (DD-MM-YYYY)	Private
Gender: “Enumeration”	Person’s gender Possible values: {“Male”, “Female”, “Unknown”}	Private
Education years: “numeric”	Person's years of education Possible value: a non-negative integer	Private
Weight: “numeric”	Person's weight	Private
Height: “numeric”	Person's height	Private
LivingCondition: “Enumeration”	Person's living condition Possible value: {“unknown”, “Alone”, “Not Alone”}	Private
Employment: “Enumeration”	Person's employment situation	Private

	Possible values: { "Working","Unemployed","Retired","Unknown"}	
SelfPerceivedHealth : "Enumeration"	Person’s Self perceived health status Possible values: {“Yes”, “No”, “Uknown”}	Private
PriorDepression: "Enumeration"	Person’s prior depression status Possible values: {“Yes”, “No”, “Uknown”}	Private
DBMI : "Enumeration"	Person’s Discrete Body Mass Index (BMI can be treated both continuously or discrete depending on the outcome. Possible values: { "Healthy", "Overweight", "Unknown"}	Private
FrailAge : "Enumeration"	Person’s age to be used for frailty outcome. Possible values: { "Unknown", "85+", "85-"} 85+ indicates age>85 85- indicates age=<85	Private
DepressiveSymptoms: "Enumeration"	Person’s depressive symptoms based on the outcome of GDS test. Possible values: { "Unknown", "GDS6+", "GDS6-"} GDS6+ indicates the test result greater than 6. GDS6- indicates the test result less than or equal to 6.	Private
PoorSelfRatedHealth: "Enumeration"	Person’s poor rated health status Possible values: { "Unknown", "Yes", "No"}	Private
Comorbidities: "Enumeration"	Person’s comorbidities status Possible values: {"Unknown", "3+", "3-"} 3+ indicates more than 3 diseases. 3- indicates less than or equal to 3 diseases.	Private
CognitiveImpairment: "Enumeration"	Person’s cognitive impairment bases on the cognitive test MMSE. Possible values: { "Unknown", "MMSE18+", "MMSE18-"} MMSE18+ indicates the test result greater than 18. MMSE18- indicates the test result less than or equal to 18.	Private

HistoryOfFalls : "Enumeration"	Person's history of falls record Possible values: { "Unknown", "Yes", "No"}	Private
HistoryOfStroke : "Enumeration"	Person's history of stroke record Possible values: { "Unknown", "Yes", "No"}	Private
Hypotension: "Enumeration"	Person's hypotension status Possible values: { "Unknown", "Yes", "No"}	Private
GaitProblems: "Enumeration"	Person's gait problems status Possible values: { "Unknown", "Yes", "No"}	Private
UrinaryIncontinence: "numeric"	Person's urinary incontinence status Possible values: a non-negative integer	Private
AlcoholProblems: "Enumeration"	Person's alcohol intake status per day Possible values: { "Unknown", "250ml+", "250ml-"} 250ml+ indicates intake alcohol greater more than 250ml per day. 250ml- indicates intake alcohol less than or equal to 250ml per day.	Private
Smoking: "numeric"	Person's smoking habit status in terms of number of smokes per day Possible values: a non-negative integer	Private
FunctionalDisability : "numeric"	Man-person's functional disability in terms of Barthel index Possible values: an integer between 0 and 100	Private
MenEarlyRetirement: "Enumeration"	Man-person's early retirement status Possible values: { "Unknown", "Yes", "No"}	Private
WomenEarlyRetirement: "Enumeration"	Woman-person's early retirement status Possible values: { "Unknown", "Yes", "No"}	Private

ColorectalCancerScreening: "Enumeration"	Person's colorectal cancer screening status Possible values: { "Unknown", "Yes", "No"}	Private
LifePartner : "Enumeration"	Person's partnership status Possible values: { "Unknown", "Partnered", "Not Partnered"}	Private
LivingArea : "Enumeration"	Person's living area status Possible values: { "Unknown", "Metropolitan", "Rural"}	Private
HouseholdSize: "Enumeration"	Person's household size status Possible values: { "Unknown", "1 Person", "More than 1"}	Private
OwningPet : "Enumeration"	Person's owning pet status Possible values: { "Unknown", "Yes", "No"}	Private
Malnutrition : "Enumeration"	Person's mal nutrition status Possible values: { "Unknown", "Yes", "No"}	Private
LifeEnjoyment : "Enumeration"	Person's joy status in the life Possible values: { "Unknown", "Enjoying", "Not Enjoying"}	Private
NeedPersonalCare : "Enumeration"	Person's dependency's to the others status Possible values: { "Unknown", "Yes", "No"}	Private
FriendsRelativesContact: "Enumeration"	Person's contacts activity with the friends Possible values: { "Unknown", "Yes", "No"}	Private
Smoking: "Enumeration"	This specifies whether the person is smoking or not. Possible values: {"Yes", "No", "Unknown"}	Private
BMI: "numeric"	Person's body mass index Possible values: a positive float	Private
Cholesterol: "Enumeration"	Person's Cholesterol level	Private

	Possible values: {“Normal”, “High”, “Unknown”}	
Diabetes: “Enumeration”	This specifies whether the person is diabetes or not. Possible values: {“Yes”, “No”, “Unknown”}	Private
BloodPressure: “Enumeration”	Person's blood pressure Possible values: {“Low”, “Normal”, “High”, “Unknown”}	Private

Methods	Methods Description	Visibility
getXXX(object)	This generic method returns the person's XXX. XXX is replaced by one of the ‘Person’ class attributes mentioned above. object: “Person”	Public
setXXX(object, value)	This generic method modifies the person's XXX. XXX is replaced by one of the ‘Person’ class attributes mentioned above. object: “Person” value: a type matching with the corresponding attribute XXX	Public

21. PersonalizedRiskModel

- Brief description: It describes the risk model for a specific person which consists of a list of different risk pools. The attributes and methods are listed as follows.

Attributes	Attributes Description	Visibility
ListOfPools: “list”	The risk models consist of different risk pools (“list”) specified for a person (“Person”) which are stored in a list.	Private
Person: “Person”	Every personalized risk model is specified for person (“Person”).	Private

Methods	Methods Description	Visibility
loadRiskPoolSpecifications(object, specificationPathRiskPool)	This method loads the specifications of the risk pool. object: “personalizedRiskModel”	Public

	specificationPathRiskPool: “character” (local path or relative path or from the Middleware)	
findPoolIndices(object, poolNames)	This method finds the pool index in the specification table. object: “personalizedRiskModel” poolNames: “character” Possible fields: { PhysiFrailtyRiskFactors, PhysiFallsRiskFactors, CogDementiaRiskFactors, PsychoDepressionRiskFactors, SocLonelinessRiskFactors, SocSocialIsolationRiskFactors }	Public
findRiskFactorsIndicesInPool(object, poolNames, riskNames)	This method finds the risk factor index in the specification table of the corresponding risk pool. object: “personalizedRiskModel” poolNames: “character” Possible fields: { PhysiFrailtyRiskFactors, PhysiFallsRiskFactors, CogDementiaRiskFactors, PsychoDepressionRiskFactors, SocLonelinessRiskFactors, SocSocialIsolationRiskFactors } riskNames: “character” Possible values: one of the risk factors specified in the ‘Person.R’ class as its attribute.	Public
calcPersonalRisk(object)	This method calculates the risk of the risk model and returns the value. object: “personalizedRiskModel” evaluated by an object of class ‘Person.R’ The output order of the calculated risk values: 1- Frailty 2- Falls 3- Dementia 4- Depression 5- Loneliness 6- Social Isolation	Public

Furthermore, there is one type of enumeration that is used:

- RiskFactorDataType

The class ‘RiskFactor’ distinguishes between “Numeric”, “Categorical” or “NumericOrCategorical” risk factors with the enumeration ‘RiskFactorDataType’.

4.2 Mathematical functions

1. **adjustedOddRatio**

- **Input parameters:** oddsRatio, riskFactorPrevelances, isRiskFactorPresent
- **Task:** This function calculates the adjusted odd ratio based on the input odd ratio and risk factor prevalence if the risk factor is present:

  1.0+ (oddsRatios-1.0) * (1.0 – riskFactorPrevelances)
  Or if the risk factor is not present:
  1.0/ (1+(oddsRatios-1.0) * riskFactorPrevelances)

- **Remarks:**
  - oddsRatio must be a non-negative numeric vector.
  - All riskFactorPrevelances should be in I=[0,1].
  - The length of oddsRatios and riskFactorPrevelances should be equal.
  - isRiskFactorPresent should be a Boolean.
  - When a risk factor is continuous its corresponding odds ratio is adjusted by passing it to another function as below.

2. **adjustedOddRatioContinious**

- **Input parameters:** oddsRatio, level, reflevel
- **Task:** This function calculates the adjusted odd ratio for continuous risk factors.
- **Remark:**
  - level is presence status of the risk factor and is a character like “Present” or “Absent”.
  - reflevel is the prevalence of the risk factor which has a numeric type.
  - If oddsRatio<1, its inverse is passed to the function.

3. **likelihoodRatio**

- **Input parameters:** sensitivity, specificity
- **Task:** This function calculates the likelihood ratio based on the input sensitivity and specificity:

  sensitivity / (1-specificity)

- **Remarks:**
  - The sensitivity and specificity should be numeric vectors.
  - The length of sensitivity and specificity should be equal.
  - The length of sensitivity and specificity should be non-negative.

4. **oddsRatiosAverage**

- **Input parameters:** oddsRatios, sampleSizes, studiesQuality
- **Task:** This function computes the odds ratios average based on D3.1 page 13:

  ∑(oddsRatios * sampleSizes * studiesQuality) / ∑(sampleSizes * studiesQuality)

- **Remarks:**
  - All arguments should be numeric vectors.
  - The length of all arguments should be equal.

5. **oddsRatiosToProbability**

- **Input parameters:** oddsRatios
- **Task:** This function converts the odds ratios to the probability:

oddsRatios / (1.0 + oddsRatios)

- **Remarks:**
    - The oddsRatios should be a numeric vector.
    - All oddsRatios should be non-negative.

6. posteriorOddsRatio

- **Input parameters:** likelihoodRatio, priorOddsRatio
- **Task:** This function calculates the posterior odd ratio based on the likelihood ratio and prior odd ratio:

likelihoodRatio * priorOddsRatio

- **Remarks:**
    - The likelihoodRatio and priorOddsRatio should be numeric vectors.
    - The length of vectors likelihoodRatio and priorOddsRatio should be equal.

7. prevelanceAverage

- **Input parameters:** prevelances, sampleSizes, studiesQuality
- **Task:** This function computes the prevelance average based on D3.1 page 12:
  ∑ (prevelances * sampleSizes * studiesQuality) / ∑(sampleSizes * studiesQuality)

- **Remarks:**
    - All arguments should be numeric vectors.
    - The length of all arguments should be equal.

8. probabilityToOddsRatios

- **Input parameters:** probability
- **Task:** This function converts the probability to odd ratios.
  probability / (1-probability)

- **Remarks:**
    - The probability should be a numeric vector.
    - All probabilities should be in I = [0, 1].

9. relativeRisk

- **Input parameters:** oddsRatios, baselineRisks
- **Task:** This function calculates the relative risk based on the odd ratios and base line risks:
  oddsRatios / (1-baselineRisks + (baselineRisks*oddsRatios))
- **Remarks:**
    - The oddsRatios should be a numeric vector.
    - All non-NA 'oddsRatios' should be non-negative.
    - The baselineRisks should be a numeric vector.
    - All baselineRisks should be in I=[0,1].

10. sensitivity

- **Input parameters:** truePositives, falseNegatives
- **Task:** This function calculates sensitivity based on the true positive and false negative rates:
  truePositives / (truePositives+falseNegatives)
- **Remarks:**
    - The truePositives and falseNegatives should be numeric vectors.

  - The length of truePositives and falseNegatives should be equal.
  - The truePositives and falseNegatives should be non-negative.
  - The truePositives and falseNegatives should be integers.

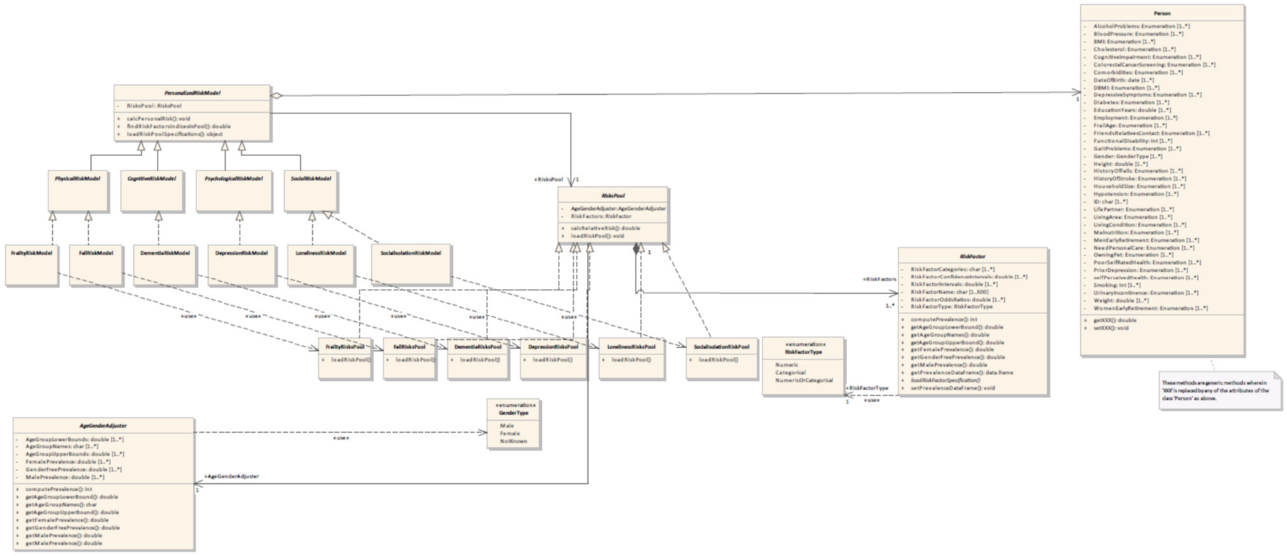
11. specificity

- **Input parameters:** trueNegatives, falsePositives
- **Task:** This function calculates specificity based on the true negative and false positive rates: trueNegatives / (trueNegatives+falsePositives)
- **Remarks:**
  - The trueNegatives and falsePositives should be numeric vectors.
  - The length of trueNegatives and falsePositives should be equal.
  - The trueNegatives and falsePositives should be non-negative.
  - The trueNegatives and falsePositives should be integers.

5 Conclusion

In this deliverable document, the technical description of the updated DSS software developed by R has been given. The methods for feeding the DSS with the users' data locally or through communicating with the Middleware have been mentioned and the functionalities of classes together with their attributes, methods and relationship have been presented. Finally the constructional mathematical functions used in the current platform have been described. This is the second document describing the DSS plat form whose ancestor is D6.3 (DSS platform I). The main upcoming module which will be deployed based on the current development and now is test version prepared is the automatic intervention assigning module. The final DSS will contain the final risk model (to be developed in WP3) wherein the interaction between all risk domains are considered.

Annex UML Class Diagram of the current status of the DSS core without considering the DSS-MW interface


PersonalizedRiskModel
- RisksPool: RisksPool
+ calcPersonalRisk(): void
+ findRiskFactorsIndicesInPool(): double
+ loadRiskPoolSpecifications(): object
PhysicalRiskModel
CognitiveRiskModel
PsychologicalRiskModel
SocialRiskModel
FrailtyRiskModel
FallRiskModel
DementiaRiskModel
DepressionRiskModel
LonelinessRiskModel
SocialIsolationRiskModel
«use»
«use»
«use»
«use»
«use»
«use»
FrailtyRisksPool
+ loadRiskPool()
FallRisksPool
+ loadRiskPool()
DementiaRisksPool
+ loadRiskPool()
DepressionRisksPool
+ loadRiskPool()
LonelinessRisksPool
+ loadRiskPool()
SocialIsolationRiskPool
+ loadRiskPool()
+RisksPool
1
RisksPool
- AgeGenderAdjuster: AgeGenderAdjuster
- RiskFactors: RiskFactor
+ calcRelativeRisk(): double
+ loadRiskPool(): void
1
+RiskFactors
1..*
RiskFactor
- RiskFactorCategories: char [1..*]
- RiskFactorConfidenceIntervals: double [1..*]
- RiskFactorIntervals: double [1..*]
- RiskFactorName: char [1..500]
- RiskFactorOddsRatios: double [1..*]
- RiskFactorType: RiskFactorType
+ computePrevalence(): int
+ getAgeGroupLowerBound(): double
+ getAgeGroupNames(): double
+ getAgeGroupUpperBound(): double
+ getFemalePrevalence(): double
+ getGenderFreePrevalence(): double
+ getMalePrevalence(): double
+ getPrevalenceDataFrame(): data.frame
+ loadRiskFactorSpecification()
+ setPrevalenceDataFrame(): void
«enumeration»
RiskFactorType
Numeric
Categorical
NumericOrCategorical
+RiskFactorType
1
«use»
1
Person
- AlcoholProblems: Enumeration [1..*]
- BloodPressure: Enumeration [1..*]
- BMI: Enumeration [1..*]
- Cholesterol: Enumeration [1..*]
- CognitiveImpairment: Enumeration [1..*]
- ColorectalCancerScreening: Enumeration [1..*]
- Comorbidities: Enumeration [1..*]
- DateOfBirth: date [1..*]
- DBMI: Enumeration [1..*]
- DepressiveSymptoms: Enumeration [1..*]
- Diabetes: Enumeration [1..*]
- EducationYears: double [1..*]
- Employment: Enumeration [1..*]
- FrailAge: Enumeration [1..*]
- FriendsRelativesContact: Enumeration [1..*]
- FunctionalDisability: int [1..*]
- GaitProblems: Enumeration [1..*]
- Gender: GenderType [1..*]
- Height: double [1..*]
- HistoryOfFalls: Enumeration [1..*]
- HistoryOfStroke: Enumeration [1..*]
- HouseholdSize: Enumeration [1..*]
- Hypotension: Enumeration [1..*]
- ID: char [1..*]
- LifePartner: Enumeration [1..*]
- LivingArea: Enumeration [1..*]
- LivingCondition: Enumeration [1..*]
- Malnutrition: Enumeration [1..*]
- MenEarlyRetirement: Enumeration [1..*]
- NeedPersonalCare: Enumeration [1..*]
- OwningPet: Enumeration [1..*]
- PoorSelfRatedHealth: Enumeration [1..*]
- PriorDepression: Enumeration [1..*]
- selfPerceivedHealth: Enumeration [1..*]
- Smoking: int [1..*]
- UrinaryIncontinence: Enumeration [1..*]
- Weight: double [1..*]
- WomenEarlyRetirement: Enumeration [1..*]
+ getXXX(): double
+ setXXX(): void
AgeGenderAdjuster
- AgeGroupLowerBounds: double [1..*]
- AgeGroupNames: char [1..*]
- AgeGroupUpperBounds: double [1..*]
- FemalePrevalence: double [1..*]
- GenderFreePrevalence: double [1..*]
- MalePrevalence: double [1..*]
+ computePrevalence(): int
+ getAgeGroupLowerBound(): double
+ getAgeGroupNames(): char
+ getAgeGroupUpperBound(): double
+ getFemalePrevalence(): double
+ getGenderFreePrevalence(): double
+ getMalePrevalence(): double
+ getMalePrevalence(): double
«use»
«enumeration»
GenderType
Male
Female
NotKnown
+AgeGenderAdjuster
1

These methods are generic methods wherein 'XXX' is replaced by any of the attributes of the class 'Person' as above.