



FP7-ICT-611140 CARRE

Project co-funded by the European Commission
under the Information and Communication Technologies
(ICT) 7th Framework Programme



D.6.4. Tools for adapting and creating personalized services

R. Kloda, J. Piwinski, R. Ugodzinski

July 2016

CARRE Contacts

Project Coordinator: Eleni Kaldoudi kaldoudi@med.duth.gr
Project Manager: George Drosatos gdrosato@ee.duth.gr

DUTH Democritus University of Thrace	Eleni Kaldoudi	kaldoudi@med.duth.gr
OU The Open University	John Domingue	john.domingue@open.ac.uk
BED Bedfordshire University	Enjie Liu	Enjie.Liu@beds.ac.uk
VULSK Vilnius University Hospital Santariškių Klinikos	Domantas Stundys	Domantas.Stundys@santa.lt
KTU Kaunas University of Technology	Arūnas Lukoševičius	arunas.lukosevicius@ktu.lt
PIAP Industrial Research Institute for Automation & Measurements	Roman Szewczyk	rszewczyk@piap.pl

Disclaimer

This document contains description of the CARRE project findings, work and products. The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any sort of responsibility that might occur as a result of using its content.

In case you believe that this document harms in any way IPR held by you as a person or as a representative of an entity, please do notify us immediately.

The content of this publication is the sole responsibility of CARRE consortium and can in no way be taken to reflect the views of the European Union.

CARRE is a Specific Targeted Research Project partially funded by the European Union, under FP7-ICT-2013-10, Theme 5.1. "Personalized health, active ageing & independent living".



Document Control Page

Project

Contract No.: 611140
Acronym: CARRE
Title: Personalized Patient Empowerment and Shared Decision Support for Cardioresenal Disease and Comorbidities
Type: STREP
Start: 1 November 2013
End: 31 October 2016
Programme: FP7-ICT-2013.5.1
Website: <http://www.carre-project.eu/>

Deliverable

Deliverable No.: D.6.4
Deliverable Title: Tools for adapting and creating personalized services
Responsible Partner: PIAP
Authors: R. Kloda, J. Piwinski, R. Ugodzinski
Input from: All partners
Peer Reviewers: Allan Third (OU), George Drosatos (DUTH)
Task: T.6.4 Tools for adapting and creating new services
Task duration: 8 months: 1 November 2015 to 30 June 2016
Work Package: WP6: Domain specific empowerment & decision support services
Work Package Leader: PIAP – Rafal Kloda

Due Date: 30 June 2016
Actual Delivery Date: 18 July 2016
Dissemination Level: PU
Nature: R & D
Files and format: Deliverable report: 1 pdf file
Software source code available from project web site, see Annex 1:
<https://www.carre-project.eu/innovation/dss-alerts-entry-system/>

Version: 04
Status: ☐ Draft
☒ Consortium reviewed
☒ WP leader accepted
☒ Coordinator accepted
☐ EC accepted

Document Revision History

Version	Date	Modifications	Contributors
v01.0	16 May 2016	Template and content structure	Rafal Kloda Jan Piwinski
v01.1	8 June 2016	Updated template and content structure based on Teleconference Meeting	Robert Ugodzinski
v01.2	24 June 2016	Draft version	Rafal Kloda Jan Piwinski
v01.3	8 July 2016	First version ready for internal review	Rafal Kloda
v02.0	11 July 2016	Internal review	Roman Szewczyk
v03.0	13 July 2016	Reviewed version	Jan Piwinski
v04.0	18 July 2016	Final version	George Drosatos

Table of Contents

Terms and Definitions	8
1. Introduction	9
2. Architecture of DSS and tools for adapting personalized services	9
3. Tools for adapting existing and creating personalized services	11
3.1 Overview	11
3.2 Designed user interface	16
3.2.1 Implementation	17
3.3 Designed back-end solution	17
3.3.1 Implementation	18
4. Conclusion	18
Annex 1 DSS Alerts Entry System	19
<i>What is CARRE: DSS Alerts Entry System?</i>	20
<i>Download</i>	20
<i>The CARRE DSS Alerts Entry System is Open Source</i>	20
Annex 2 Risk Alerts code as examples of output from DSS Alerts Entry System	21
Annex 3 Design of an interactive GUI for multimedia data exchange using SUR40 multi-touch panel	44
Abstract	45
1. Introduction	45
2. SUR40 multi-touch panel	45
2.1 Unit description	46
2.2 PixelSense technology	46
2.3 Microsoft Surface 2.0 SDK	46
2.4 Tagged objects and blobs	46
3. General guidelines for interface designing	47
3.1 Interface designing guidelines for Surface devices	47
4. SUR40 as the interactive user interface in CARRE project	48
4.1 About CARRE project	48
4.2 Decision Support System in CARRE project	48
4.3 Designed interface	48
5. Summary	49
Acknowledgment	50
References	50

List of Figures

Figure 1. General structure of a decision support system	10
Figure 2. A conceptual idea of the tool for a adapting and creating personalized services	10
Figure 3. Device placed on Microsoft Surface displaying patient biometrics ²	12
Figure 4. An examples of user interfaces using Pixel Sense technology from: a) Texas Health Resources b) Dubai Health Authority.....	12
Figure 5. Doctor is sharing data and recommend dose of medications for patient	13
Figure 6. An example of DSS algorithm described in D.6.2	14
Figure 7. An example of BPMN process workflow diagram of patient clinic visit ⁵	14
Figure 8. Alfresco Activiti workflow modeler tool for graphical BPMN process definition	15
Figure 9. Snapshot of the logical expression builder.....	16
Figure 10. Screenshot of the risk alert component of application	17
Figure 11. Decision support service exposed methods (part of RESTful API).....	18

Executive Summary

This document is a deliverable report of D6.4 “Tools for adapting existing and creating personalized services” of WP6 in CARRE project. Some parts are based on previously submitted report of D6.1 “DSS Infrastructure”. In particular it covers the DSS tasks designated in Task 6.1. The aim is to identify the best methods to provide knowledge to users and what kind of knowledge shall be provided, for example: information, processes and experiences. This deliverable report focuses on the design and implementation of administrative tools and graphical interface for adapting/extending the provided services so that medical experts and patients can adapt existing services in the DSS or create new ones. Without proper tools it is impossible to achieve this goal.

About CARRE

CARRE is an EU FP7-ICT funded project with the goal to provide innovative means for the management of comorbidities (multiple co-occurring medical conditions), especially in the case of chronic cardiac and renal disease patients or persons with increased risk of such conditions.

Sources of medical and other knowledge will be semantically linked with sensor outputs to provide clinical information personalised to the individual patient, to be able to track the progression and interactions of comorbid conditions. Visual analytics will be employed so that patients and clinicians will be able to visualise, understand and interact with this linked knowledge and take advantage of personalised empowerment services supported by a dedicated decision support system.

The ultimate goal is to provide the means for patients with comorbidities to take an active role in care processes, including self-care and shared decision-making, and to support medical professionals in understanding and treating comorbidities via an integrative approach.

Terms and Definitions

The following are definitions of terms, abbreviations and acronyms used in this document.

Term	Definition
API	Application programming interface
CARRE Repository	The backend major component of the CARRE platform responsible for storing, indexing and accessing the public and private RDF data
DoW	Description of Work
DSS	Decision Support Service
LOD	Linked Open Data cloud
PHR	Personal Health Record
RDF	Resource Description Framework: a standard model for data interchange on the Web
RESTful API	A Web API that adheres to the REpresentational State Transfer architectural constraints
SPARQL	A RDF query language.
Triple	A statement in the subject-predicate-object expression
XML	Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

1. Introduction

Task 6.4 aims to develop a user friendly mechanism so that new decision support alerts can be easily developed by medical experts and patients alike to account for requirements either of different clinical practices or needs of different individuals. The current document aims to identify the best methods to provide knowledge to users and what kind of knowledge shall be provided.

Task 6.4 involves the development of tools for medical expert and patient, which aim is to:

- customize specific risk alert algorithms existing in repository, especially certain conditions;
- creating new risk alerts to decision support from latest publications of clinical trial results

in most common natural and user friendly way.

A full list of DSS algorithms as well as DSS messages with detailed explanation is in D6.2 and D6.3. In this task we made an attempt to generalize them in order to better understand the process of creating an algorithm and to provide correct design tools for adapting services for their end-users. Upon generalization of algorithms described in D6.2 and D6.3 we can state that to define each algorithm we require:

- knowledge,
- condition, and
- outcome

Knowledge is an input from one or more observables from patient sensor data & personal health records and input from public medical evidence & knowledge.

Condition is a specific formula which consists set of observables. This expression is later used by decision support service consequently resulting in desired outcome. An algorithm has at least one or more condition to execute specific action or is a trigger to another algorithm as well as other conditions to describe knowledge.

Outcome is a text message associated with diagnosis. The addressee of this DSS message is patient or medical expert. Form of this message could be various: SMS or e-mail notification, integrated in the main CARRE visual interface (e.g. as colour coding importance level or as message).

The following section describes the tools, which we have designed together with their connection to decision support service (DSS).

The remainder of this document is structured as follows: Section 2 presents the overall, architecture of DSS and tools for adapting personalized services. Section 3 describes the concept of tools for adapting existing and creating personalized services. This section also discusses design of user interface and connected with its interface back-end solution. Finally, Section 4 concludes the report.

2. Architecture of DSS and tools for adapting personalized services

A decision support system is a method of presenting large amounts of knowledge from a given problem domain and using this knowledge for solving problems. Typical system consists of four parts: a *knowledge base*, an *inference engine*, a *user interface* and a *working memory*¹. An architecture of such system is presented in Figure 1.

¹ P. Golanski, M. Perz-Osowska, M Szczekala, a demonstration model of a mobile expert system with augmented reality user interface supporting M-28 aircraft maintenance, Journal of KONBiN 3(31) 2014, ISSN 1895-8281

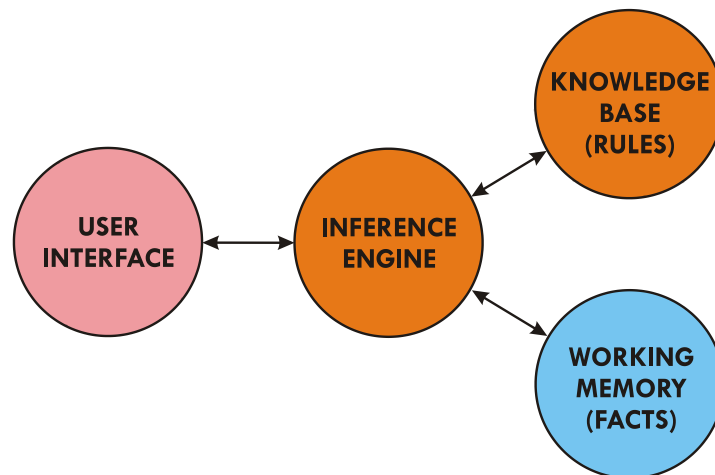


Figure 1. General structure of a decision support system

In order to solve a problem the user is conducting a dialogue with the machine using the *user interface*. During this dialog the user provides information about the problem to be solved. The *inference engine*, basing on rules in the *knowledge base* and problem-specific data in the *working memory*,

In case of DSS basic elements of the system: the *inference engine*, the *knowledge base* and the *working memory* constitute the *interpreter module*. Data for the *working memory* (set of facts) and a *knowledge base* (which is a set of rules for using facts), are encoded using Python. In DSS the *knowledge base* has a form of production rules. Those rules are in the form of sequences with a structure as follows:

if <conditions> **then** <actions>,

Rules are checked by the *inference engine* and if the information supplied by the operator meets the conditions in the rules, specific actions are executed.

One most important challenge in Task 6.4 was to implement the user-friendly risk alerts editor to enable experts and patient to customize specific decision for particular patient previously developed algorithms and also prepare new ones algorithms. There is no possibility to have hard coded *Knowledge Base* in DSS. It need special tool to modify existing and creating new algorithms. To achive our aim, we propose an idea as shown on Figure 2.

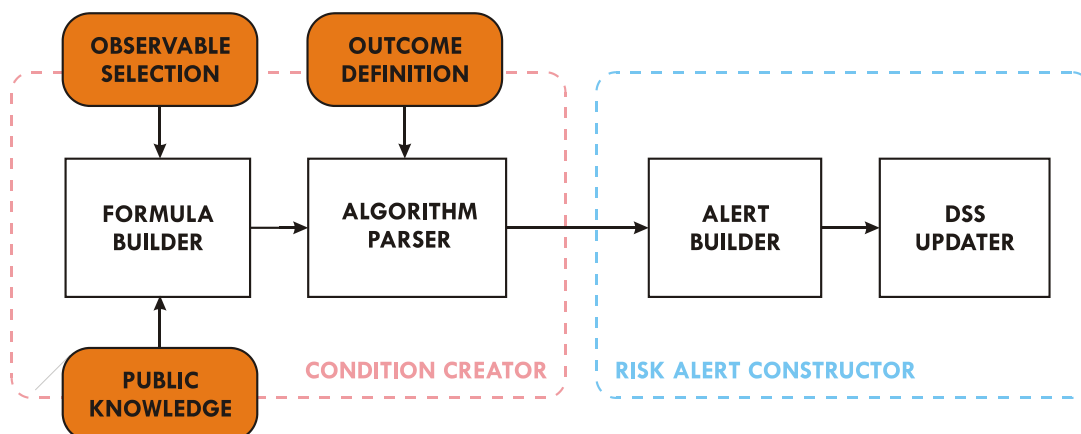


Figure 2. A conceptual idea of the tool for a adapting and creating personalized services

Condition creator is user-friendly interface which provides access to CARRE repositories and allows end-users to administer the purpose of algorithms. Due to the fact that end-users are not programmers the basic functionality of designed tools should be easy to operate (create or modify such algorithms). Another essential tool is a risk alert constructor. Its goal is to build, debug and compile working code from the pseudo code. In the next step such processed code should be dynamically integrated with other decision support component on the server side. That is the role of DSS updater shown on Figure 2.

This idea implies that one of these tools is dedicated to user (condition creator) and second to service (risk alert constructor). In both cases it is necessary to develop specialized data parser and integrator in order to translate human prepared condition and rules to machine inference engine.

The following section describes the designed tool for adapting existing and creating new algorithms for DSS service namely alert entry system.

3. Tools for adapting existing and creating personalized services

3.1 Overview

During Task 6.4 there was considered three technologies as solution for user friendly interface.

At the beginning phase of CARRE project we intended to use PixelSense technology, which allows each pixel on the screen to act as a sensor. This enables detection of objects that interact with the device. Integrated PixelSense sensors are built directly into the layers of the LCD screen. These sensors enable detection, identification and reaction to objects with predefined tags and to untagged objects. This technology allows simultaneous identification of 52 touch points. Samsung SUR40 is an example of a large-format multi-touch screen that provides PixelSense technology. The device is the size of an average 40" TV screen and thanks to the 360-degree interface lets a group of people use the SUR40 simultaneously. It is possible to create an application that allows one person to present the information to others, or allows a group of people to make a collaborative decision. We investigated this technology deeply (see Annex 3 for more detailed information). Our aim was to explore the possibilities of using the SUR40 device and its software as visualization platform in CARRE project.

This inspiration resulted in a broad overview of today's technologies used in widely understood health care. All trends suggested that by the year 2015 rapid advances in screen technology and the diminishing size of microprocessors will make it possible to invent new archetypes for the computer, coupled with new gestural and semantic languages. In an age of ubiquitous computing, our walls, tables and other elements in our environment will become platforms for us to interact on. Conducted research suggested that as we move towards the future of health care, people will increasingly need to feel involved and in control of their own health. People will also need tools to help them collaborate closely with health care providers, doctors and other people they trust to help them manage their health. Strategic work of this research resulted in an eco-system that described the Integrated Future of Health Care in the year 2015 with the patient at the center of all activities, services, devices and products.

To tangibly visualize all data and a glimpse of the future, the team developed the eco-system described in the Integrated Future of Health Care into an application for the Microsoft Surface platform (PixelSense technology). This technology was chosen for its unique ability to invite people into interactions and conversations around the display. Representing a future smart surface, it also provided our team with the opportunity to explore natural user interfaces (NUI) and at the same time challenged the team to design for a full 360-degree interaction and multi-input, multi-user collaboration.

Authors' aim was to illustrate the body as a container of biometric data. The simple act of placing your hand on 'a table' or any other type of smart surface, triggered an enlightened experience, e.g. you will be able to share and compare your biometric data with people you trust, subscribe to personalized treatment software and also have easy and constant access to your health care professionals (Figure 3)².

² <http://www.core77.com/posts/21488/case-study-minime-and-the-future-of-integrated-health-care-by-ergonomidesign-21488>



Figure 3. Device placed on Microsoft Surface displaying patient biometrics²

These solutions are becoming more common and their aim is to improve patient-medical expert interaction as well as interaction between medical experts. Typical interfaces are shown in Figure 4.

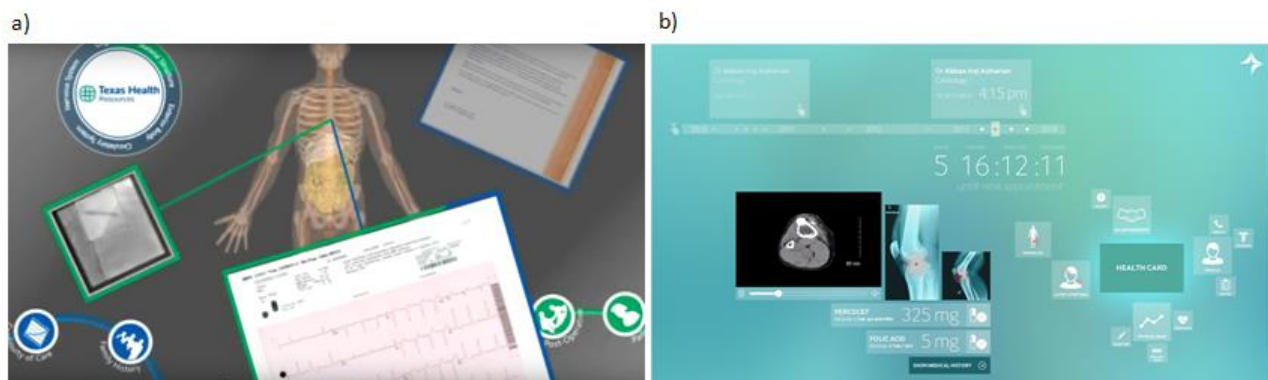


Figure 4. An examples of user interfaces using Pixel Sense technology from:
a) Texas Health Resources b) Dubai Health Authority

These kind of interface can assist patients and doctors by providing advices, recommendations and diagnosis of problems in aging and growing population with chronic diseases, by means of interactive visualization interface, variables and recommendation to intuitive and user-friendly visualization in patient application. Moreover interface provides especially to medical expert possibility review all types of data from the hospitals' records and the patient's records together with patient. They can sit side-by-side or across from each other viewing the same information using hand gestures to scroll through, open, zoom, rotate in 3D, push across the table, and drag and drop records from one storage repository to the other (Figure 5).



Figure 5. Doctor is sharing data and recommend dose of medications for patient

Examples presented here it just only a part of functionality of PixelSense Technology. In fact this technology is much more powerful. Here we give a brief summary:

- direct interaction allows the ability to "grab" digital information with hands and move them around freely on the surface;
- the multi-touch ability recognizes multiple points of contacts on the surface at the same time;
- surface also allows multiple users to interact on the same surface together;
- object recognition will use physical objects and will interact with them on the Surface's screen such as pictures on a camera, phones or sensor devices.

This is powerful technology with great support for hardware to create novel method of interact doctors and patients. This technology could help in healthcare (CARRE) by streamlining patient-doctor interaction and disaster management. Frankly speaking it is big screen, but functionality of interface could be quite easy adopted to further any mobile device platform. Certainly it is good platform for prototyping of user interface with hardware. As a result of a thorough analysis of the applicability of the SUR40 and PixelSense technology, we figured to the conclusion that it goes far beyond of Task 6.4.

Next tool which we considered was Alfresco Activiti. Alfresco Activiti provides a suite of user-friendly tools to model and deploy business processes³. Activiti is a light-weight workflow. It provides a fast and reliable BPMN 2.0 (Business Process Model and Notation) process engine for Java. It is open-source and distributed under an Apache license. Activiti is lightweight, based on open standards and is designed to integrate well with Spring applications, like Alfresco⁴. The Alfresco Activiti workflow engine executes BPMN 2.0 process definitions.

Motivation for this approach was fact that structure of building an algorithm presented in D6.2 and D6.3 is very similar to BPMN process definition as shown on Figure 6 and Figure 7.

³ <http://www.alfresco.com/products/activiti>

⁴ https://wiki.alfresco.com/wiki/Workflow_with_Activiti

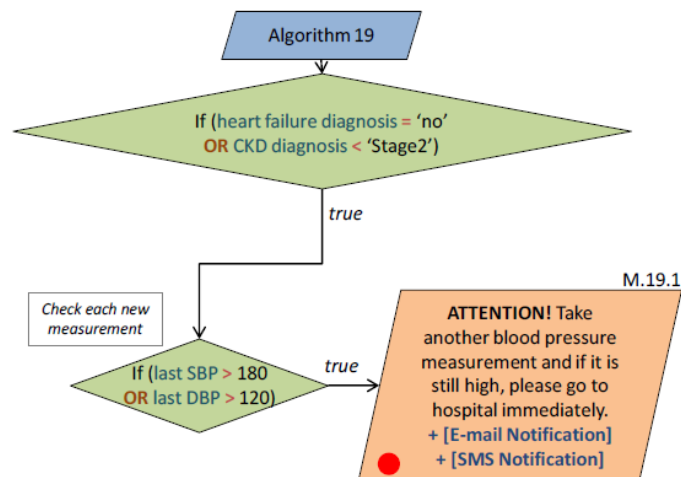


Figure 6. An example of DSS algorithm described in D.6.2

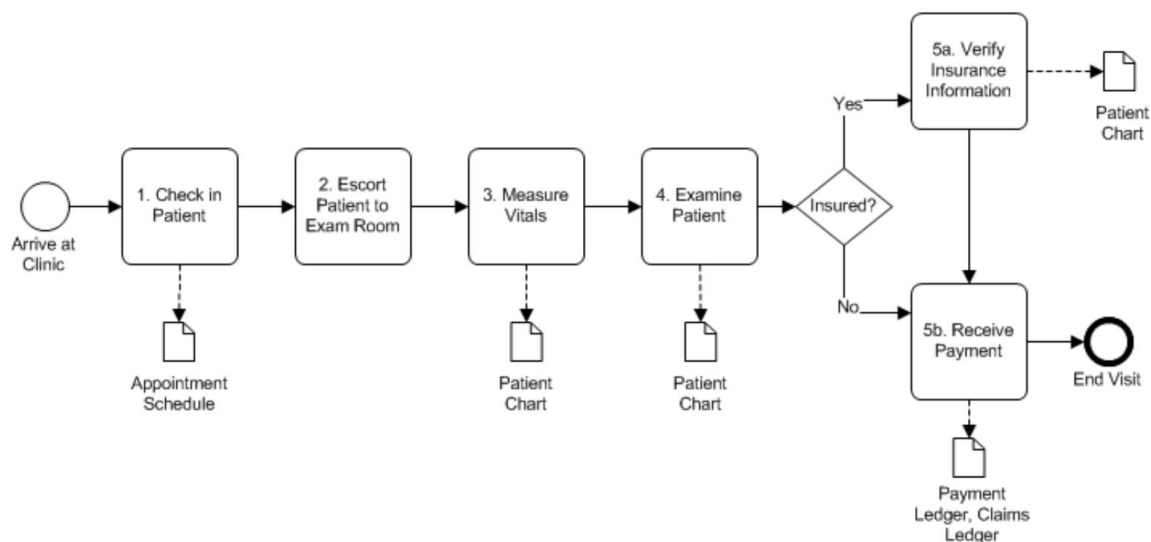


Figure 7. An example of BPMN process workflow diagram of patient clinic visit⁵

BPMN 2.0 (Business Process Model and Notation) is an open standard developed by the Object Management Group (OMG) to provide a simple and understandable mechanism for creating process models, including the ability to express complex processes. Specific shapes used to organize graphical aspects of the notation were defined to accomplish this goal. The result of this strategy provides a small set of shapes so that any process model reader will be able to easily recognize the basic elements and understand the diagrams. The simple core objects can be extended to add variations and information to support complex processes while maintaining the simplicity of the diagram^{5,6}. The second version extended the standard to include execution semantics, a common exchange format and to extend BPMN towards executive decision support.

A graphical workflow modeler is often used to create a workflow as shown on Figure 8. The workflow consists of tasks (work to perform), a gates (control flows in a process), and events (e.g. start and end) mentioned before represented as different shapes.

⁵ Business Process Modeling Notation (BPMN) – An Introduction to Process Diagrams, Association of State and Territorial Health Officials, <http://www.astho.org/Informatics/Documents/NYDOH-Business-Process-Modeling-Notation>

⁶ <http://docs.alfresco.com/4.1/concepts/wf-what-is-workflow.html>

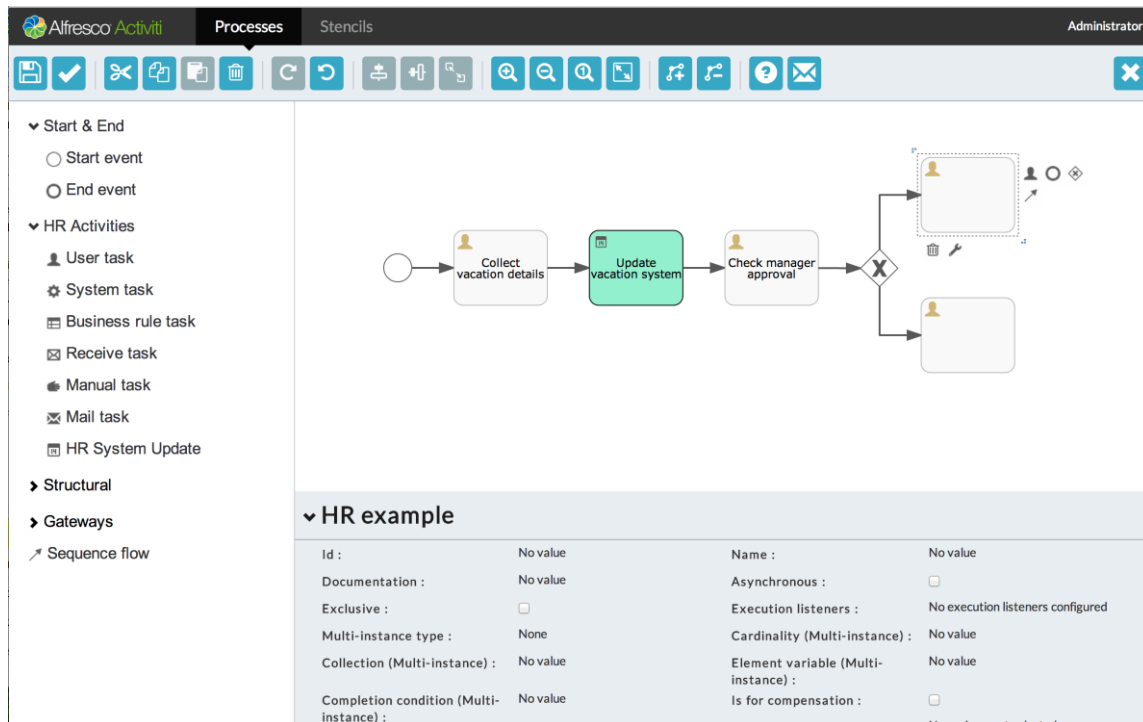


Figure 8. Alfresco Activiti workflow modeler tool for graphical BPMN process definition

In Alfresco Activiti a workflow is a sequence of connected tasks. Each task can be performed automatically (by scripts). For each task in the process definition can associate a task description. The description specifies the information that may be attached to a task: properties (name and datatype) and associations (name and type of associated object). Process definitions describe the events, activities (tasks) and gateways (choices) of a workflow. Tasks may be user tasks or script (system) tasks. User tasks are assigned to human performers (users). System tasks perform some kind of operation against the Alfresco repository. Both are described and implemented in the process definition. A process definition is an XML document which allow the integration of existing components of the CARRE system connected with DSS.

Third and the latest tool which we have taken into account was the observable and logical expression modeling tool presented on Figure 9. This tool was developed by members of this consortium and in part within CARRE project and certainly it will allow an easier implementation with other components. To aid this process we have developed a web-based system for the description of care plans which includes a graphical logical expression editor.

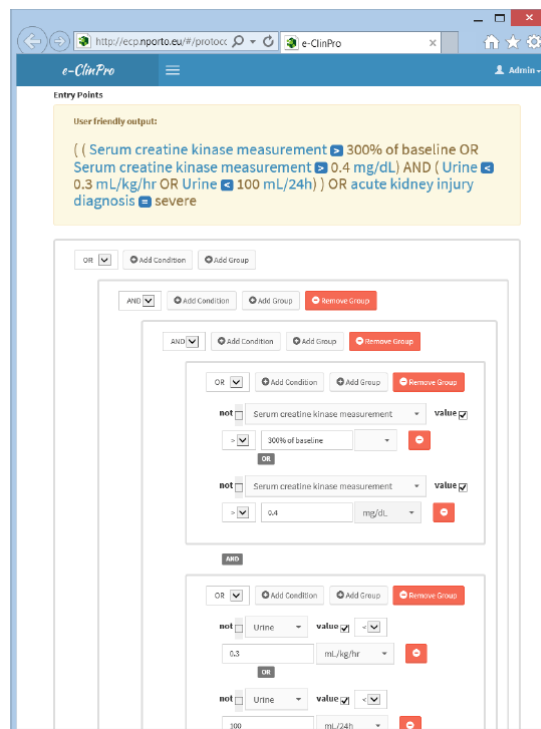


Figure 9. Snapshot of the logical expression builder⁷

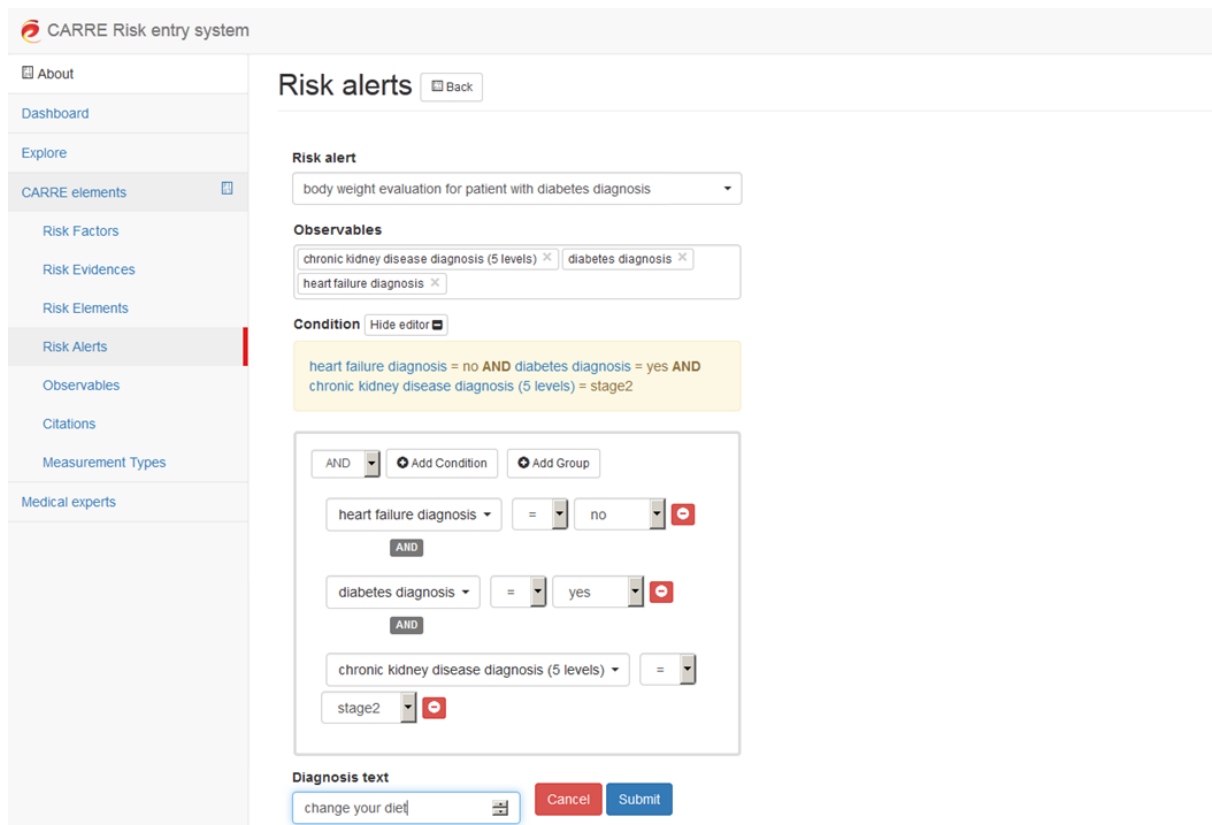
On the basis of this analysis we decided to provide a tool which could be easily integrated with other CARRE subsystems. It should be mentioned that all considered tools have ready to use environment with developer environment. In two first cases: Microsoft PixelSense technology and Alfresco Activiti, if we think about this as technology to provide graphical interface for our tool, they will be completely different from the others already developed and tested interfaces in the project. Following this way of thinking, we decided to use last mentioned tool to describe decision support algorithms in tool for adapting of existing and creating new personalized services.

3.2 Designed user interface

The basic feature of designed interface of DSS alert system is administrative purpose for adapting/extending the provided services so that medical experts and patients can adapt existing services in the DSS or create new ones. The CARRE alert entry system is designed for patients as well doctors.

Figure 10 below shows the particular DSS risk alert editor.

⁷ E. Kaldoudi, G. Drosatos, N. Portokallidis, A. Third. An Ontology based Scheme for Formal Care Plan Meta-Description. In Proc. of the 14th Mediterranean Conference on Medical and Biological Engineering and Computing (MEDICON 2016), Paphos, Cyprus, 31 Mar. – 2 Apr. 2016



CARRE Risk entry system

Risk alerts [Back](#)

Risk alert

body weight evaluation for patient with diabetes diagnosis

Observables

chronic kidney disease diagnosis (5 levels) × diabetes diagnosis ×
heart failure diagnosis ×

Condition [Hide editor](#)

heart failure diagnosis = no AND diabetes diagnosis = yes AND
chronic kidney disease diagnosis (5 levels) = stage2

AND Add Condition Add Group

heart failure diagnosis = no

AND

diabetes diagnosis = yes

AND

chronic kidney disease diagnosis (5 levels) = stage2

Diagnosis text

change your diet [Cancel](#) [Submit](#)

Figure 10. Screenshot of the risk alert component of application

There are three necessary functionalities that can be accessed by the end-user from the interface. A user can view, edit or create a new risk alerts in the *Risk Alerts* page component on their dashboard in CARRE Risk entry system. To present the user all available list of all risk alerts, interface provides in default view as a front page. The user can interact with all controls on the screen. The condition editor to edit or build more detailed algorithm.

3.2.1 Implementation

It should be mentioned that this tool DSS Alerts entry system was implemented as a part of CARRE Risk entry system described in D3.4. The expression builder follows a web component architecture and it is implemented in Javascript and HTML5 using the AngularJS framework and deployed on a Linux environment. Each component is placed in a DIV element in the component container and the container is managed automatically as tabs in the main user interface.

The decision support data are all fetched from the CARRE server and need to be accessed by SPARQL queries.

3.3 Designed back-end solution

It should be mentioned that for the implementation of the *inference engine* and the *knowledge base* we considered to use of the expert system tool namely CLIPS⁸ widely used in PIAP in similar purpose. This tool has also a module to interface Python called PyCLIPS⁹. The aim of PyCLIPS is to provide Python with a strong, reliable, widely used and well documented inference engine. Till now there are no official PyClips build for Python 2.7 in which is implemented CARRE RESTful API and DSS. Moreover from other side

⁸ <http://clipsrules.sourceforge.net/>

⁹ <https://sourceforge.net/projects/pyclips/>

analysing complexity of algorithms provided from Task 6.2 and 6.3 RETE¹⁰ implementation appears to be sufficient. In other words Python environment used as *inference engine* in this case is sufficient to implement such knowledge and provide efficient DSS service without additional tool for building such systems.

As we stated before, that is no possible to have hard coded *Knowledge Base* in DSS. We add to DSS special tool to for automatic risk alert alghoritm insertion to core alghoritm of DSS. Proposed alghoritm of back-end tool is stable and quite fast. Its aim is dynamic update Knowledge Base of decision support service. As it was mentioned this part of DSS service was implemented as part of RESTful API so this tool has also been implemented in Python on server side.

3.3.1 Implementation

It should be mentioned that this tool API for supporting alerts entry system was implementad as a part of CARRE RESTful API described in D.4.1. This application is a RESTful API developed using Flask, which is a Python-based web Framework. This component expands and exposes a number of metod conected to DSS. The homepage of the above service is served as swagger web pages in the following address:

<https://carre.kmi.open.ac.uk/ws>

GET	/DSS
GET	/riskAlertsList
GET	/riskAlerts

Figure 11. Decision support service exposed methods (part of RESTful API)

Figure 11 is a screenshot of the service homepage. The aim of this page is to list all methods concerning DSS available together with a documentation of each metod connected to DSS and mentioned tools. Currently, the methods implemented are the following:

- **DSS:** GET method that takes as input the *action* of the DSS (e.g. AddNewAlert) and the *token* of the user. The response is a JSON object with the results of the query.
- **riskAlertsList:** GET method that takes as input the token of a user. If valid, returns user alerts list information.
- **riskAlerts:** GET method that takes as input the *token* of a user and alert *id*. If valid, returns user alert outcome of particular alert.

The homepage of the service does not only serve as documentation. It allows developers to use the web services directly and test the behaviour of each method.

4. Conclusion

This document is a report of activities undertaken in task *T.6.4 Tools for adapting and creating personalized services*. This deliverable report focuses on the design and implementation of user friendly interface in order to provide possibility modify and creating new personalized services to both patient application and medical expert application.

¹⁰ https://en.wikipedia.org/wiki/Rete_algorithm

Annex 1

DSS Alerts Entry System

What is CARRE: DSS Alerts Entry System?

In CARRE system, **DSS Alerts Entry System** undertake the development of tools and the corresponding graphical interfaces to allow medical experts and patients to adapt existing personalized empowerment and decision support services and to build new ones. The main parts of this system are the **user interface** and the back-end service.

- The **user interface** is implemented as a part of [CARRE Risk Entry System](#) using Javascript, HTML5 and AngularJS framework. In this environment, a user (medical expert or patient) can view, manage, modify or create new personalized risk alerts.
- The **back-end service** is implemented as a part of [CARRE RESTful API](#) using Flask, a Python-based web framework.

Download

User interface for DSS alerts:

v0.5 (Released 10 July 2016, Deliverable 6.4)

- Source code: [DSS Alerts Entry System UI.zip](#) (Javascript)

Back-end service for DSS alerts:

v3.0 (Released 10 July 2016, Deliverable 6.4)

- Source code: [DSS Alerts Entry System back-end.zip](#) (Python)

The CARRE DSS Alerts Entry System is Open Source

CARRE DSS Alerts Entry System is Open Source and can be freely used in Open Source applications under the terms GNU General Public License (GPL).

Copyright © 2016, [CARRE Project](#), Industrial Research Institute for Automation & Measurements (PIAP), Poland and Democritus University of Thrace (DUTH), Greece and Open University (OU), UK

Annex 2

Risk Alerts code as examples of output from DSS Alerts Entry System

```

def mailNotification(msg):
    print "mail: " + msg

def smsNotification(msg):
    print "sms: " + msg

def message(alert, msg):
    print alert
    print msg

import carre

M1 = {1: 0, 2: 0, 3: 0, 4: 0, 5: 0}
M2 = {1: 0, 2: 0, 3: 0, 4: 0}
M3 = {1: 0, 2: 0, 3: 0, 4: 0}
M4 = {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0}

def getValue(predicate, token):
    import virtuoso
    username = virtuoso.getUsernameFromToken(token)

    pred = carre.getPredicate(predicate)

    sql = """
        SELECT ?value FROM <https://carre.kmi.open.ac.uk/users/"" +
username + ""> WHERE
    {
        ?subject ?predicate ?object.
        ?subject "" + pred + "" ?diagnosis.
        ?diagnosis <http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_value>
?value.
        ?subject <http://carre.kmi.open.ac.uk/ontology/sensors.owl#has date> ?d.
        ?d <http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_value> ?date.
    }
    ORDER BY DESC(?date)
    LIMIT 1
    """

    data = virtuoso.executeSPARQL(sql, token)

    return data

def getValues(predicate, token, startdate, group=False):
    import virtuoso
    username = virtuoso.getUsernameFromToken(token)

    pred = carre.getPredicate(predicate)

    if group:
        sql = """
            SELECT MAX(?values) xsd:date(?date) as ?date FROM
<https://carre.kmi.open.ac.uk/users/"" + username + ""> WHERE
        {
            ?subject ?predicate ?object.
            ?subject "" + pred + "" ?w.

```

```

        ?w
http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_value> ?values.
        ?subject
<http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_date> ?d.
        ?d
<http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_value> ?date.
        FILTER ( ( xsd:date(?date) >= '"" + startdate +
""'^^xsd:date) )
    }
    GROUP BY xsd:date(?date)
    ORDER BY ASC(?date)
    ""

else:
    sql = ""
        SELECT ?values ?date FROM <https://carre.kmi.open.ac.uk/users/"" +
username + ""> WHERE
    {
        ?subject ?predicate ?object.
        ?subject "" + pred + "" ?w.
        ?w http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_value>
?values.
        ?subject <http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_date>
?d.
        ?d <http://carre.kmi.open.ac.uk/ontology/sensors.owl#has_value>
?date.
        FILTER ( ( xsd:date(?date) >= '"" + startdate + ""'^^xsd:date) )
    }
    ORDER BY ASC(?date)
    ""

    data = virtuoso.executeSPARQL(sql, token)

    return data

def getAverage(data):
    suma = 0
    for i in len(data):
        suma += data[i]
    return suma / len(data)

def getDifference(data):
    min = data[0]
    max = 0

    for i in data:
        if min > i:
            min = i
        if max < i:
            max = i

    return max - min

# RA 1
def RA1(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)

```

```

if HFdiag['value'] == 'yes' or CKDDiag['value'] >= 'stage2':
    message("green", "Measure blood pressure once per day(morning)")
    M1[5] = 1
else:
    message("green", "Measure blood pressure twice per day (morning and
evening) for a week.")
    M1[1] = 1

startDate = carre.getStartDate(7)

SBPpw = getValues("SystolicBloodPressure", token, startDate)
DBPpw = getValues("DiastolicBloodPressure", token, startDate)

averageSBPpw = getAverage(SBPpw['values'])
averageDBPpw = getAverage(DBPpw['values'])
if averageSBPpw <= 135 or averageDBPpw <= 85:
    message("green",
            "Congratulations! Your blood pressure is well controlled. "
            "Please measure blood pressure once per week (morning).")
    M1[3] = 1

    lastSBP = getValue("SystolicBloodPressure", token)
    lastDBP = getValue("DiastolicBloodPressure", token)

    if lastSBP['value'] > 135 or lastDBP > 85['value']:
        message("green",
                "Your blood pressure reached abnormal values! Please
start your blood pressure"
                " measurements twice per day (morning and evening) for a
week.")
        M1[4] = 1
    else:
        M1[2] = 1

#####

# RA 2
def RA2(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)

    if HFdiag['value'] == 'yes' or CKDDiag['value'] >= 'stage2':
        message("green", "Measure body weight once per day(morning)")
        M2[1] = 1
    else:
        diabetesDiag = getValue("DiabetesDiagnosis", token)
        if diabetesDiag['value'] == 'yes':
            message("green", "Measure body weight once per week.")
            M2[2] = 1
        else:
            BMI = getValue("BMI", token)
            if BMI['value'] >= 25:
                message("green", "Measure body weight once per week.")
                M2[3] = 1
            else:
                message("green", "Occasionally measure body weight(at least once
per month,preferably once per week).")
                M2[4] = 1

```



```

# RA 3
def RA3(token):
    diabetesDiag = getValue("DiabetesDiagnosis", token)

    if diabetesDiag['value'] == 'yes':
        message("green", "Measure blood glucose three times per day(before
breakfast, before lunch and before dinner.")
        M3[1] = 1
    else:
        HFdiag = getValue("HeartFailureDiagnosis", token)
        CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
        BMI = getValue("BMI", token)
        hyperDiag = getValue("Hypertension", token)
        LDLC = getValue("LDL-C", token)
        TC = getValue("TC", token)

        if HFdiag['value'] == 'yes' or CKDDiag['value'] >= 'stage2' or
BMI['value'] >= 25 or hyperDiag[
        'value'] == 'yes' or LDLC['value'] >= 130 or TC['value'] >= 200:
            message("green", "Measure blood glucose once per month (before
breakfast).")
            M3[3] = 1
        else:
            message("green", "Occasionally measure blood glucose (before
breakfast, at least once per 6 months).")
            M2[4] = 1

# RA 4
def RA4(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)

    if HFdiag['value'] == 'yes':
        message("yellow", "Please ask your doctor for recommended steps per
day.")
        M4[1] = 1
    else:
        CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
        if CKDDiag['value'] == 'stage5':
            message("green", "Please walk 30 min (~3000 steps or 2.4 Km) 3-4
times per week.")
            M4[2] = 1
        else:
            hyperDiag = getValue("Hypertension", token)
            LDLC = getValue("LDL-C", token)
            TC = getValue("TC", token)
            if 'stage2' <= CKDDiag['value'] <= 'stage4' or hyperDiag['value'] ==
'yes' or LDLC['value'] >= 130 or TC[
            'value'] >= 200:
                message("green", "Please walk at least 30 min (over 3000 steps
or 2.4 Km) per day.")
                M4[3] = 1
            else:
                diabetesDiag = getValue("DiabetesDiagnosis", token)
                BMI = getValue("BMI", token)
                if diabetesDiag['value'] == 'yes' and BMI['value'] >= 25:
                    message("green", "Please walk at least 90 min (over 9000
steps or 7.2 Km) per day.")
                    M4[5] = 1

```

```

else:
    if diabetesDiag['value'] == 'yes' or BMI['value'] >= 25:
        message("green",
                "Please walk at least 60 min (over 6000 steps or
4.8 Km) per day.")
        M4[4] = 1
    else:
        message("green",
                "If it is possible, try to walk instead of using
transportation.")
        M4[6] = 1

# RA 5
def RA5(token):
    if M1[1] == 1 or M1[4] == 1:

        from datetime import timedelta, time
        cnt = 0
        cntF = 0

        startDate = carre.getStartDate(7)

        measurements = getValues("BloodPressure", token, startDate)

        p = 0
        t0 = time(hour=0)
        t1 = time(hour=12)
        t2 = time(hour=23, minute=59)

        for i in range(14):
            measurement = str(measurements['date'][p]).split("T")
            measurement[1] = measurement[1].replace("Z", ":00")
            if measurement[0] == str(startDate + timedelta(hours=12 * i)):
                if i % 2 == 0:
                    if str(t0) < measurement[1] < str(t1):
                        cnt += 1
                        cntF = 0
                        p += 1
                    else:
                        cntF += 1
                        cnt = 0

                else:
                    if str(t1) <= measurement[1] < str(t2):
                        cnt += 1
                        cntF = 0
                        p += 1
                    else:
                        cntF += 1
                        cnt = 0

            else:
                cntF += 1
                cnt = 0

        if cnt == 14:
            message("green", "Congratulations! You correctly follow your
instructions for 7 days.")
        elif cntF == 4:
            message("green",

```

```

        "Please follow your instructions: Measure blood pressure
twice per day (morning and evening).")
    elif cntF == 10:
        message("yellow",
            "You haven't taken measurement for blood pressure 5 days, "
            " please measure blood pressure twice per day.")
    elif cntF == 14:
        message("red",
            "You haven't taken measurement for blood pressure 7 days, "
            "please measure blood pressure at least twice per day.")
        mailNotification(
            "You haven't taken measurement for blood pressure 7 days, "
            "please measure blood pressure at least twice per day.")

# RA 6
def RA6(token):
    if M1[5] == 1:
        from datetime import timedelta

        cnt = 0
        cntF = 0
        startDate = carre.getStartDate(7)

        measurements = getValues("BloodPressure", token, startDate, group=True)
        p = 0

        for i in range(7):
            measurement = str(measurements['date'][p])
            if measurement == str(startDate + timedelta(days=i)):
                cnt += 1
                cntF = 0
                p += 1

            else:
                cntF += 1
                cnt = 0

        if cnt == 7:
            message("green", "Congratulations! You correctly follow your
instructions for 7 days.")
            elif cntF == 1:
                message("green", "Please follow your instructions:Measure blood
pressure once per day(morning).")
            elif cntF == 3:
                message("yellow", "You have high risk, pplease take your
measurements at least once per day.")
            elif cntF == 5:
                message("yellow",
                    "You haven't taken measurement for blood pressure 5 days, "
                    "please measure blood pressure once per day.")
                mailNotification(
                    "You haven't taken measurement for blood pressure 5 days, please
measure blood pressure once per day.")
            elif cntF == 7:
                message("red", "You haven't taken measurement for blood pressure 7
days, "
                    " please measure blood pressure once per day.")
                mailNotification(

```

```

        "You haven't taken measurement for blood pressure 7 days, please
measure blood pressure once per day.")
        smsNotification(
            "You haven't taken measurement for blood pressure 7 days, please
measure blood pressure once per day.")

# RA 7
def RA7(token):
    if M1[3] == 1:
        from datetime import timedelta

        cnt = 0
        cntF = 0
        startDate = carre.getStartDate(30)

        measurements = getValues("BloodPressure", token, startDate, group=True)
        p = 0

        for i in range(4):
            measurement = str(measurements['date'][p])
            if str(startDate + timedelta(days=i * 7)) <= measurement <
str(startDate + timedelta(days=7 + i * 7)):
                cnt += 1
                cntF = 0
                p += 1

            else:
                cntF += 1
                cnt = 0

        if cnt == 4:
            message("green", "Congratulations! You correctly follow your BP
monitoring regime for the last month.")
            elif cntF == 1:
                message("green", "Please follow your instructions: "
                    "Measure blood pressure once per week (morning
before breakfast).")

            elif cntF == 2:
                message("yellow",
                    "You haven't taken measurement for blood pressure 2 weeks, "
                    "please measure blood pressure once per day.")
                mailNotification("You haven't taken measurement for blood pressure 2
weeks, "
                    "please measure blood pressure once per day.")

# RA 8
def RA8(token):
    from datetime import timedelta
    HFDDiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    bodyWeightRegime = getValue("BodyWeightSelfMonitoringRegime", token)

    if (HFDDiag['value'] == 'yes' or CKDDiag['value'] >= 'stage2') \
        and bodyWeightRegime['value'] == "1 measurement per day":
        cnt = 0
        cntF = 0
        startDate = carre.getStartDate(7)

```

```

measurements = getValues("BodyWeight", token, startDate, group=True)
p = 0

for i in range(7):
    measurement = str(measurements['date'][p])
    if measurement == str(startDate + timedelta(days=i)):
        cnt += 1
        cntF = 0
        p += 1

    else:
        cntF += 1
        cnt = 0

    if cnt == 7:
        message("green", "Congratulations! You correctly follow your body
weight instructions for 7 days.")
        elif cntF == 1:
            message("green", "Please follow your instructions:Measure body
weight once per day(morning).")
            elif cntF == 3:
                message("yellow", "You have high risk, please measure your body
weight at least once per day.")
                elif cntF == 5:
                    message("yellow",
                        "You haven't taken measurement for body weight 5 days,
please measure body weight once per day.")
                    mailNotification(
                        "You haven't taken measurement for body weight 5 days, please
measure body weight once per day.")
                    elif cntF == 7:
                        message("red",
                            "You haven't taken measurement for body weight 7 days,
please measure body weight once per day.")
                        mailNotification(
                            "You haven't taken measurement for body weight 7 days, please
measure body weight once per day.")
                        smsNotification(
                            "You haven't taken measurement for body weight 7 days, please
measure body weight once per day.")

# RA 9
def RA9(token):
    if M1[3] == 1:
        from datetime import timedelta
        diabDiag = getValue("DiabetesDiagnosis", token)
        BMI = getValue("BMI", token)
        bwsmr = getValue("BodyWeightSelfMonitoringRegime")
        if (diabDiag['value'] == 'yes' or BMI['value']) and bwsmr['value'] == "1
measurement per week":
            cnt = 0
            cntF = 0
            startDate = carre.getStartDate(30)

            measurements = getValues("BodyWeight", token, startDate, group=True)
            p = 0

            for i in range(4):

```

```

        measurement = str(measurements['date'][p])
        if str(startDate + timedelta(days=i * 7)) <= measurement <
str(startDate + timedelta(days=7 + i * 7)):
            cnt += 1
            cntF = 0
            p += 1

    else:
        cntF += 1
        cnt = 0

    if cnt == 4:
        message("green", "Congratulations! You correctly follow your
body weight monitoring regime"
                " for the last month.")
    elif cntF == 1:
        message("green", "Please follow your instructions: "
                "Measure body weight once per week (morning
before breakfast).")
    elif cntF == 2:
        message("yellow",
                "You haven't taken measurement for body weight 2 weeks,
"
                "please measure body weight once per day.")
        mailNotification("You haven't taken measurement for body weight
2 weeks, "
                "please measure body weight once per day.")

# RA 10
def RA10(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    diabDiag = getValue("Diabetes", token)
    BMI = getValue("BMI", token)
    if HFdiag['value'] == 'no' and CKDDiag['value'] < 'stage2' and
diabDiag['value'] == 'no' and BMI['value'] < 25:
        from datetime import timedelta
        cnt = 0
        cntF = 0

        startDate = carre.getStartDate(60)

        measurements = getValues("BodyWeight", token, startDate, group=True)
        p = 0

        for i in range(2):
            measurement = str(measurements['date'][p])
            if str(startDate + timedelta(days=i * 30)) <= measurement <
str(startDate + timedelta(days=30 + i * 30)):
                cnt += 1
                cntF = 0
                p += 1

            else:
                cntF += 1
                cnt = 0

        if cnt == 2:

```

```

        message("green", "Congratulations! You correctly follow your body
weight monitoring regime"
               " for 2 months.")

    elif cntF == 1:
        message("green", "Please follow your instructions: "
               "Measure body weight once per month (morning before
breakfast),preferably once a week.")

    elif cntF == 2:
        message("yellow",
               "You haven't taken measurement for body weight 2 months, "
               "please measure body weight once per month.")
        mailNotification("You haven't taken measurement for body weight 2
months, "
               "please measure body weight once per month.")

# RA 11
def RA11(token):
    diabDiag = getValue("Diabetes", token)
    bgsmr = getValue("BloodGlucoseSelfMonitoringRegime", token)

    if diabDiag['value'] == 'yes' and bgsmr['value'] == "3 measurements per
day":

        from datetime import timedelta, time
        cnt = 0
        cntF = 0

        startDate = carre.getStartDate(7)

        measurements = getValues("BloodGlucose", token, startDate)

        p = 0
        t0 = time(hour=0)
        t1 = time(hour=12)
        t2 = time(hour=18)
        t3 = time(hour=23, minute=59)

        for i in range(21):
            measurement = str(measurements['date'][p]).split("T")
            measurement[1] = measurement[1].replace("Z", ":00")
            if measurement[0] == str(startDate + timedelta(days=i / 3)):
                if i % 3 == 0:
                    if str(t0) < measurement[1] < str(t1):
                        cnt += 1
                        cntF = 0
                        p += 1
                    else:
                        cntF += 1
                        cnt = 0
                elif i % 3 == 1:
                    if str(t1) < measurement[1] < str(t2):
                        cnt += 1
                        cntF = 0
                        p += 1
                    else:
                        cntF += 1
                        cnt = 0
                else:
                    cntF += 1
                    cnt = 0

```

```

        if str(t2) <= measurement[1] < str(t3):
            cnt += 1
            cntF = 0
            p += 1
        else:
            cntF += 1
            cnt = 0

    else:
        cntF += 1
        cnt = 0

    if cnt == 21:
        message("green", "Congratulations! You correctly follow your blood
glucose measurement"
                " instructions for 7 days.")
    elif cntF == 3:
        message("green",
                "Please follow your instructions: Measure blood glucose "
                "three times per day (before breakfast,lunch & dinner).")
    elif cntF == 9:
        message("yellow",
                "You have high risk, please measure your blood glucose at
least three times per day.")
    elif cntF == 15:
        message("yellow",
                "You haven't taken measurement for blood glucose 5 days, "
                "please measure blood glucose at least three times per
day.")

    mailNotification(
        "You haven't taken measurement for blood glucose 5 days, "
        "please measure blood glucose at least three times per day.")
    elif cntF == 21:
        message("red",
                "You haven't taken a measurement for blood glucose for 7
days, please measure blood "
                "glucose three times per day.")
    mailNotification(
        "You haven't taken a measurement for blood glucose for 7 days,
please measure blood "
        "glucose three times per day.")
    smsNotification(
        "You haven't taken a measurement for blood glucose for 7 days,
please measure blood "
        "glucose three times per day.")

# RA 13
def RA13(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    diabDiag = getValue("Diabetes", token)
    BMI = getValue("BMI", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)
    bgsmr = getValue("BloodGlucoseSelfMonitoringRegime", token)

    if diabDiag['value'] == 'no' and (HFdiag['value'] == 'yes' or
CKDDiag['value'] >= 'stage2' or

```



```

BMI['value'] >= 25 or
hyperDiag['value'] == 'yes' or LDLC[
    'value'] >= 130 or TC['value'] >= 200) and bgsmr['value'] == "1
measurement per month":

    from datetime import timedelta
    cnt = 0
    cntF = 0

    startDate = carre.getStartDate(60)

    measurements = getValues("BloodGlucose", token, startDate, group=True)
    p = 0

    for i in range(2):
        measurement = str(measurements['date'][p])
        if str(startDate + timedelta(days=i * 30)) <= measurement <
str(startDate + timedelta(days=30 + i * 30)):
            cnt += 1
            cntF = 0
            p += 1

        else:
            cntF += 1
            cnt = 0

    if cnt == 2:
        message("green", "Congratulations! You correctly follow your blood
glucose monitoring regime"
                " for 2 months.")

    elif cntF == 1:
        message("green", "Please follow your instructions: "
                    "Measure blood glucose once per month (before
breakfast).")

    elif cntF == 2:
        message("yellow",
                "You haven't taken measurement for blood glucose 2 months, "
                "please measure blood glucose once per month.")
        mailNotification("You haven't taken measurement for blood glucose 2
months, "
                        "please measure blood glucose once per month.")

# RA 14
def RA14(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    if HFdiag['value'] == 'yes':
        recNumbOfSteps = getValue("NumberOfStepsRecommendedByDoctor", token)
        if recNumbOfSteps != None:
            numbOfSteps = getValue("Steps", token)
            if numbOfSteps['value'] >= recNumbOfSteps['value']:
                message("yellow", "Please be careful!!! Don't overdo it when
exercising.")
            else:
                message("", "Ask patient: how many steps your doctor has
recommended? no. of steps = X")

# RA 15

```

```
def RA15(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    physActRegime = getValue("PhysicalActivityRegime", token)

    if HFdiag['value'] == 'no' and CKDDiag['value'] == 'stage5' and
physActRegime[
    'value'] == "walk 30 min 3-4 times per week":

        from datetime import timedelta, date

        cnt = 0

        startDate = carre.getStartDate(7)
        steps = getValue("Steps", token)
        if steps['value'] >= 3000:
            message("green", "Congratulations! You reached your goal for
today!")

            measurements = getValues("Steps", token, startDate, group=True)

            for i in range(7):
                measurement = measurements['values'][i]
                if measurement >= 3000:
                    cnt += 1

            if date.today().weekday() == 0:
                if cnt < 3:
                    message("green", "You only walked " + cnt + " days (with 3000
steps) this week. "
                                                                    "Next week, please
try to walk a bit more.")
                elif 3 <= cnt <= 4:
                    message("green", "Congratulations! You reached your goal this
week "
                                                                    "(3000 steps " + cnt + " times this week.")
                elif cnt > 4:
                    message("green", "No need to walk more than 3-4 times per
week.")
                elif cnt == 4:
                    message("green", "You already have 4 days this week with 3000 steps
per day.")

# RA 16
def RA16(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)

    if HFdiag['value'] == 'no' and CKDDiag['value'] < 'stage5' and \
        (CKDDiag['value'] >= 'stage2' or hyperDiag['value'] == 'yes' or
LDLC['value'] >= 130 or TC['value'] >= 200):

        from datetime import timedelta, date

        cnt = 0
        startDate = carre.getStartDate(30)
```

```

steps = getValue("Steps", token)
stepstm = 3000 - float(steps['value'])

if steps['value'] >= 3000:
    message("green", "Congratulations! You reached your daily goal!")
elif steps['value'] >= 2000:
    message("green", "You already have " + steps['value'] + " steps
today."
                                " Please try
more! You have " + stepstm + " steps to achieve your goal.")
    elif steps['value'] >= 1000:
        message("green", "You already have " + steps['value'] + " steps
today."
                                " Please try
more! You have " + stepstm + " steps to achieve your goal.")

measurements = getValues("Steps", token, startDate, group=True)

for i in range(30):
    measurement = measurements['values'][i]
    if measurement >= 3000:
        cnt += 1
day = str(measurements['date'][29]).split("-")

if day[2] == "01":
    if cnt > 20:
        message("green",
                "Congratulations! You were highly active (" + cnt + "
highly active days) this month.")
    elif cnt == 10:
        message("green", "You already have " + cnt + " high active days
this month. Please try more!")
    elif cnt == 20:
        message("green", "You already have " + cnt + " high active days
this month. Please try little more!")

# RA 17
def RA17(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)
    diabDiag = getValue("Diabetes", token)
    BMI = getValue("BMI", token)

    if HFdiag['value'] == 'no' and CKDDiag['value'] < 'stage2' and
hyperDiag['value'] == 'no' and \
        LDLC['value'] < 130 and TC['value'] < 200 and
(diabDiag['value'] == 'yes' or BMI['value'] >= 25):

        from datetime import timedelta, date

        cnt = 0

        steps = getValue("Steps", token)
        stepstm = 6000 - float(steps['value'])

        if steps['value'] >= 6000:

```

```

        message("green", "Congratulations! You reached your daily goal!")
    elif steps['value'] >= 4000:
        message("green", "Bravo! You already have " + steps['value'] + "
steps today. Please try little more!")
    elif steps['value'] >= 2000:
        message("green", "Great! You already have " + steps['value'] + "
steps today."
        "
Please try more! You have " + stepstm + " steps to achieve your goal.")
    elif steps['value'] >= 1000:
        message("green", "You already have " + steps['value'] + " steps and
you are not so far from your goal.")

startDate = carre.getStartDate(30)
measurements = getValues("Steps", token, startDate, group=True)

for i in range(30):
    measurement = measurements['values'][i]
    if measurement >= 6000:
        cnt += 1
day = str(measurements['date'][29]).split("-")

if day[2] == "01":
    if cnt > 20:
        message("green",
        "Congratulations! You were highly active (" + cnt + "
highly active days) this month.")
    elif cnt == 10:
        message("green", "Great! You already have " + cnt + " high active
days this month. Please try more!")
    elif cnt == 20:
        message("green",
        "Bravo! You already have " + cnt + " high active days this
month. Please try little more!")

# RA 18
def RA18(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)
    diabDiag = getValue("Diabetes", token)
    BMI = getValue("BMI", token)

    if HFdiag['value'] == 'no' and CKDDiag['value'] < 'stage2' and
hyperDiag['value'] == 'no' and \
        LDLC['value'] < 130 and TC['value'] < 200 and
(diabDiag['value'] == 'yes' and BMI['value'] >= 25):

        from datetime import timedelta, date

        cnt = 0
        startDate = carre.getStartDate(30)
        steps = getValue("Steps", token)
        stepstm = 9000 - float(steps['value'])

        if steps['value'] >= 9000:
            message("green", "Congratulations! You reached your daily goal!")

```

```

        elif steps['value'] >= 6000:
            message("green", "Bravo! You already have " + steps['value'] + "
steps today. Please try little more!")
        elif steps['value'] >= 3000:
            message("green", "Great! You already have " + steps['value'] + "
steps today."
            "
Please try more! You have " + stepstm + " steps to achieve your goal.")
        elif steps['value'] >= 2000:
            message("green", "You already have " + steps['value'] + " steps and
you are not so far from your goal.")

    measurements = getValues("Steps", token, startDate, group=True)

    for i in range(30):
        measurement = measurements['values'][i]
        if measurement >= 9000:
            cnt += 1
    day = str(measurements['date'][29]).split("-")

    if day[2] == "01":
        if cnt > 20:
            message("green",
                    "Congratulations! You were highly active (" + cnt + "
highly active days) this month.")
        elif cnt == 10:
            message("green", "Great! You already have " + cnt + " high active
days this month. Please try more!")
        elif cnt == 20:
            message("green",
                    "Bravo! You already have " + cnt + " high active days this
month. Please try little more!")

# RA 19
def RA19(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    if HFdiag['value'] == 'no' or CKDDiag['value'] < 'stage2':
        SBP = getValue("SystolicBloodPressure", token)
        DBP = getValue("DiastolicBloodPressure", token)
        if SBP['value'] > 180 or DBP['value'] > 120:
            message("red", "ATTENTION! Take another blood pressure measurement
and "
                    "if it is still high, please go to hospital
immediately.")
            smsNotification("ATTENTION! Take another blood pressure measurement
and "
                            "if it is still high, please go to hospital
immediately.")
            mailNotification("ATTENTION! Take another blood pressure measurement
and "
                             "if it is still high, please go to hospital
immediately.")

# RA 20
def RA20(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)

```

```

hyperDiag = getValue("HypertensionDiagnosis", token)

if hyperDiag['value'] == 'no' and (HFDiag['value'] == 'yes' or
CKDDiag['value'] >= 'stage2'):
    SBP = getValue("SystolicBloodPressure", token)
    DBP = getValue("DiastolicBloodPressure", token)
    startDate = carre.getStartDate(7)
    SBPpw = getValues("SystolicBloodPressure", token, startDate)
    DBPpw = getValues("DiastolicBloodPressure", token, startDate)
    avgSBPpw = getAverage(SBPpw)
    avgDBPpw = getAverage(DBPpw)

    if SBP['value'] > 160 or DBP['value'] > 110:
        message("red", "ATTENTION! Take another blood pressure measurement
and "
                    "if it is still high go to nearest hospital
immediately.")
        smsNotification("ATTENTION! Take another blood pressure measurement
and "
                    "if it is still high go to nearest hospital
immediately.")
        mailNotification("ATTENTION! Take another blood pressure measurement
and "
                    "if it is still high go to nearest hospital
immediately.")

    if 135 <= avgSBPpw <= 145 or 85 <= avgDBPpw <= 90:
        message("yellow", "Check your ankles for fluid detention.")
        mailNotification("Check your ankles for fluid detention.")
        anklesAreSwollen = input("Are ankles swollen?(yes/no)")
        if anklesAreSwollen == "yes":
            message("yellow", "Please call your doctor and reduce salt
intake.")

    elif avgSBPpw > 145 or avgDBPpw > 90:
        message("yellow", "Please visit your doctor and reduce salt
intake.")
        mailNotification("Please visit your doctor and reduce salt intake.")

    elif avgSBPpw < 100 or avgDBPpw < 60:
        message("yellow", "Please visit your doctor.")
        mailNotification("Please visit your doctor.")

# RA 21
def RA21(token):
    HFDiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)
    diabDiag = getValue("Diabetes", token)
    BMI = getValue("BMI", token)

    if hyperDiag['value'] == 'no' and HFDiag['value'] == 'no' and
CKDDiag['value'] < 'stage2' and \
        (BMI['value'] >= 25 or diabDiag['value'] == 'yes' or LDLC['value']
>= 130 or TC['value'] >= 200):

        startDate = carre.getStartDate(30)

```

```

SBPpw = getValues("SystolicBloodPressure", token, startDate)
DBPpw = getValues("DiastolicBloodPressure", token, startDate)

avgSBPpw = getAverage(SBPpw['values'])
avgDBPpw = getAverage(DBPpw['values'])
if 120 < avgSBPpw <= 135 or 80 < avgDBPpw <= 85:
    message("green",
            "Start walking more than half an hour (over 3000 steps) per
day, reduce salt intake and "
            "increase vegetables intake.")
    if diabDiag['value'] == 'no':
        message("green", "Please increase fruits intake.")

startDate = carre.getStartDate(7)
steps = getValues("Steps", token, startDate, group=True)
avgSteps = getAverage(steps)

if avgSteps < 3000:
    message("green",
            "Please increase your daily activity, at least half an
hour walking (over 3000 steps) per day.")

# RA 22
def RA22(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)
    diabDiag = getValue("Diabetes", token)
    BMI = getValue("BMI", token)

    if hyperDiag['value'] == 'no' and HFdiag['value'] == 'no' and
CKDDiag['value'] < 'stage2' and \
        BMI['value'] < 25 and diabDiag['value'] == 'no' or
LDLC['value'] < 130 or TC['value'] < 200:

        startDate = carre.getStartDate(90)

        SBPpw = getValues("SystolicBloodPressure", token, startDate)
        DBPpw = getValues("DiastolicBloodPressure", token, startDate)

        avgSBPpw = getAverage(SBPpw['values'])
        avgDBPpw = getAverage(DBPpw['values'])

        if 120 < avgSBPpw <= 135 or 80 < avgDBPpw <= 85:
            message("green",
                    "Start walking more than half an hour (over 3000 steps) per
day, reduce salt intake and "
                    "increase vegetables intake.")

# RA 23
def RA23(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)

    if HFdiag['value'] == 'yes' or CKDDiag['value'] >= 'stage2':

```

```

startDate1 = carre.getStartDate(1)
startDate3 = carre.getStartDate(3)

bodyWeight1 = getValues("BodyWeight", token, startDate1)
bodyWeight3 = getValues("BodyWeight", token, startDate3)

diffBW1 = getDifference(bodyWeight1['values'])
diffBW3 = getDifference(bodyWeight3['values'])

if diffBW1 >= 1:
    message("yellow", "Check your ankles for fluid detention.")
    mailNotification("Check your ankles for fluid detention.")

    dyspnea = input("Do you have dyspnea?(yes/no)")
    if dyspnea == 'yes':
        message("red", "Please urgently go to hospital.")
        mailNotification("Please urgently go to hospital.")
        smsNotification("Please urgently go to hospital.")

    ankles = input("Are your ankles swollen?(yes/no)")
    if ankles == 'yes':
        message("yellow", "Please call your doctor.")
    else:
        message("green", "Reduce salt intake.")

# RA 24
def RA24(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    diabDiag = getValue("Diabetes", token)

    if HFdiag['value'] == 'no' and CKDDiag['value'] < 'stage2' and
    diabDiag['value'] == 'yes':

        startDate = carre.getStartDate(7)
        bodyWeight = getValues("BodyWeight", token, startDate)
        diffBW = getDifference(bodyWeight['values'])

        if diffBW > 1:
            message("green",
                    "Start Walkin about an hour (e.g. 6000 steps) per day. "
                    "Reduce the amount of food."
                    "Change your diet.")

            startDate = carre.getStartDate(7)
            steps = getValues("Steps", token, startDate, group=True)
            avgSteps = getAverage(steps)
            if avgSteps < 6000:
                message("green",
                        "Please increase your daily activity, at least an hour
walking (e.g. 6000 steps) per day.")

# RA 25
def RA25(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    diabDiag = getValue("Diabetes", token)

```



```

BMI = getValue("BMI", token)

if HFdiag['value'] == 'no' and CKDDiag['value'] < 'stage2' and
diabDiag['value'] == 'no' and BMI['value'] >= 25:

    startDate = carre.getStartDate(7)
    bodyWeight = getValues("BodyWeight", token, startDate)
    diffBW = getDifference(bodyWeight['values'])

    if diffBW > 2:
        message("green",
            "Start Walkin about an hour (e.g. 6000 steps) per day. "
            "Reduce the amount of food."
            "Change your diet.")

    startDate = carre.getStartDate(7)
    steps = getValues("Steps", token, startDate, group=True)
    avgSteps = getAverage(steps)
    if avgSteps < 6000:
        message("green",
            "Please increase your daily activity, at least an hour
walking (e.g. 6000 steps) per day.")

# RA 26
def RA26(token):
    diabDiag = getValue("Diabetes", token)

    if diabDiag['value'] == 'yes':

        startDate = carre.getStartDate(1)
        bloodGlucose = getValue("BloodGlucose", token)
        bloodGlucosepd = getValues("BloodGlucose", token, startDate, group=True)

        if bloodGlucose['value'] <= 70:
            message("red", "Consume 15-20 grams of glucose or simple
carbohydrates. Call an ambulance.")
            mailNotification("Consume 15-20 grams of glucose or simple
carbohydrates. Call an ambulance.")
            smsNotification("Consume 15-20 grams of glucose or simple
carbohydrates. Call an ambulance.")
            if 140 < bloodGlucosepd['values'] <= 200:
                message("yellow",
                    "Change your diet. Do you take your pills? "
                    "Start walking about an hour (e.g. 6000 steps) per day.")
                mailNotification("Change your diet. Do you take your pills? "
                    "Start walking about an hour (e.g. 6000 steps) per
day.")
            elif bloodGlucosepd > 200:
                message("red", "Please call your doctor.")
                mailNotification("Please call your doctor.")

# RA 28
def RA28(token):
    HFdiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    hyperDiag = getValue("HypertensionDiagnosis", token)
    LDLC = getValue("LDL-C", token)
    TC = getValue("TC", token)

```

```

diabDiag = getValue("Diabetes", token)
BMI = getValue("BMI", token)

if diabDiag['value'] == 'no' and (HFDiag['value'] == 'yes' or
CKDDiag['value'] >= 'stage2' or \
                                                                    BMI['value'] >= 25 or
hyperDiag['value'] == 'yes' or LDLC[
'value'] >= 130 or TC['value'] >= 200):

    startDate = carre.getStartDate(30)

    bloodGlucosepm = getValues("BloodGlucose", token, startDate, group=True)
    maxBGpm = bloodGlucosepm['values'][0]
    for i in bloodGlucosepm['values']:
        if maxBGpm < i:
            maxBGpm = i

    if 100 < maxBGpm <= 126:
        message("green", "Change your diet. Start walking about half an hour
(e.g. 3000 steps) per day")
        bloodGlucose=getValue("BloodGLucose", token)

        avgBloodGlucose = getAverage(bloodGlucosepm['values'])

        if bloodGlucose['value'] >avgBloodGlucose:
            message("yellow", "Please visit your doctor.")

    elif maxBGpm > 126:
        message("yellow", "Please visit your doctor")
        mailNotification("Please visit your doctor")

# RA 29
def RA29(token):
    HFDiag = getValue("HeartFailureDiagnosis", token)
    CKDDiag = getValue("ChronicKidneyDiseaseDiagnosis", token)
    AFDiag = getValue("AtrialFibrillationDiagnosis", token)

    if (HFDiag['value'] == 'yes' or CKDDiag['value'] >= 'stage2') and
AFDiag['values'] == 'no':
        startDate = carre.getStartDate(3)
        HRp3d = getValues("HeartRate", token, startDate)
        avgHRp3d = getAverage(HRp3d['values'])
        if avgHRp3d>90:
            bodyWeight=getValues("BodyWeight", token, startDate)
            bloodPressure=getValues("BloodPressure", token, startDate)
            avgBodyW = getAverage(bodyWeight['values'])
            avgBloodP = getAverage(bloodPressure['values'])
            bodyW = getValue("BodyWeight", token)
            bloodP = getValue("BloodPressure", token)
            if bodyW['value'] >avgBodyW or bloodP['values']>avgBloodP:
                message("yellow", "Please call your doctor. You may need ECG
monitoring.")
                mailNotification("Please call your doctor. You may need ECG
monitoring.")

# RA 30
def RA30(token):
    listofRF = []
    for i in carre.risk_factors.keys:

```

```

listofRF.append(i)

from datetime import timedelta, date
cntTM = [] # this month
cntPM = [] # previous month
startDate = carre.getStartDate(60)

for j in listofRF:
    riskFactor = getValues(j, token, startDate, group=True)

    for k in range(len(riskFactor['values'])):
        if riskFactor['date'][k] < str(date.today() - timedelta(days=30)):
            prevMonth = riskFactor['value'][k]
        else:
            thisMonth = riskFactor['value'][k]

        if prevMonth > thisMonth:
            if thisMonth == 'no':
                message("green", "Congratulations! There is no more the risk
factor: " + j + ".")
            else:
                message("green", "Great! There is a decrease in risk factor: " +
j + ".")
        elif prevMonth < thisMonth:
            if prevMonth == 'no':
                message("yellow", "Detected a new risk factor: " + j + ". Please
contact with your doctor.")
                mailNotification("Detected a new risk factor: " + j + ". Please
contact with your doctor.")
            else:
                message("yellow", "Attention! There is an increase in risk
factor: " + j + ".")
                mailNotification("Attention! There is an increase in risk
factor: " + j + ".")

# RA 31
def RA31(token):
    listofEducationalResources = []
    for i in observables:
        listofEducationalResources.append(searchMedlinePlus(i))
        listofEducationalResources.append(searchWikipedia(i))

    for j in risk_elements:
        listofEducationalResources.append(searchMedlinePlus(j))
        listofEducationalResources.append(searchWikipedia(j))

    listofRatedEduRes = rating(listofEducationalResources)

    for k in listofRatedEduRes:
        showToUser(k)

def showToUser(resource):
    print resource

observables = ["Hearth Failure", "Chronic kidney disease"]
risk_elements = ["Anemia", "Asthma"]

```

Annex 3

Design of an interactive GUI for multimedia data exchange using SUR40 multi-touch panel

This annex reproduces the conference paper publication:

R. Kloda, J Piwinski, A Nowak. Design of an interactive GUI for multimedia data exchange using SUR40 multi-touch panel. Systems Control and Information Technologies (SCIT 2016), Warsaw, Poland, 20-21.05.2016

Abstract

Designed interface is a proposed solution for the FP7 CARRE project. The project focuses on the development of medical experts supporting technologies. The most important part of this paper is the evolution of the developed interface, which allows to present and exchange multimedia data to the external devices. The design has been developed to ensure the convenience of usage for both, medical expert and the patient.

The paper also presents basic design guidelines, as well as tools that are available for developers. Overview of components for application development gives an idea of the possibilities and is a good starting point for further exploration of issues associated with the selected multi-touch panel.

1. Introduction

Graphical User Interface (GUI) plays a key role in interaction between human and device. For a common user the quality of the interface is equivalent to quality of the whole product. He uses the interface without thinking about the complicated application architecture. Ergonomics of the interface is the main factor that determines whether the application is seen as useful. Designing the interface should be the first thing that is done while creating a program, since it is the most important element for the user. The user, however, as the recipient of the application, is the most important for the creator [1, 2, 3, 4]. User experience is a new term that has been established in last few years and it is described as a person's total experience using an interactive product. Providing a positive experience while interacting with application became an additional challenge for the designers [2].

The way of designing interfaces changes with the constant development of technology. The invention of touch screens was revolutionary for interface design. Many elements of the ordinary interface can be replaced by gestures. Therefore, the aim is to minimize the interface and to concentrate on the content. Nowadays programs are more intuitive and ergonomic, because the intermediary items like keyboard or mouse are no longer needed.

Designing interfaces for multi-touch screens is a specific challenge. Multi-touch interface has to provide simultaneous access to content for many users. The way of accessing the content has to be intuitive and simple. When it comes to working with electronic devices, this type of interaction has a huge potential in terms of performance, usability and intuitiveness. There are many possible uses of multi-touch screens. Currently the most popular devices with multi-touch screens are tablets and smart phones. Touch screens for laptops and PCs are also being produced. Multi-touch screens can also be used as large-format interactive walls in stores or shopping centers. Touch screen used as an interactive table can significantly improve the cooperation of people working on a common project. Some of the technologies that are being developed, enable objects recognition using specially prepared tags. It gives endless possibilities of interacting with computers using physical objects. This kind of user experience is much more immersive and satisfying than the usual work with a keyboard and mouse.

2. SUR40 multi-touch panel

Samsung SUR40 is an example of a large-format multi-touch screen. It has been developed in cooperation by Samsung and Microsoft. It is provided with PixelSense technology. The device is the size of an average

40" TV screen and thanks to the 360-degree interface lets a group of people use the SUR40 simultaneously. It is possible to create an application that allows one person to present the information to others, or allows a group of people to make a collaborative decision.

2.1 Unit description

Samsung SUR40 with Microsoft PixelSense ships with the AMD Athlon II X2 2.9GHz Dual Core processor and The AMD Radeon HD 6570M desktop graphics card, both of which deliver clear and vibrant visuals. The device has four built-in speakers. The Samsung SUR40 with Microsoft PixelSense utilizes a Full HD LED display. Featuring a large 16:9 40" design with 1920 x 1080 resolutions. Pixel size is 0.46125 x 0.46125 mm. The SUR40 includes the world's largest sheet of Gorilla Glass bonded to any display. The material of the protective layer is touted by the manufacturer as an extremely strong, lightweight and resistant to scratches. One-hour water ingress protection is also featured. The device is 4" thin. It features four USB ports, HDMI port, a Wi-Fi 802.11n router and Bluetooth and Ethernet connections. Samsung SUR40 with Microsoft PixelSense ships with Windows 7 operating system and additional component named Surface Shell. This component is responsible for working in Surface mode [5, 9, 10].

2.2 PixelSense technology

PixelSense is a technology that allows each pixel on the screen to act as a sensor. This enables detection of objects that interact with the device. Integrated PixelSense sensors are built directly into the layers of the LCD screen. These sensors enable detection, identification and reaction to objects with predefined tags and to untagged objects. This technology allows simultaneous identification of 52 touch points [6, 8].

2.3 Microsoft Surface 2.0 SDK

Microsoft Surface 2.0 SDK is intended for developing applications based on Microsoft PixelSense platform. It comprises two development environments .NET 4.0 and XNA. The SDK provides two APIs. The first one is *Presentation Layer* integrated with Windows Presentation Foundation (WPF), which is based on .NET 4.0. Presentation Layer interface extends WPF, adding controls designed for Microsoft Surface multi-touch platform. GUI is designed in XAML markup language and the behavior of the application is programmed in the programming language C#. The second one API is *Core Layer* based on XNA Game Studio 4.0. This programming interface can be used to develop 3D graphic applications [8, 11, 12]. In order to program applications using the Surface SDK, the following software is required:

- Microsoft Visual C# 2010 Express Edition or Microsoft Visual Studio 2010,
- Microsoft .NET Framework 4.0,
- Microsoft XNA Framework Redistributable 4.0,
- Windows 7 operating system.

It should be mentioned that SDK allows writing applications not only for PixelSense devices, but also for computers with a touch-screen and a Windows7 operating system.

2.4 Tagged objects and blobs

Microsoft PixelSense technology enables Samsung SUR40 to recognize tagged and untagged objects. Tag is a marker located on the object has a special pattern (Fig. 1), which thanks to PixelSense technology, can be read using the infrared rays. Markers can be used to identify objects or people.

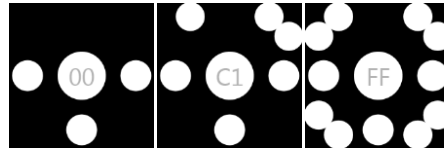


Fig. 1. Examples of Tags [13]

Tags store 8 bits of information, so they can contain 256 different values. Each tag to work properly must have dimensions of exactly 0.75 x 0.75 inches. For the application to react properly to the markers, special control, *TagVisualizer* must be used. The application can be programmed to turn on itself when appropriate marker is placed on the screen. The object on which marker was placed should not reflect infrared rays, to prevent generating additional touch points.

Blobs are objects that do not have tags and are not fingers. These objects are unlabeled. The application can be programmed so that it behaves in a certain way when an unmarked object is detected, for example by displaying a visualization object on the screen [11, 12, 13].

3. General guidelines for interface designing

The main goal in designing an interface is to find a way to explain to user how the application is working. This involves the creation of graphic elements, which are demonstratively illustrating what the application does and how it should be used. The interface should ensure proper communication between the user and the computer [2].

When designing the interface, it is crucial to determine for whom the product is intended. This determines the selection of graphics, colors, and arrangement of interface elements. The basic features of a well-designed interface are following:

- Intelligibility - the interface should be easy to use and understand. Users of intelligible interfaces are less likely to be confused and they work more efficiently.
- Brevity - it is important that the interface does not contain unimportant or repetitive content.
- Familiarity - the interface should use elements and symbols that are obvious to users.
- Accessibility - the interface should work quickly and provide feedback to ensure the user that he/she performed a proper operation.
- Consistency - maintaining consistency allows the user to quickly identify behavior patterns in each following application window.
- Aesthetics - although it is not necessary for the proper operation of the program, it is important because of its impact on the level of user satisfaction. Aesthetic plays an important role in creating a positive user experience.
- Efficiency - interface is designed to increase user productivity.
- Forbearance - the possibility of withdrawal from accidental or unintended actions [2, 4, 14, 15].

3.1 Interface designing guidelines for Surface devices

Microsoft developers team established guidelines for creating GUI on Surface devices. They are designed to make application intuitive, engaging and visually appealing. Many of those guidelines refers to general rules of designing interface:

- Simple - the application must be clear. The way to use it should be obvious to the user. Excessive decorative elements should be avoided.
- Organized - elements should be arranged hierarchically and they should form a consistent structure.
- Content Oriented - information and data are always the core of the application, controls are secondary.
- Dynamic - it is important to take care of move and animation smoothness.

Samsung SUR40 device with PixelSense technology can be attached horizontally or vertically, depending on the intended use of the device. If SUR40 unit is placed horizontally, the interface has to provide possibility of

using the application from all sides, it has to be capable of 360-degrees interaction. Therefore, the following guidelines should be taken into account:

- avoiding elements oriented along one edge of the display,
- enabling the change of elements' orientation by the user,
- orienting the newly opened content in the direction of the person who opened it,
- providing access to any element for every user, regardless of its position in relation to the table,
- ensuring readability of the contents from every side of the device.

The elementary rule of multi-touch interface designing is to use developed standards for operating applications using gestures (eg. changing the size of the item with two fingers, dragging items with one finger).

4. SUR40 as the interactive user interface in CARRE project

4.1 About CARRE project

CARRE is an EU FP7-ICT funded project with the goal to provide innovative means for the management of comorbidities (multiple co-occurring medical conditions), especially in the case of chronic cardiac and renal disease patients or persons with increased risk of such conditions.

Sources of medical and other knowledge will be semantically linked with sensor outputs to provide clinical information personalized to the individual patient, to be able to track the progression and interactions of comorbid conditions. Visual analytics will be employed so that patients and clinicians will be able to visualize, understand and interact with this linked knowledge and take advantage of personalized empowerment services supported by a dedicated decision support system.

The ultimate goal is to provide the means for patients with comorbidities to take an active role in care processes, including self-care and shared decision-making, and to support medical professionals in understanding and treating comorbidities via an integrative approach [7].

4.2 Decision Support System in CARRE project

Decision making in healthcare is a complex process in terms of number of parameters and variables, outcome possibilities and amount of information must be processed.

Decision support systems (DSS) can assist patients and provide to him advices, recommendations and diagnosis of problems in cardiorenal domain, where the optimal solutions for a given sort of data about the possible consequences are determined similar as human experts in the field.

A modern intelligent decision support system not only provides access to data and models. It is also a significant development in the field of analytical data processing, data warehousing and artificial intelligence-aided methods of knowledge discovery in databases i.e. data mining [7].

Our aim was to explore the possibilities of using the SUR40 device and its software as visualization platform in CARRE project.

4.3 Designed interface

The interface was designed for CARRE project purposes and its goal is to improve the communication between patient and internist (or medical expert). The interface was equipped with set of visual elements, which enable the future end user to acquire the medical domain knowledge in more useful as well as user-friendly way which is shown in the figure 2 below. The designed interface can assist patients and doctors by providing advices, recommendations and diagnosis of problems in aging and growing population with chronic

diseases, by means of interactive visualization interface, variables and recommendation to intuitive and user-friendly visualization in patient application.



Fig. 2. Designed interface presented on SUR40 device

The interface has scalable architecture, which enable further development. The interface has been created using rules and guidelines described in subsection 3. It is also possible to present multimedia data and to send it to an external device using Bluetooth protocol. Information about contents are accessible through QR codes generated by application. Designed interface provides to medical expert possibility review all types of data from the hospitals' records and the patient's records together with patient. They can sit side-by-side or across from each other viewing the same information using hand gestures to scroll through, open, zoom, rotate in 3D, push across the table, and drag and drop records from one storage repository to the other.

The proposed interface improve the doctor-patient communication by the use of following features:

- possibility of multiple users interaction with the one application,
- easy manipulation of the elements that display the contents,
- quick and easy access to information,
- transmission of multimedia data,
- intuitive navigation system.

5. Summary

Designed graphical interface supports the exchange of information using multi-touch Surface device. The designed GUI has been implemented for the SUR40 unit. The application has been tested and it is working correctly. It can be used in patient - doctor cooperation. In further development of our interface we consider to add following possibilities:

- animating transitions between windows to make interface more dynamic and consistent,
- enriching applications with sound effects that would indicate an action made by the user. Sounds could also provide feedback on whether an operation is executable,
- currently, the removal of elements is only possible directly from code, so that user can not accidentally remove any important component. There is a possibility of providing extra fields for confirming the intentions of the user, therefore reducing the risk of removing the essential elements. The user will be able to use the interface more freely,
- allowing the user to undo last action,
- placing search engine in the application,
- extending application database.

Acknowledgment

This work was supported by the FP7-ICT project CARRE (No. 611140), funded in part by the European Commission.

References

1. Kuliński M.: Obiektywne i subiektywne czynniki jakości użytkowej graficznych interfejsów użytkownika. Praca doktorska, 2007.
2. Sikorski, M.: Interfejs użytkownika: od pracy, przez emocje, do relacje. Unpublished paper presented at Interfejs użytkownika - Kansei w praktyce Conference, Warszawa 2006. Warsaw.
3. Najmiec A.: Ergonomia oprogramowania – od przepisów do praktyki. Bezpieczeństwo Pracy – Nauka i Praktyka, 5/2002.
4. Fadeyev D.: Interfejsy użytkownika w nowoczesnych aplikacjach webowych. The Smashing Book #1, Helion, 2009.
5. <http://www.samsung.com/>
6. <http://www.microsoft.com/en-us/pixelsense/>
7. <http://www.carre-project.eu/>
8. Designing and Developing Microsoft Surface Application, <http://www.microsoft.com/en-us/pixelsense/training20/index.html>.
9. Samsung SUR40 – Specification Sheet.
10. Samsung SUR40 – User Manual.
11. Developing Applications for Microsoft Surface 2.0, 2011.
12. Microsoft Surface 2.0 Design and Interaction Guide, 2011.
13. Tagged Object Integration for Surface 2.0, 2012.
14. Inchauste F.: Widoczne kontra niewidoczne. The Smashing Book #2, Helion, 2011.
15. Chapman C.: The Smashing Idea Book: From Inspiration to Application, Helion, 2011.