

Trackme RASD

REQUIREMENTS ANALYSIS AND SPECIFICATION DOMAIN (RASD)

Release date: 11/11/2018

Version: 0.1 (usiamo 0.xx per indicare le bozze, la 1 sarà per la definitiva)

Come table of content propongo il sommario, che poi sarà pure quello che intende il prof. Come font propongo Georgia o Calibrì light (come quello del progetto di reti logiche), come tema il terzo da sinistra di office. Grazie ciao

Table of content

[controllare anche il server [la communcazione??] 3](#_Toc529569455)

[Introduction 3](#_Toc529569456)

[Purpose 3](#_Toc529569457)

[Purpose of the document 3](#_Toc529569458)

[Purpose of the project (presentation of goals) 3](#_Toc529569459)

[Scope 4](#_Toc529569460)

[Analysis of the world and the shared phenomena 4](#_Toc529569461)

[World phenomena 5](#_Toc529569462)

[Shared phenomena 5](#_Toc529569463)

[Machine phenomena 5](#_Toc529569464)

[Goals for both services 6](#_Toc529569465)

[Goals for Data4Help 6](#_Toc529569466)

[Goals for AutomatedSos 6](#_Toc529569467)

[Definitions, acronyms, Abbreviations 6](#_Toc529569468)

[Definitions 6](#_Toc529569469)

[Acronyms 7](#_Toc529569470)

[Abbreviations 7](#_Toc529569471)

[Revision history 7](#_Toc529569472)

[Reference Documents 7](#_Toc529569473)

[Document structure 7](#_Toc529569474)

[Overall description 7](#_Toc529569475)

[Product perspective 7](#_Toc529569476)

[Class Diagram 7](#_Toc529569477)

[Product functions 7](#_Toc529569478)

[Both Data4Help and AutomatedSos: Registration 7](#_Toc529569479)

[Data4Help: Data transmitting dealing with the privacy issue 8](#_Toc529569480)

[Data4Help: Data presentation and new data 8](#_Toc529569481)

[AutomatedSos: fast emergency handling and conservative approach 8](#_Toc529569482)

[User characteristics 9](#_Toc529569483)

[Actors 9](#_Toc529569484)

[Assumptions, dependencies and constraints 9](#_Toc529569485)

[Domain Assumptions 9](#_Toc529569486)

[Assumptions and constraints 9](#_Toc529569487)

[Specific requirements 10](#_Toc529569488)

[External Interface Requirements 10](#_Toc529569489)

[User interfaces 10](#_Toc529569490)

[Hardware Interfaces 10](#_Toc529569491)

[Software Interfaces 11](#_Toc529569492)

[Communication Interfaces 11](#_Toc529569493)

[Functional Requirements 11](#_Toc529569494)

[[G1] Provide a form of unique identification (registration/login) of all the clients of the application 11](#_Toc529569495)

[[G2] Allow the user to avoid being associated to his data without his permission 11](#_Toc529569496)

[[G3] Allow third parties to request access to data of some specific individuals 11](#_Toc529569497)

[[G4] Allow third parties to request access to anonymized data of groups of individuals 12](#_Toc529569498)

[[G5] Allow third parties to subscribe to new data 12](#_Toc529569499)

[[G6] Allow third parties to obtain the most adapt data for their needs 12](#_Toc529569500)

[[G7] Third parties are alerted, whenever a user is in danger of life, the application is working properly and there is internet connection 12](#_Toc529569501)

[[G8] If the user’s health status is not clear due to malfunctions, an Emergency number is alerted within an hour 12](#_Toc529569502)

[[G9] The user can temporarily suspend *AutomatedSos* in special cases 13](#_Toc529569503)

[Non-functional Requirements 13](#_Toc529569504)

[Accuracy 13](#_Toc529569505)

[Performance 13](#_Toc529569506)

[Availability and reliability 13](#_Toc529569507)

[Security 14](#_Toc529569508)

[Maintainability 14](#_Toc529569509)

[Design for reuse 14](#_Toc529569510)

[Design Constraints 14](#_Toc529569511)

[Standards compliance 14](#_Toc529569512)

[Hardware limitations 14](#_Toc529569513)

[Any other constraint 14](#_Toc529569514)

[Scenarios 14](#_Toc529569515)

[Scenario 1- Case of emergency 14](#_Toc529569516)

[Scenario 2-Application breakdown 15](#_Toc529569517)

[Scenario 3-. Sensor breakdown 15](#_Toc529569518)

[Scenario 4-Data anonymization and data presentation 16](#_Toc529569519)

[Scenario 5-subscribe to new data 16](#_Toc529569520)

[Scenario 6-Personal Use 17](#_Toc529569521)

[UML modelling 17](#_Toc529569522)

[Formal analysis using alloy 17](#_Toc529569523)

[Effort spent 18](#_Toc529569524)

[References 18](#_Toc529569525)

# controllare anche il server [la communcazione??]

# Introduction

## Purpose

### Purpose of the document

This is the Requirements Analysis and Specification Document of the TrackMe project. Its purpose is to provide a complete description of the system to develop in order to build up the services *Data4Help* and *AutomatedSoS*. In concrete, this means to identify the goals and (most important of all) the requirements (both functional and non-functional), to model the system and the portion of reality it is going to affect in a formal, logic and unambiguous manner, to present all the most common scenarios and use cases, to show relevant constraints and issues.

This document is intended to be a binding yet useful guide for stakeholders, project managers, developers, analysts and testers.

### Purpose of the project (presentation of goals)

TrackMe wants to develop two software-based services, *Data4Help* and *AutomatedSoS.*

The main goal of *Data4Help* is to provide data collected by the subscribed users to subscribed third parties, helping companies in their business, together with respecting users’ privacy. By and large, main beneficiaries of the service are the third parties, which can make use of some useful tools in examining data: a big number of constrains and parameters helps to define more specific queries, and there are many user-friendly ways to show data. Third parties can also choose between asking for data of specific individuals or of groups of people.

The two services are not independent each other, but *AutomatedSos* is built on top of the first one: this means that it’s designed as an additional feature which is implemented after Data4Help. Nonetheless, the two offered services differ each other: through Data4Help TrackMe collects data about the position and the health status just to extract useful information for companies of various types, while AutomatedSos is supposed to allow, through data about health status, third parties to provide a medical assistance in case of emergencies. The assistance is defined as non-intrusive, which means that the service must not affect in a negative way their lives, and personalized, which means that the system is able to set personalized thresholds for each (kind of) user: when parameters go under or below these thresholds, assistance is provided. As a conservative approach, when we are expected to provide the service, but we are not able to do that, we want to inform an emergency number.

## Scope

### Analysis of the world and the shared phenomena

First of all, with Data4Help it wants to support third parties to get data about people’s health status and location. In order to make data available to third parties, it aims at collecting all data about health status from a sensor (for further details about this topic, see *Definitions, Health status).*  The sensor, which is external to our system, is supposed to be a wearable device connected via Bluetooth to a smartphone or integrated in the device where the app runs (e.g. a smartwatch).

Third parties could be interested in data of a single user or of a group of people, specifying different constraints (e.g. geographical). They also may want to know elaborated data or statistics (e.g. average, maximum, media, sum…). It may happen that a third party is also particularly interested in some data that are not currently available (because they regard a period in the future, because they have not elaborated yet…), or in periodic updates of the same information. Nonetheless, all of them are not interested in real time updates.

The system is concerned in guaranteeing in every moment the privacy of the users and knowing always all the people online to struggle misuses:the violation of users’s privacy occurs when a third party is in control of a user’s data whitout his permission, and misuses occur when someone using the application is not correctly identified (the case when he is using a non-official version of the application itself is not considered, cfr *Domain Assumptions*).

The second service offered by TrackMe is *AutomatedSos,* which is in some way inspired from Data4Help and is implemented afterwards. Indeed, having access to health status values of almost all the population, TrackMe realises that it could be useful to provide to elderly people (or with health problems in general) an automated assistance service. *AutomatedSos* exploits all data detected by *Data4Help* making a different use of them: it compares them with personalized thresholds to check whether an immediate emergency is occurring.

Although it, in practise, needs the same permissions and of *Data4Help*, ad additional registration is required, both for third parties and users: only the interested members should be addressed by this service, to avoid misuses.

When values go below of over these thresholds, all the subscribed third parties receive an alarm: they can ack the message and immediately send the necessary help to the user (i.e, an ambulance). Of course, together with the alert the system communicates the user position (relying on GPS service) and the health values detected, so that the third parties are able to send the most appropriate and fastest assistance.

Since it is necessary to consider the enormous importance of the efficiency of the service, we may want to detect and handle the cases where either the sensor, the application or the network does not provide appropriate data o does not send data at all (i.d. the system is not able to monitor the user’s health status). The case in which GPS is not working properly is not considered (*cfr Domain Assumptions).*

We imagine that, although health is of course a key issue for everyone, there could be anyway some moments where the user needs to switch off the physical device (e.g. for charging, or maybe in the swimming pool, or in the mountains, or on an airplane, or at a party or in some other special cases): though they are rare, they are relevant for our modelling, too.

To summarize all what we have exposed, in the following chapters there can be found some lists.

### World phenomena

* Diseases [only emergencies in our modelling] /absence of diseases (in both cases there is relevant data creation)
* Unknown health status
* GPS does not work [not of our interest, *cfr Domain Assumptions*]
* Downloading the app on the smartwatch or smartphone from the official store
* The user must switch off the device/special cases
* An ambulance is alerted (in our modelling, the same as *A third party is alerted)*
* Misuse of the service
* The users’ privacy is violated
* Third parties’ interest in some specific data
* Third parties’ interest in offering an emergencies medical assistance
* [only for Data4Help] Third parties differ each other (i.e.: there is no assumption we can do which is valid for all them)
* Previous relevant medical history
* Unique identification of clients (i.e. associating every operation to a client’s profile)

### Shared phenomena

[Controlled by the world]

* Detection of good health values (above thresholds)
* Detection of bad health values (below thresholds)
* Request for data from the third part
* Request for downloading data from the third party
* Third party’s taking charge of an emergency
* Confirmation of good health status by the user
* Registration to the service by a user or a third party
* Setting the device non-active/active for *AutomatedSos*
* User registration (for *Data4Help* or for *AutomatedSos*)
* Third party registration (for *Data4Help* or for *AutomatedSos*)

[The following ones have been put here and not in machine phenomena because, in our modelling, they can be in some ways detected by the server or other components of our machine]

* Sensor breakdown (not detection of correct data)
* Application breakdown/application working properly
* Network breakdown/network working properly

[controlled by the machine]

* Sending/showing data to the third parties
* Sending an alert to third parties
* asking the user to confirm his health status
* asking to a user permission for individual personal data

### Machine phenomena

* Database queries
* Database inserts
* Database creation
* Data analysis and comparation with thresholds
* Communication between the application and the server: i.e., sending and receiving messages between the app and the server
* Data elaboration for showing
* Data storing by the application
* Threshold calculation for each user
* Anonymizing data

Having classified the phenomena of interest, we want in the following chapters to state formally all the goals.

### Goals for both services

* [G1] Provide a form of unique identification (registration/login) of all clients of the services;
  + [G1.1] Provide a form of unique identification (registration/login) of all users using the services
  + [G1.2] Provide a form of unique identification (registration/login) of all third parties using the services

### Goals for Data4Help

* [G2] allow the user to avoid being associated to his data without his permission
* [G3] Allow third parties to request access to data of some specific individuals
* [G4] Allow third parties to request access to anonymized data of groups of individuals
* [G5] Allow third parties to subscribe to new data
* [G6] Allow third parties to obtain the most adapt data for their needs

### Goals for AutomatedSos

* [G7] Third parties are alerted, whenever a user is in danger of life (PERSONALIZED), the application is working properly and there is internet connection
* [G8] If the user’s health status is not clear due to malfunctions, an Emergency number is alerted within an hour
* [G9] The user can temporarely suspend AutomatedSos in special cases (NON-INTRUSIVE)

## Definitions, acronyms, Abbreviations

### Definitions

* “***Health status***”: when in the following parts we state “health status” we are meaning the following values:
  + Blood saturation: it’s an indicator of the status of lungs and of respiratory system in general (e.g. to detect suffocation)
  + Heart rate: it’s an indicator of heart diseases (to detect heart attacks)
  + Blood pression: it hardly ever helps to detect an emergency, but it’s useful for third parties and statistics (blood pressure out of range can indicate/cause a huge number of chronical diseases)
  + Body temperature: it’s an indicator of fever
  + User’s falling: if the user has suddenly fallen there could be various causes and effects that, though other values are not able to detect them, put in serious risk the user’s life.
* ***“Data Anonymization”***: deleting the fiscal code associated to every data tuple obtained by the query;
* ***“Tresholds”***:
* ***“Active/ non-active”:***
* ***“Client”:*** everyone using the service;
* ***“feasible constraints”:***  simple constraints which is possible to set in a data manipulation language (e.g. SQL), and in which there are no logical contradictions.

### Acronyms

* RASD: Requirements analysis and specification domain
* U: user
* TP: third party

### Abbreviations

* [G-n]: n-goal
* [D-n]: n-domain assumption
* [R-n]: n-functional requirement

## Revision history

This is the first release of the document.

## Reference Documents

* Specification Document “A.Y. 2018-2019 Software Engineering 2 Mandatory project”

## Document structure

# Overall description

## Product perspective

### Class Diagram

## Product functions

With respect to the expressed goals, we state hereunder a more precise description of the most relevant functions of our application.

### Both Data4Help and AutomatedSos: Registration

After downloading the app, users must provide all their data, which means medical history (in order to improve the quality of the analysis) and personal information (in Italy, we accept the fiscal code), through a registration form. In this way all the users can be unique identified. All third parties must be registered to the system too, to make them recognizable by the system and by the users when they request for individual data.

### Data4Help: Data transmitting dealing with the privacy issue

When data about a group of people are requested, the system handles the request keeping always in mind the privacy issue: because of that, it anonymizes all data shared in order to avoid the third parties to identify the single users who are members of the required group.

When the users involved in the request are few, there is the concrete risk that the third party manages to discover the names of the subjects, although their personal information have been hidden. Consequently, the application denies the request. In concrete, it is forced that a query should involve at least 1000 people.

When a third party asks for data regarding a specific individual, the system asks for his permission: if he denies, data are not shown, and the query is refused. If the accepts the system should be able to retrieve his specific data and send them without anonymizing.

### Data4Help: Data presentation and new data

*Data4Help* and *AutomatedSos* are supposed to communicate with a huge number of third parties, but we can’t realistically force our system to deal with so many external softwares (the ones of all the third parties), considered that the they could have been developed for different purpose ( the third parties can belong to different markets, *cfr Assumptions*) : it may even happen that the third party does not have any informative system at all! Due to this, we feel necessary to develop a user interface (e.g. a web app) which all third parties’ managers can use to visualize data or download them in a limited format (e.g. Excel document). To deal with that, our system must provide some forms of understandable presentation of data (e.g. not only tuples), which would have been done by external sofwares if we had made a different modelling of the word.

The user should also the possibility to express some preferences in data and to be informed by the system when there are produced: this means the system must create a profile of preferences for each third party.

### AutomatedSos: fast emergency handling and conservative approach

When the app reports to the system an emergency, the system itself sends a notification to all third parties who are subscribed to AutomatedSOS; the first third party that responds to the alert is the one who takes charge of the emergency. When this happens, the emergency is marked as “handling” by the system, and the notification disappears from all the other third parties.

Concerning the conservative approach, though in the *Purpose* we stated that *AutomatedSos* is non-intrusive, we must implement a way to detect breakdowns.

For this purpose, some regular intervals of time are established, after which the application expects input data. If they are not available (or not readable, or absurd), it can conclude that something is not working properly, and takes the appropriate countermeasures: it asks for confirmation of good health status to the user, and, in case of no answer, it contacts an emergency number.

For the same reason, although we are aware that we are forcing a strict machine constrain, we state that the all the applications should send data for back up purpose to the system every hour: if an app misses these periodic deadline, the system realizes that it is not working correctly (for various reasons: physical damage, misuse, etc.) and sends immediately a message to the emergency number.

To give a non-intrusive assistance and handle those special cases where the user would prefer to switch off the service, the system gives the user the opportunity to set manually the device as “non-active”, to stop the *AutomatedSos* service and avoid improper detection of malfunctions of the system.

## User characteristics

### Actors

* Users (only for *AutomatedSos*): the person who download *AutomatedSos* from the app store, wearing the device and allowing the application to monitor his health status and to manage his data;
* Third party: a company which is interested in monitoring population’s health status and obtaining a useful resource of data (e.g. a health insurance, a pharmaceutical company, the government, an hospital);

## Assumptions, dependencies and constraints

The given description of the problem appears to be incomplete and ambiguos: due to this, we feel necessary to make the following feasible assumptions

### Domain Assumptions

* [D1] The client has correctly downloaded the application [i.e. he is not using a crack version]
* [D2] There is internet connection when the request is submitted
* [D3] The third party knows the fiscal code of the specific individual
* [D4] There is internet connection when the request is submitted
* [D5] Third parties express realistic constrains
* [D6] GPS always works properly indicating the user’s position
* [D7] For every location, there is at least a third party which is able to handle the emergency
* [D8] The user sets on “active” the application every time he feels he could need medical assistance

### Assumptions

* Data4Help is a software-based service included in an application which is presented to the public to have a different purpose (e.g. a pedometer, a diet monitoring application, a simple application to register health data) for marketing reasons. Otherwise, there is no clear advantage for the user to register to the service (which is possible, but unlikely). This happens many times in business models: the goal of the application has nothing in common with the use the people do of it.
* On the other hand, AutomatedSos is required (by the problem specification) to rely on *Data4Help’s* data but is deployed as an independent application (cfr. the next text assumption) with no other “marketing purposes”.
* The registration to Data4Help is a necessary condition to register to AutomatedSos, but not all the third parties and the users registered to Data4Help are necessarily registered to AutomatedSos (e.g. all the companies which are interested only in data about location can avoid register to AutomatedSos, because they may have no way to handle emergencies in any case, or a user has no reason to be concerned about his health status): In practise, when a user downloads *AutomatedSos* he is asked to log in *Data4Help* or, if he does not have an account, to register to *Data4Help.*
* All users are identified through their fiscal code (which is also their username), while all third parties are identified through their official email
* When talking about health status we mean some specific parameters, which could be found in the section “*Definitions”*
* The personalized thresholds are calculated by the system and are based on age, gender and clinical history inserted by the user. The algorithm to calculate thresholds has been elaborated with the agreement of a medical equipe.
* The problem specification does not clarify who must take charge of the ambulance service, assuming it is very unlikely that TrackMe implements itself the medical assistance. Among other solutions, we feel that the third parties are the main focus of our project, and they are the ones who are most interested in providing a medical assistance: they are the most likely candidates to handle this service. In addition to this, the assumption that the 911 service (118 in Italy) has an api to interface with for emergency alarms seemed to us a bit unrealistic, known the current status of health systems.
* Data4Help is not planned such that third parties can control in every moment the health status of subscribed users, but we mean that they can receive data only under request: these requests are assumed to be atomic in time.
* Third parties work in very different markets, use different software and have different grades of automatization of their processes

# Specific requirements

## External Interface Requirements

### User interfaces

To give a generic idea of what the application will look like we deployed the following mock ups:

[Data4Help]

…….

[AutomatedSos]

……

### Hardware Interfaces

[in case the sensor is physically part of the device where the application runs]

* The sensor which detects data (both for *Data4Help* and *AutomatedSos*): in our modelling it is supposed to be able to detect a huge number of parameters, but we can accept cases when low level sensors detect only a part of them, provided that the service is not put at risk.

### Software Interfaces

* Data4Help communicates with the user through the application we have suppose it has been created on purpose (cfr *Assumptions and Constrains).* Thanks to the facts that it is deployed by the same developers and little communication occurs, there shouldn’t be so many problems in adapting the code and no need for big software interfaces. In best solutions, *Data4Help* is a piece of code included in the application. No software interfaces are required for *AutomatedSos.*
* No software interfaces are required for interaction with third parties’ sofwares (*cfr Product functions, data Presentation*)

### Communication Interfaces

When the sensor is not included in the physical device where the application runs, the sensor is a wereable device (e.g. a bracelet) which sends data via Bluetooth technology: the app needs an appropriate communication interface to interact with the sensor.

## Functional Requirements

### [G1] Provide a form of unique identification (registration/login) of all the clients of the application

* [R1] If the client does not insert a identifier and password, the application does not let the user access any functionality[login]
* [R2] If the client declares that it has not an account (i.e. it’s the first access), he can fill a registration form to create a new one, providing an identifier and a password[registration];
* [R3] If the identifier provided in the registration form is already associated to a password, the application refuses the registration [unique identification]
* [D1] The client has correctly downloaded the application [i.e. he is not using a crack version]

#### [G1.1] Provide a form of unique identification (registration/login) of all users using the application

* [R4] If the user does not fill the registration form with his fiscal code and all other personal data, the application refuses the registration

#### [G1.2] Provide a form of unique identification (registration/login) of all third parties using the application

* [R5] If the third party does not fill the registration form with his official e-mail and all other public data, the system refuses the registration

### [G2] Allow the user to avoid being associated to his data without his permission

* [R6] If a third part asks for data of a single user, the system asks for the user’s permission
* [R7] The system refuses the request when the user denies access
* [R8] if a third part asks for data that involves less than 1000 people, the application refuses
* [R9] if a third part asks for data that involves more than 1000 people, the application anonymizes data before sending

### [G3] Allow third parties to request access to data of some specific individuals

* [D2] There is internet connection when the request is submitted
* [D3] The third party knows the fiscal code of the specific individual
* [R10] The system retrieves and shows data regarding an individual
* [R6] If a third part asks for data of a single user, the system asks for the user’s permission

### [G4] Allow third parties to request access to anonymized data of groups of individuals

* [D4] There is internet connection when the request is submitted
* [R9] if a third part asks for data that involves more than 1000 people, the system anonymizes data before sending
* [D5] Third parties express realistic constrains
* [R11] The system allows third parties to specify constraints to filter data
* [R12] The system provides data grouped and selected according to the requests

### [G5] Allow third parties to subscribe to new data

* [R13] the system notifies third parties when un update is available
* [D5] Third parties express realistic constrains
* [R14] Third parties can specify whether they are interested in updates
  + [R14.1] Third parties can specify constraints on data which are not yet available (i.e. they haven’t been elaborated yet or they regard a future period of time)
  + [R14.2] Third parties can ask for periodic data

### [G6] Allow third parties to obtain the most adapt data for their needs

* [R15] The system allows third parties to obtain aggregated data and statistics (average, maximum, minimum…)
* [R16] The system offers different options to visualize data
* [R17] Third parties can download data
* [D5] Third parties express feasible constrains

### [G7] Third parties are alerted, whenever a user is in danger of life, the application is working properly and there is internet connection

* [R18] If input data show a severe disease or the user communicates that an emergency is occurring, the machine contacts all third parties (with a notification containing the position and the health values)
* [D6] GPS always works properly indicating the user’s position
* [D7] For every location, there is at least a third party which is able to handle the emergency
* [R19] The application knows the correct thresholds for each type of user
* [D8] The user sets on “active” the application every time he feels he could need medical assistance

### [G8] If the user’s health status is not clear due to malfunctions, an Emergency number is alerted within an hour

* [R20] If the application does not read properly input data every 500 Ms (including absurd data), asks for confirmation of good health status.
* [R21] If the user does not respond to confirmation within 5 minutes, the system sends a message to the emergency number, provided through the registration form
* [R22] If the application does not send data for back up purpose every hour, the system sends a message to the emergency number
* [D9] The emergency number is correct
* [R4] If the user does not fill the registration form with his fiscal code and all other personal data, the application refuses the registration

### [G9] The user can temporarily suspend *AutomatedSos* in special cases

* [R23] The system allows to specify a “non-active” state, where nobody is alerted in any case

## Non-functional Requirements

Accuracy

This is the non-functional requirement we consider the most relevant. While assuming (here and in all other parts of RASD, cfr, *domain assumptions*) that GPS works properly, our biggest concern is about the sensor which detects values: there is a concrete risk of frequent false positives, caused by some little offsets. Such cases are particularly annoying because we can’t ignore the alarms but at the same time, they make us waste time, money and resources, threatening the correct working of our machine. We pointed out that there is a trade-off between this issue (which would request to wait for a lot of consecutive signals of emergency to be sure) and the issue of performance (which would request to react immediately). We decided to wait for 1,5 sec before alerting, which means, values below thresholds for three times in a row. However, provided that false positives are defined in statistics as *type one errors,* we state that the sensor must have prob (type I err) < 5%. -> deve diventare un domain assuption

Performance

[The following non-functional requirement regards only *AutomatedSos*]

Of course, we need a fast reaction to emergencies. Concretely, we state that the machine must guarantee a reaction time of less than 5 seconds from the time the parameters are below the thresholds. This means that the application should process, compares data and send a message to the server in less than 3,5 sec (provided that only three consecutive data below the threshold are a clear signal of severe disease and data are sent every 500 Ms).

In addition to this, 4G connection is required to ensure immediate communication within the system.

### Availability and reliability

[The following non-functional requirement regards only *AutomatedSos*]

* Concerning the server: We need a server available 24/7 to handle emergency messages as fast as possible because, among other reasons, there is no way for the app the handle an emergency without the server;
* Concerning the app: we also need availability for the app, but not (with so much relevance) reliability , due to the fact that it takes time to detect a disease ( the sensor does not send data in real time ecc..), in which there could be an app break down with no significative consequences for the service ( provided that recovery time is under 500 ms). Moreover, sometimes the app must stop working because its state is “non-active”

Security

Thinking of a market such as the American one, where health care is subject to negotiation between user s and companies, security of all sensitive information which could advantage malicious companies, is a very important concern for our application. Due to this reason, data encryption should be implemented in communications between the app and the server and the web apps.

### Maintainability

The system, because it could be considered a medical equipment (especially when talking about AutomatedSos) is intended to last several years to collect a significative number of data: to reach this goal, easy mantainability is required.

### Rubustness/Exception handling

Since the crucial importance of its goal, the system should be able to deal in a cautious way with all possible exceptions and problems.

## Design Constraints

### Hardware limitations

[ the user]:

* iOS or Android smartphone with 4G connection and Bluetooth connection
* GPS connection
* Wearable sensor with Bluetooth connection

In alternative

* iOS or Android smartwatch (which includes a sensor)
* 4G connection
* GPS connection

For visualizing data [Third parties],

* Modern browser able to render graphs and statistical models

### Any other constraint

Machine constrain: the applications should send data for back up purpose to the system every hour (cfr. *AutomatedSos: fast emergency handling and conservative approach*)

# Scenarios

## Scenario 1- Case of emergency

Gianni is a 76 years-old man and lives alone quite far from his daughter, Livia.

He suffers from hearts problems, so Livia decides to enrol him to “AutomatedSos”. Then she downloads the app on her father’s smartphone and buys him a smart bracelet to connect to the system.

After downloading the application helps him fill out the registration form to Data4Help, providing the Gianni’s fiscal code and general information. She also adds in medical information his problems of heart. Eventually she indicates her number as the “emergency number” required by the AutomatedSos service and connecting the device.

There is no problem connecting to the internet because in the home of Gianni there has been a Wi-Fi network for some years. Gianni also recharges the device every afternoon during the visits of his daughter so that battery is fully charged when he is alone at home. During this visit he sets the system *“non-active”.* When she goes away he clicks on the button “on” to reset the system *“active”.*

A day Gianni has a heart attack while alone in the house, and then his heart values fall sharply below the thresholds laid down for him by the application.

The system immediately sends an alarm to the companies offering the assistance service, together with the values of the heart rate and the position of Gianni’s home.

The Policlinic Hospital responds first to the warning transmitted and takes charge of the emergency, sending an ack. After receiving the confirmation, the system marks the State of emergency as *“handling”.*

## Scenario 2-Application breakdown

Derek is a man in his fifties and for some years has installed on his smartwatch an app to keep an eye on his health. The application is the one which supports Data4Help and Derek has to register to the service. He fills out the registration form with his fiscal code and his general infos.

One day he receives a notification about an additional service offered by TrackMe, *AutomatedSoS,* which guarantees automated assistance, then Derek, feeling intrigued, decides to add *AutomatedSos* on his smartwatch.

He has already an account, so he has just to add a phone number to contact in case of emergency: he chooses his wife’s number.

During the week-end Derek descends to the cellar to help his son to repair his bike forgetting to set the app status manually "*non-active*": unfortunately, the cellar, being underground, is not covered by their home Wi-Fi network.

The application then fails to send the data every hour and the server, not getting the back-up, sends a message to the emergency number to notify that the application is not working properly.

## Scenario 3-. Sensor breakdown

Anna is an elderly lady who has recently retired. Instead of retiring too, her husband works all day outside home so convinces Anna to register to AutomatedSos to be safer when he is not at home.

She buys a small smartwatch on which she installs the application. The first time she accesses the app, she is asked to register to Data4Help, so she provides her data and indicates her husband’s number as number of emergencies.

She wears the smartwatch every day when her husband is out and recharge it when he is at home. Before recharging the device, she sets it “*non-active*”: she opens the app on the smartwatch and pushes the button “off”.

One day she forgets to take it off before getting into the bath: a bit of water enters the smartwatch, causing a sensor breakdown. The sensor is no more able to send data correctly.

The application, not receiving data for more than 1 minute, sends a notification to Anna in order to know her health status. A notification appears on Anna’s smartwatch display, asking if she is okay or not. She presses on the “yes” button to confirms that she is okay.

Anna sees the message and confirms that she is okay but decides to manually disable the application, putting its status “non-active”, so that she can bring the smartwatch for repairing, without alerting the number of emergencies.

## Scenario 4-Data anonymization and data presentation

The Saint Francis Medical Clinic would like to open a new geriatric ward then turns to external consultants to figure out if it is convenient, or to understand how many people they might have.

Therefore, the consultants decide to register to Data4Help to collect some prediction data concerning the population living near the clinic.

They download the web application on their laptop and provide the company’s mail to create an account, then they can start to gather information.

First, they click on the button “advanced request” in order to obtain aggregated (manipulated) data. Then they set the filters for the request and ask the average pression and heart rates of people between 70 and 90 years living within a radius of 20 km from the clinic. The application has the data of more than 1500 people, and then accepts the request by providing the media required by the company.

To make things more understandable, the application shows a histogram indicating the number of people in every pression band, to distinguish correctly the big number of people with normal pression values from the little group of people with values very far from the average.

The company requires then the number of people who have heart problems in their medical history but who live within a radius of 8 km from the clinic. Again, they choose “simple request” and then fill the standard fields for the research, but this time being that only a small number of users meets these criteria, the request is rejected by the system. Then appears on the screen a warning that the data is not available because it is no possible to make them anonymous.

## Scenario 5-subscribe to new data

The municipal administration of Novate Milanese had a very positive impression about Data4Help before the last elections, finding some data very useful and interesting to get an idea on health of citizens.

In particular, they used the data provided by the application to propose some prevention programs or help for some diseases.

This year the Education Commissioner wants to allocate funds for a smoking-prevention program in all the schools of the municipality, because he is afraid that more and more kids start smoking during high school.

He decides to exploit Data4Help. He opens the app on his laptop and sign in providing the municipal official mail. From the main menu he moves the section for subscribing new data, selecting the item from the main menu.

Here he subscribes to new data on blood saturation of teenagers between 14 the 18 years in the next year.

After 1 year, when the deadline comes, the system notifies that the data he aimed at are now available. He receives an email at the municipal address. Then he accesses to the web app and sees a new notification message. After he clicks on, the system shows the new data collected, which confirm his fear.

## Scenario 6-Personal Use

Betty has just begun a new fitness program at “TonicPeople” gym, after being stopped for a few years. At the first lesson, the fitness coaches recommend using an app that monitors her health status to check whether the program is too stressful for her health. They suggest the app offered by TrackMe.

Betty downloads the app on her smartwatch and so she must register to Data4Help for using the app.

Betty enrols the service providing her social fiscal code, basic information about herself (weight, height, gender) and her health (pre-existing conditions, chronical diseases, pathologies…).

The “TonicPeople” gym has been using Data4Help for some years for monitoring its users, thanks to the fact that they provide their fiscal code when they sign up to the gym.

The Betty’s coach wants to check out her progresses and so he accesses to Data4Help from his laptop, using the gym’s email address. He gets on the search bar and type in the Betty’s fiscal code.

Betty immediately receives the request on her smartwatch, recognizes the company and clicks on the accept button to authorize using her data.

Data4Help finds and sends all kind of information it has from Betty’s profile stored in the database to the gym account.

The coach receives data and wants to show them to Betty, then he clicks on the download button and saves them.

# UML modelling

# Formal analysis using alloy

# open util/integer

# open util/boolean

# //signatures

# abstract sig Client{

# automatedSos: one Bool

# }

# sig User extends Client{

# fiscalCode: one String,

# status: one UserStatus,

# thirdpartiesallowed: String -> Data

# }

# sig UserStatus{

# active: Bool

# }

# sig ThirdParty extends Client{

# email: one String,

# alerts: set Int,

# datareceived: set Data,

# datapool: set Datapool

# }{

# #alerts > -1

# #datareceived > -1

# }

# sig Alert{

# data: one Data,

# status: one EmergencyStatus,

# alertID: one Int

# }{

# alertID > 0

# }

# sig EmergencyStatus{

# handling: Bool

# }

# abstract sig Request{

# requestID: Int,

# requester:String,

# time: Time,

# parameters: Parameters

# }

# {

# requestID > 0

# }

# sig IndividualRequest extends Request{

# identifier: one String

# }

# sig GroupRequest extends Request{

# //qua bisognerà per forza aggiungere qualcosa

# }

# sig Data{

# identifier: one String,

# parameters: one Parameters,

# healthValues: one HealthValue,

# location: one Location,

# time: one Time

# }

# sig HealthValue{

# userIdentifier: one String,

# heartRate: one Int,

# pression: one Int,

# bloodSaturation: one Int,

# bodyTemperature: one Int,

# }

# sig Location{

# latitude: one Int,

# longitude: one Int

# }

# sig Parameters{

# atomicparameter: some AtomicParameter

# }

# sig AtomicParameter{}

# sig Time{

# day: one Int,

# month: one String,

# year: one Int,

# hour: one Int,

# minute: one Int,

# second: one Int

# }{

# day > 0 and day < 32

# hour > -1 and hour < 25

# minute > -1 and minute < 61

# second > -1 and second < 61

# }

# sig Datapool{

# data: set Data

# }

# //facts

# //requests must regard subscribed users

# fact IndivdualRequestmustRegardAsubscribedUser{

# all r1: IndividualRequest | IsSubscribedtoData4Help[r1.identifier]

# }

# //data must regard subscribed users

# fact DataMustRegardSubscribedUser{

# all d1: Data | IsSubscribedtoData4Help[d1.identifier]

# }

# //There can't be alerts with the same id

# fact AlertsIDAreunique{

# no disjoint al1, al2 : Alert | al1.alertID = al2.alertID

# }

# //there can't be requests with the same id

# fact RequestsAreUnique{

# no disjoint req1, req2 : Request | req1.requestID = req2.requestID

# }

# //requests must have requesterString corresponding to existing Third parties

# fact CorrectRequestsID{

# all r1: Request | one t1: ThirdParty | r1.requester = t1.email

# }

# //third parties must have ids corresponding to unique alerts

# fact CorrectAlertsID{

# all t1: ThirdParty | all number: t1.alerts | one al: Alert | number = al.alertID

# }

# //alerts must regard subscribed users [To AutomatedSos]

# fact EmergenciesmustRegardASubscribedUSer{

# all em1: Alert | IsSubscribedtoAutomatedSos[em1.data.identifier]

# }

# //a third party can handle alerts only if he is subscribed to AUtomatedSOS

# fact HandleOnlyIFSubscribed{

# all t1: ThirdParty | (some number: Int | number in t1.alerts) implies isTrue[t1.automatedSos]

# }

# //there are alerts only if the user is active

# fact IfNonActiveThereareNoEmergencies{

# all al:Alert | IsActive[al.data.identifier]

# }

# //there can't be contemporary emergencies for the same user

# fact NocontemporaryEmergenciesForTheSameUSer{

# all disj al1, al2: Alert | al1.data.identifier = al2.data.identifier implies al1.data.time != al2.data.time

# }

# //there are no users with the same fiscal code

# fact FiscalCodeeIsUnique{

# no disjoint u1,u2: User | u1.fiscalCode = u2.fiscalCode

# }

# //there are no third parties with the same email

# fact EmailsAreUnique{

# no disjoint t1,t2: ThirdParty | t1.email = t2.email

# }

# //there are no two health values regarding the same user of the same dateTime

# fact NoSameTimeSameUserHealthValues{

# all disj d1,d2: Data | d1.identifier = d2.identifier implies d1.time != d2.time

# }

# //if an alert is marked as "handling" there is exactly one third party which is handling it

# fact ExactlyOneHandling{

# all al1: Alert | isTrue[al1.status.handling] implies one t1: ThirdParty | al1.alertID in t1.alerts

# }

# //request are atomic: just one request per time by the same third party to the same user

# fact OnePerTime{

# //no disjoint r1,r2: IndividualRequest | r1.requester = r2.requester

# //come la modello sta cosa????

# }

# // useful preds

# pred IsSubscribedtoData4Help[u:String]{

# one u1:User | u = u1.fiscalCode

# }

# //this implies the previous

# pred IsSubscribedtoAutomatedSos[u:String]{

# one u1:User | u = u1.fiscalCode and isTrue[u1.automatedSos]

# }

# //if a user is active or inactive

# pred IsActive[u:String]{

# one u1:User | u = u1.fiscalCode and isTrue[u1.status.active]

# }

# //Data4Help

# //accessingtoindividualData

# pred AccessToIndividualData[u1, u1': User, req: IndividualRequest, t1, t1': ThirdParty, d1: Data] {

# //preconditions

# d1 not in t1.datareceived

# req.requester = t1.email

# req.identifier = u1.fiscalCode

# d1.identifier = u1.fiscalCode

# t1.email -> d1 not in u1.thirdpartiesallowed

# d1.parameters = req.parameters

# //postconditions

# d1 in t1'.datareceived

# t1'.email -> d1 in u1'.thirdpartiesallowed

# }

# //refusingaccesstoindividualData

# pred RefuseAccessToindividualData[u1, u1': User, req: IndividualRequest, t1, t1': ThirdParty, d1: Data] {

# //preconditions

# d1 not in t1.datareceived

# req.requester = t1.email

# req.identifier = u1.fiscalCode

# d1.identifier = u1.fiscalCode

# t1.email -> d1 not in u1.thirdpartiesallowed

# d1.parameters = req.parameters

# //postconditions

# d1 not in t1'.datareceived

# t1'.email -> d1 not in u1'.thirdpartiesallowed

# }

# //cannotanonymizeData

# pred noaccesstoAnonymizedData[t1,t1': ThirdParty, req: GroupRequest, datapool: Datapool]{

# //preconditions

# # datapool.data.identifier < 3

# req.requester = t1.email

# //in datapool there are all and only data

# all d1:datapool.data | d1.parameters = req.parameters

# no d1:Data | d1.parameters= req.parameters and d1 not in datapool.data

# //all data are not known by the third party (assumption for simplicity)

# all d1:datapool.data | d1 not in t1.datareceived

# //postcondition

# all d1:datapool.data | d1 not in t1'.datareceived

# }

# //cananonymizedata

# pred accesstoAnonymizedData[t1,t1': ThirdParty, req: GroupRequest, datapool: Datapool]{

# //preconditions

# # datapool.data.identifier > 3

# req.requester = t1.email

# //in datapool there are all and only data

# all d1:datapool.data | d1.parameters = req.parameters

# no d1:Data | d1.parameters= req.parameters and d1 not in datapool.data

# //all data are not known by the third party (assumption for simplicity)

# all d1:datapool.data | d1 not in t1.datareceived

# //postcondition

# all d1:datapool.data | d1 in t1'.datareceived

# }

# //AutomatedSos

# //handling emergencies

# //commands

# run noaccesstoAnonymizedData for 10 but exactly 10 String, 8 Int

# run AccessToIndividualData for 5 but exactly 8 String, 8 Int

# run RefuseAccessToindividualData for 5 but exactly 8 String, 8 Int

# //data4Help

# //

# //i gruppi di data devono comprendere più di mille

# //una alert deve essere gestitaReferences