Software Design Description

Preventive Domotics - SOAP Interface

ABSTRACT: THIS DOCUMENT DEFINES THE SOFTWARE IMPLEMENTATION TO BUILD A SOAP INTERFACE ON OUR MAIN PROJECT ON PREVENTATIVE DOMOTICS (DOMOPREV).

KEYWORDS: OM2M, PREVENTION, ROBOT, SMART SENSORS, ELDERLY ASSISTANCE, ADREAM

AUTHOR	PROJECT MANAGER	PRODUCT OWNER
M.Berthe S.Fahmi R.Gautier K. Persand		LAAS CNRS

SDD

DomoPrev - SOA Interface

General information/recommendations

A SDD provides a representation of a software system created to facilitate analysis, planning, implementation, and decision making. It is a blueprint or model of the software system. The SDD is used as the primary medium for communicating software design information.

The SDD shows how the software system will be structured to satisfy the requirements identified in the software requirements specification (e.g. SRS IEEE 830). It is a translation of requirements into a description of the structure and behaviour of the software product/system, the software components, the interfaces, and the data necessary for implementing the software solution.

In a complete SDD, each requirement must be traceable to one or more design entities.

This template can be used directly or it can be adapted in order to better fit the followed software design methodology.

The structure of this document follows the IEEE 1016 standard.
You can find more information about SDDs at http://en.wikipedia.org/wiki/Software_Design_Description or http://www.cmcrossroads.com/bradapp/docs/sdd.html

Revision History

Version	Date	Author	Change Description
1.0		M.Berthe S.Fahmi R.Gautier K. Persand	Initial Document

TABLE OF CONTENTS

1. INTRODUCTION

1. *Purpose*

The DomoPrev project aims to provide assistance to the elderly by using the data of several smart devices in order to produce prognosis for any kind of disease. The SOA interface will focus on providing access to the vital data of the user, our diagnosis, controlling the robot, and configuring the alarm system.

2. *Scope*

This document details the implementation of each software part the SOA interface is going to rely on. It describes the constraints, the interfaces and then the inner configuration of the SOA interface.

This document is intended for people who are going to work/are working on this project, so that they can understand our design and our goals. It is also intended for teachers and students, who want to supervise our project.

3. *Definitions, Acronyms, and Abbreviations*

Term/Acronym	Definition
IoT	Internet of Things
OM2M	Implementation of IoT communication protocol oneM2M by the LAAS

2. References

PMP-SOAP; 18/11/2016; INSA 5ISS Domotic Prevention

SRS-SOAP; 29/11/2016; INSA 5ISS Domotic Prevention

3. SYSTEM ARCHITECTURAL DESIGN

1. *System Description*

The system will implement the following use case:

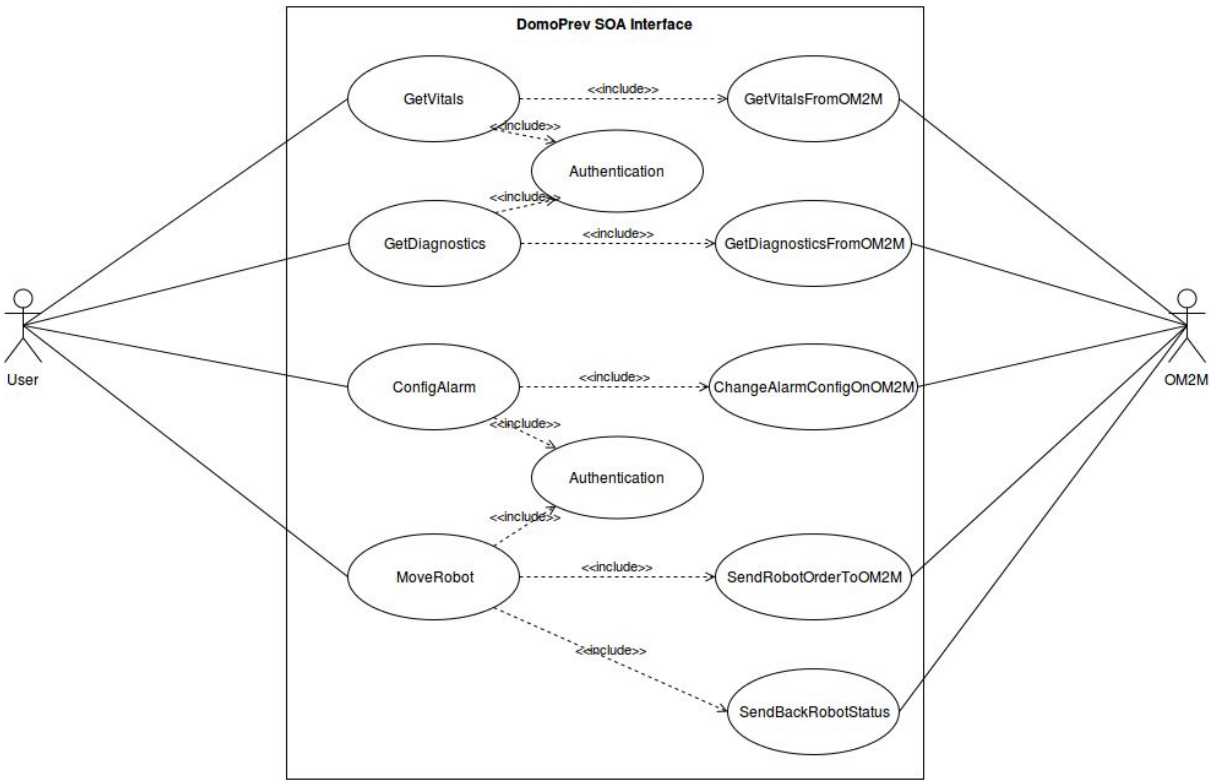


Figure 1: Use case for SOA interface

In figure 1, we can see four main functionalities that the user can access. Each of them will need to pass an authentication step, in order to bolster the security of the application. The data that will be handled by the SOA interface are of a sensitive and private nature, hence access to these data should be highly secured.

Once the authentication step completed, the interface will then call the wanted function, either to retrieve data from OM2M, or to write new orders on it. In the case where we send orders to the robot, we make sure to send back the status of the robot (SendBackRobotStatus).

2. *System Architecture*

The system can be described with a client/server architecture, with the SOA interface as the server interface. A Java application (client) will use the SOA interface (running on the server side) and interact with OM2M.

Another possibility is to host a website on the same server, so that the user can use the SOA

interface's method from his browser. This type of implementation can be considered as a MVC design pattern. We choose not to implement this architecture for schedule purposes.

3. *Design Constraints*

1. General constraints

To ensure the privacy of the collected data, the SOA interface will implement an authentication step before executing each request, for security purposes.

When controlling the robot, the user shall be able to access the actual state of the robot in real-time.

2. Hardware constraints

N.A.

3. SW Constraints

The different web services will be implemented using Java 1.6, so an environment having JDK-6 installed will be necessary.

The interface we offer is compatible with OM2M 1.0.0, as of december 2016.

Services will be built using BPEL based on the respective WSDL file of each component. The WSDL version is 1.0.

4. Components description

1. *Introduction*

The DomoPrev system provides several features that the user can simply use by choosing an item in the following menu:

- Get Vitals
- Get Diagnosis
- Move Robot
- Configure alarms.

2. *Decomposition description*

The following figure highlights the key components of our SOA interface which will deliver different functionalities to the user:

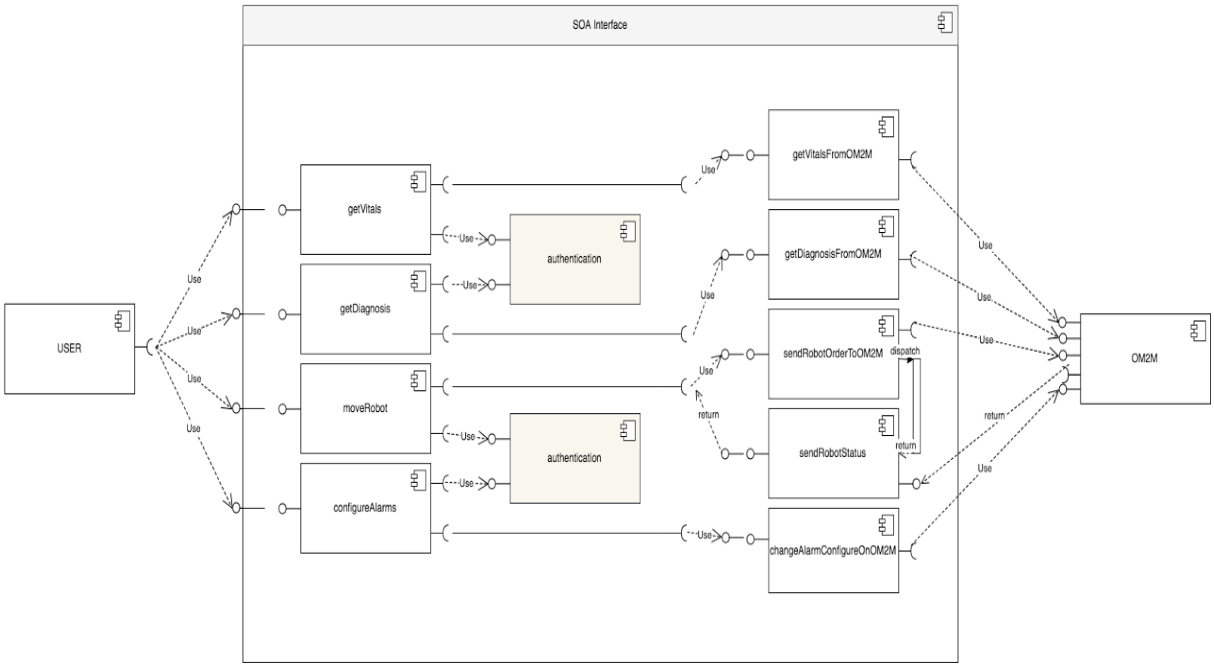


Figure 2: Components Diagram

1. **SOA interface component:**

The SOA interface is the component which interacts with the user. Since our project does not require lots of user interactions, this component is not complexly structured. It has a simple interface and simple purpose. With this component, users can manage to get both vital and diagnosis data needed, or simply send commands to move the robot or configure alarms after authentication.

The SOA interface component delivers the web services below:

getVitals: sends request to get the vital data needed from the OM2M platform via getVitalsFromOM2M

.getDiagnosis: sends request to get diagnosis stored in the OM2M platform via getDiagnosisFromOM2M.

moveRobot: simplified functionality that enables sending commands to robot via sendRobotOrderToOM2M.

configureAlarms: allows to set the alerting parameters such as designating number or email to contact in emergency cases.

2. User component:

User can either be the patient or a specific physician who is given exemptions to use the application that allows to give medical assistance to the patient by giving instructions and sending commands via the SOA interface.

5. External interfaces

1. *Introduction*

This section summarises the external interfaces (i.e. all interfaces not developed throughout the scope of the SOA project).

2. *User interfaces*

1. Java application

The Java application which interacts with our system, is a user interface which will be developed within another project development phase. It will be a basic interface relaying the user's commands to the web service concerned (i.e. the services being developed in this project), which will in turn invoke the required OM2M commands. The Java interface should be able to use invoke the following web services:

- getVitals
- getDiagnosis
- moveRobot
- configureAlarms

3. *External system interfaces*

1. OM2M Service Platform

Our main external interface for this project would be the OM2M platform. This entity allows us to interact with the system whilst providing an abstraction layer with respect to the different components' own communication specifications. All commands to be sent to the system will be sent to the OM2M server, and likewise for the data to be retrieved. OM2M provides uses a REST API and implements oneM2M standards. The OM2M communications used in this project should hence, be fully compliant with OM2M 1.0.

6. Detailed design

1. *Introduction*

The following sequence diagrams illustrate the general interactions between the different

services implemented and the external interfaces.

1. getVitals

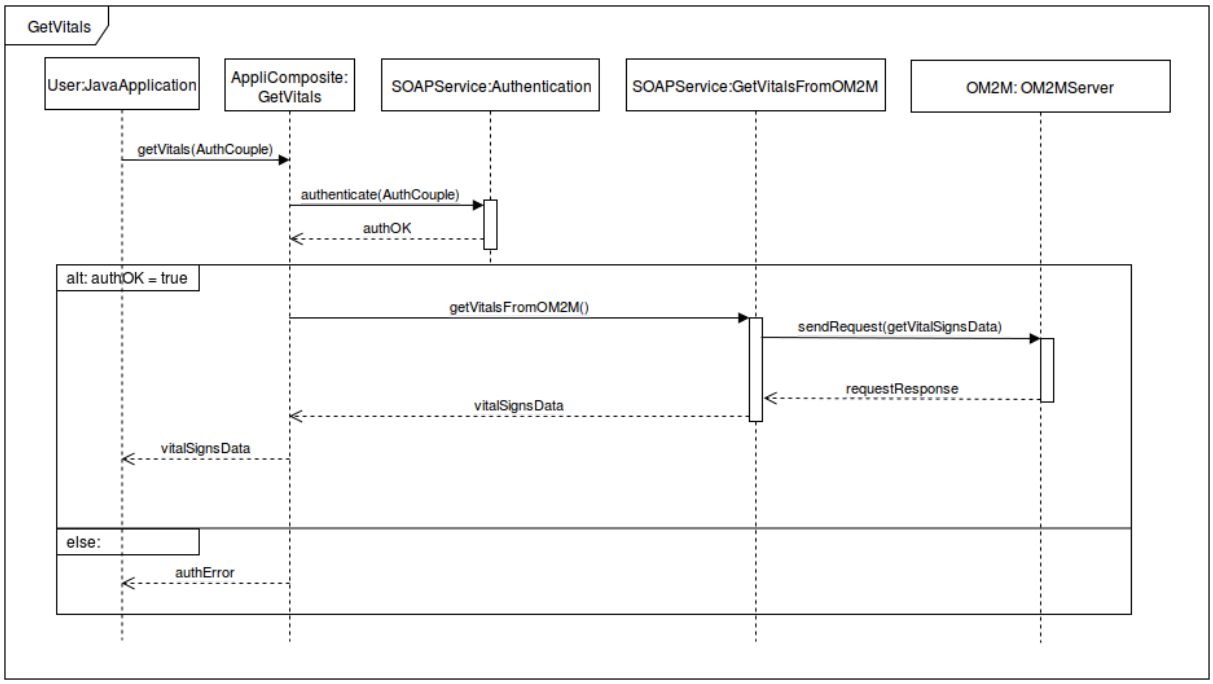


Figure 3: getVitals's sequence diagram

2. getDiagnosis

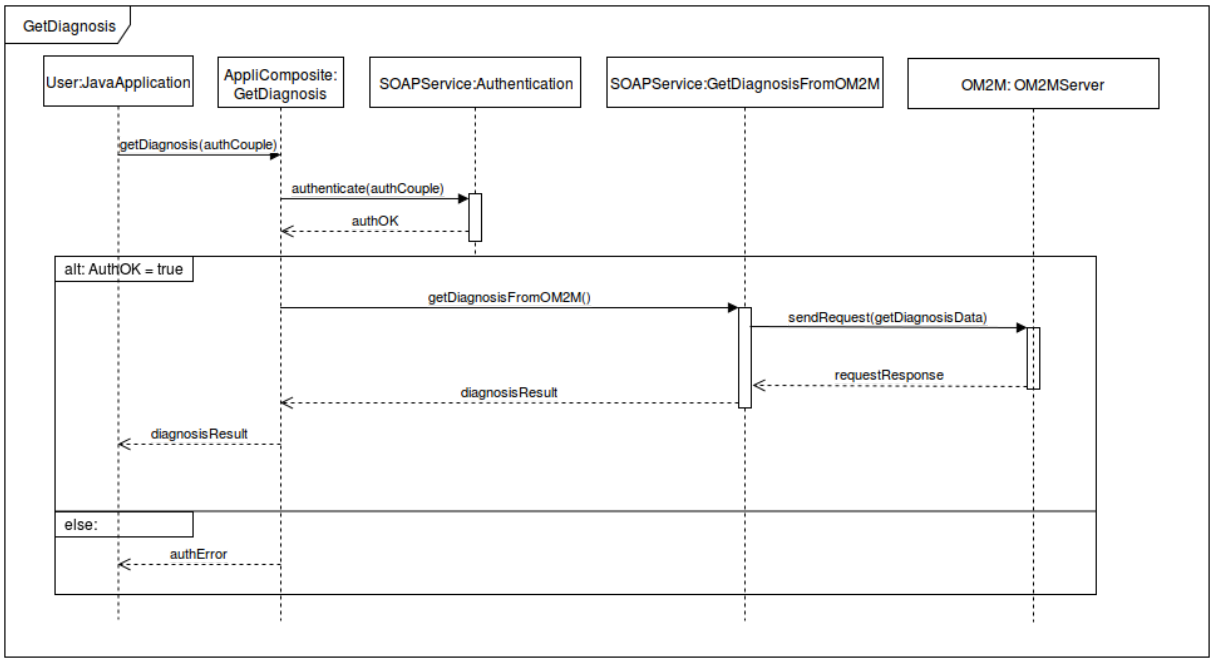


Figure 4: getDiagnosis's sequence diagram

3. moveRobot

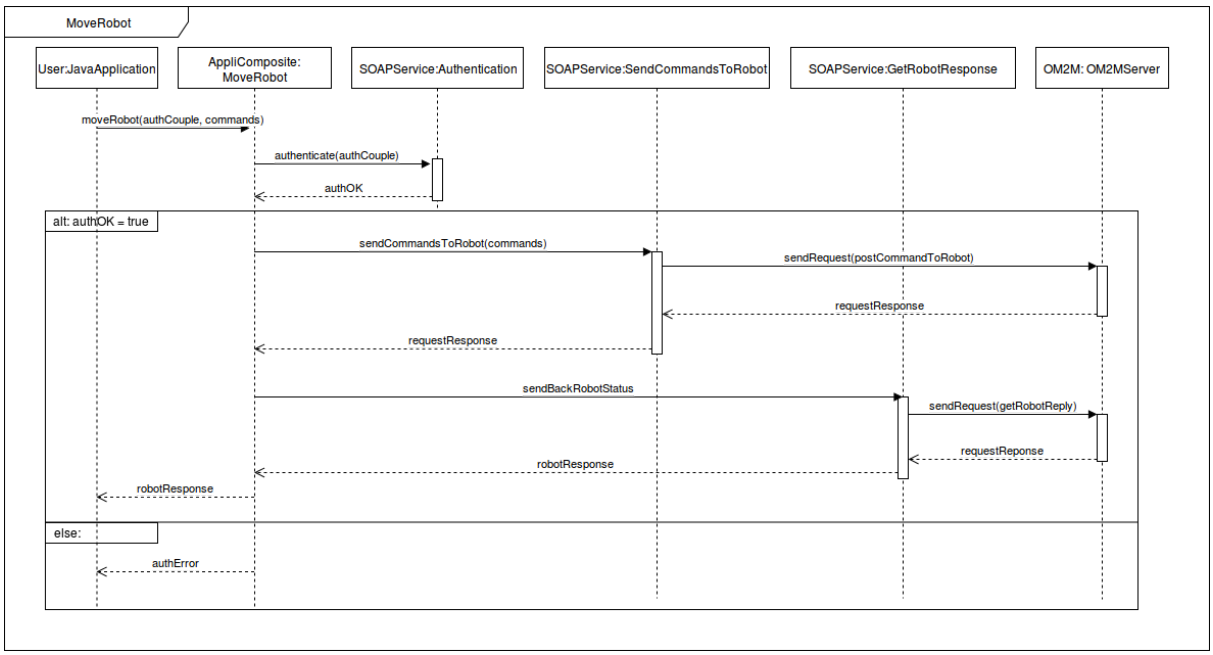


Figure 5: moveRobot's sequence diagram

4. configureAlarms

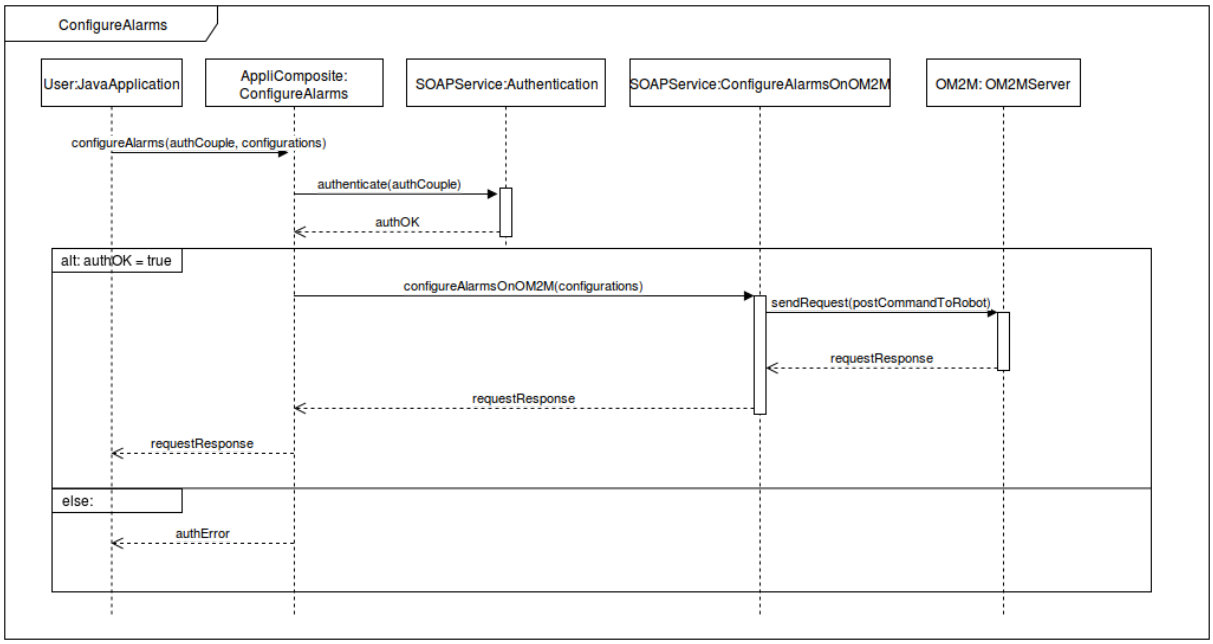
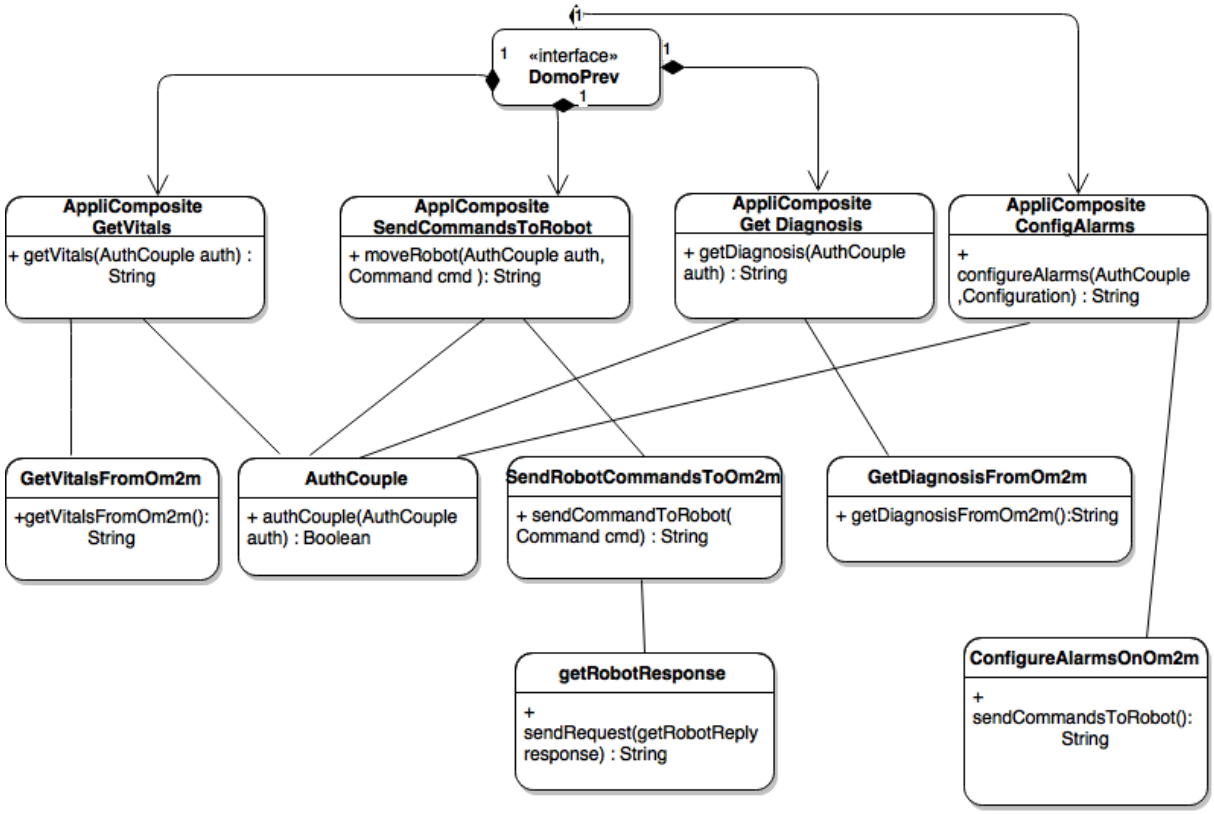


Figure 6: configureAlarms's sequence diagram

Class Diagram

This class diagram gives an overview of what we are going to implement. We have an interface which will be the WSDL with four composites classes. Each composite class has a BPEL and contains a specific service.



7. Annexes

1. *Traceability*

This project will be implemented using Git, so that each modification can be traced back and reverted.