Politecnico di Milano

Master's Degree in
Computer Science and Engineering

Software Engineering 2

---

# TrackMe
# Requirements Analysis and Specification Document

---

*Authors*
Alberto Archetti
Fabio Carminati

*Reference professor*
Elisabetta Di Nitto

v.0 - October 28, 2018

# Contents


**1 Introduction** **1**
- 1.1 Purpose . . . 1
  - 1.1.1 Project description . . . 1
  - 1.1.2 Goals . . . 1
- 1.2 Scope . . . 2
  - 1.2.1 World . . . 2
  - 1.2.2 Shared phenomena . . . 2
- 1.3 Definitions, acronyms, abbreviations . . . 3
  - 1.3.1 Definitions . . . 3
  - 1.3.2 Acronyms . . . 3
  - 1.3.3 Abbreviations . . . 3
- 1.4 Revision history . . . 3
- 1.5 Reference documents . . . 3
- 1.6 Document structure . . . 3

**2 Overall description** **5**
- 2.1 Product perspective . . . 5
- 2.2 Product functions . . . 5
  - 2.2.1 Profile management . . . 6
  - 2.2.2 Data gathering . . . 6
  - 2.2.3 Data sharing . . . 6
  - 2.2.4 Data management . . . 7
  - 2.2.5 Payment handling . . . 7
  - 2.2.6 SOS handling . . . 8
- 2.3 User characteristics . . . 8
- 2.4 Assumptions, dependencies, constraints . . . 9
  - 2.4.1 Domain assumptions . . . 9
  - 2.4.2 Dependencies . . . 9
  - 2.4.3 Constraints . . . 10

**3 Specific requirements** **11**
- 3.1 Scenarios: . . . 11
- 3.2 External interface requirements . . . 11
  - 3.2.1 User interfaces . . . 11
  - 3.2.2 Hardware interfaces . . . 11
  - 3.2.3 Software interfaces . . . 11

3.2.4 Communication interfaces . . . . . 11
3.3 Functional requirements . . . . . 11
3.4 Performance requirements . . . . . 11
3.5 Design constraints . . . . . 11
3.5.1 Standards compliance . . . . . 11
3.5.2 Hardware limitations . . . . . 11
3.5.3 Any other constraint . . . . . 11
3.6 Software system attributes . . . . . 11
3.6.1 Reliability . . . . . 11
3.6.2 Availability . . . . . 11
3.6.3 Security . . . . . 11
3.6.4 Mantainability . . . . . 11
3.6.5 Portability . . . . . 11

**4 Formal analysis using Alloy** **12**

**5 Effort spent** **13**

# 1 Introduction

## 1.1 Purpose

### 1.1.1 Project description

TrackMe wants to develop a software-based service that allows individual users to collect health data, called `Data4Help`. This data can be retrived from the system and visualized according to different filters by a user interface.
The system allows third parties registration. Third parties can request access to users'collected data in two ways:

**Single-person data** After the the third party makes a request to the system for a single user data sharing (providing for example the fiscal code of the user), the system asks the user for authorization; if positively provided, the third party is granted access to the user's data

**Amonymous-group data** Thirs parties can be interested in big amounts of data, but not in who are the people that are providing it; the system, once the request by the third party is sent, checks if the data can be effectively anonymized (it must find at least 1000 people that match the third party request) and, if positively evaluated, grants access to the anonymized data to the third party that requested it

Third parties can subscribe to new data and receive it as soon as it is collected by the system.

Another service that TrackMe wants to develop is `AutomatedSOS`, built on `Data4Help`. This service analyzes users'data and calls a SOS whenever data exceedes the basic health parameters. For this particular purpose, system performances will be a critical aspect to be taken into account, because even the slightest delay matters in critical health situations.

### 1.1.2 Goals

Here we present the goals that will be reached once the project is completed:

`G.U1` Users can collect, store and manage their health data

`G.U2` Users can choose to have their health monitored; if their health is critical, an ambulance will be dispatched

`G.T1` Third parties can ask single users for their health data sharing

`G.T2` Third parties can request access to anonymized data that comes from groups of people

`G.T3` Third parties can subscribe to new data and receive it as soon it is produced

## 1.2 Scope

### 1.2.1 World

Our *world* is composed of two main types of actors: users and third parties. Users are interestes in monitoring their health parameters and third parties are interested in developing services or researches that exploit the data gathered by the users. `Data4Help` is the service that acts as a bridge between these actors'needs.
Phenomena that occur in the *world* and are related to our application domain are

- physical conditions of our users
- third parties projects
- SOS system that is able to dispatch ambulances

These phenomena exist in the *world*, but cannot be observerd directly by our system.

### 1.2.2 Shared phenomena

In order to communicate with the *world*, our system needs to share some aspects with it. We will list the aspects controlled by the world, but observable by the machine:

`S.1` physical parameters of the users, gathered through sensors on wereable devices

`S.2` third parties requests to the system for the data they need

`S.3` users'location, acquired through GPS signals

On the other hand, the aspects that occur in the machine, but are observable by the world are

`S.4` interfaces that organize the gathered data that can be filtered according to time or type of data

`S.5` messages for the SOS system, that are sent in case of critical health of a user

## 1.3 Definitions, acronyms, abbreviations

### 1.3.1 Definitions

### 1.3.2 Acronyms

### 1.3.3 Abbreviations

## 1.4 Revision history

| Version | Log |
|---|---|
| `v.0` | First version of RASD completed |

## 1.5 Reference documents

See References for details on the consulted documents.

## 1.6 Document structure

This document uses the IEEE standards for requirement analysis documents as a guideline towards a clear and logical explanation of its contents:

- Section 1 gives a brief introduction on the project to be developed, by describing its goals and the environment in which it will be released; keeps track of the revision history
- Section 2 describes the world and the shared phenomena, by defining assumptions and constraints; it identifies the main functions of the project
- Section 3, as the main part of this document, is about requirement analysis; it has also sections about interfaces of the system and software attributes
- Section 4 contains the `Alloy` model that certifies correctness of goals implication by requirements and domain assumptions
- Section 5 lists the work sessions that drove this project's development, ordered by date, as the hour counter of effort spent by each group member

# 2 Overall description

## 2.1 Product perspective

`Data4Help` is a service oriented to data acquisition and data sharing. Its software nature rises the necessity to combine it with another service able to directly retrive raw data from the *world*. Today we can find for sale multiple wearables that can acquire data from users and make it readable from software side. `Data4Help` users should already own these devices in order to exploit the data acquisition functionality (Section 1.2.2: **`S.1`**). Once the user registered to the service, its interface will gather the last data collected and organize it in a chart view, that can be filtered by data or type and rendered by the user interface (Section 1.2.2: **`S.4`**; Section 3.2.1).

It is mandatory for the user's wearable to provide a GPS signal, if the user wants to apply to the `AutomatedSOS` service. GPS will be used to track the user's location in case of health danger and the signal will be shared to the SOS service that already exists in the *world* (Section 1.2.2: **`S.3`**, **`S.5`**). This SOS service accepts messages that contain the GPS location of the person in health danger and an emergency log that `AutomatedSOS` generates from the acquired data. The log explains the suspected health danger and the data that passed the defined thresholds.

Third party organization are interested in data gathering. In order to allow them to make requests for specific types of data, `Data4Help` provides a user interface that is in charge of composing their request, to make it understendable by the software (Section 1.2.2: **`S.2`**; Section 3.2.1). The interface provides all the possible options that the third party can compose in order to provide the closest data request to its needs.

## 2.2 Product functions

Here we present the major functions that our product will offer. Some of them will entirely be handled internally by our system, but for others it will rely on external services. In the latter case, we will specify that the system will not directly provide the service and we will add example systems that already exist in the *world*, in order to guarantee the feasibility of the functions.

### 2.2.1 Profile management

The system will provide a registration form at which users and third parties can apply. Once registered, they will have a uniquely identifiable account, provided the requested information for its creation (see Table 1). Once the account has been successfully created, its owner can exploit the system's functionalities.

Note that accounts for users and third parties must be distinguishable from the system perspective, as it should offer different functionalities to different account types, in order to reflect the account owners'needs.

| **Account type** | **Required information** | **Optional information** |
|---|---|---|
| User | *email, password, fiscal code, date of birth, weight, height* | *social status, address, hours of work per day, hours of sport per week* |
| Third party | *email, password, third party name, third party description* | *website, research interests* |

Table 1: Example of login form information

### 2.2.2 Data gathering

Data gathering is exploited through physical wearables that users wear and that can communicate with our software by API calls (ex. Google Fit API [6]). Data entries are identified with a timestamp and the owner user.

Collected data types depend on wearables. For example, the most common sensors for health parameters that we find on smart clothing [5] are pulse, body temperature, electrocardiogram, myocardial and blood oxygen.

Data type handling should be implemented in a flexible and extensible way in the system, because wearables technologies change rapidly and our software should adapt in the least invasive way possible.

### 2.2.3 Data sharing

`Data4Help` relies on an internal database for data storage. Once data has just been added to the system, only the user that produced it will have access. Data sharing is exploited by granting access to the stored data to third parties, if their requests have been accepted.

Third party requests are encoded in the system by filling a form that contains information like Table 2. They can be accepted or rejected. In the former case the third party is granted access to the data, while in the latter it is not granted access.

| **Request type** | **Accept condition** | **Filters** |
|---|---|---|
| Single-user | Target user should accept the request through his/her account | *fiscal code of target user, from-date, to-date, data types (weight, heart rate, etc.), time granularity (seconds, minutes, hours)* |
| Anonymous-group | Every user should have accepted the automatic sharing of requested data | *size, from-date, to-date, data types (weight, heart rate, etc.), user characteristics (age interval, weight interval, etc.)* |

Table 2: Example of third party request form

### 2.2.4 Data management

Once a user collected some data, he/she can organize it by changing the view options in the user interface. These options depend on the device the user is working on, but will always provide basic filters such as time interval or data type. Once filters have been selected by the user, the graphical interface will render a chart that organizes collected data according to them (see Section 3.2.1).

### 2.2.5 Payment handling

Payment will entirely be handled from an external software system that should

- initialize payment processes
- check if the payment is feasible and, if not, notify to our system
- handle the payment operation

- notify to our system if the operation has concluded correctly, otherwise notify the error occurred

There are plenty of payment handlers that exist in our *world* and can be paired to our system (ex. PayPal API [7]).

#### 2.2.6 SOS handling

This function is heavily dependant on the country in which the SOS will be handled. Our system will only communicate to the emergency service that already exists in the *world* through automatic API calls (ex. RapidSOS Emergency system for US [8]).

Calls to SOS are handled by `AutomatedSOS` that signals users GPS position and health status feedback. Calls occur, if the user previously subscribed to `AutomatedSOS`, when his/her health parameters are below certain thresholds. The time between health danger detection and emergency call must be less than 5 seconds.

### 2.3 User characteristics

The following list contains the actors that are involved in the system functions:

**User** Person interested in the `Data4Help` **data management** service; he/she is required to register to the `Data4Help` service in order to exploit its functionalities; he/she can also apply to the `AutomatedSOS` service; every user owns an appropriate device for `Data4Help` acquisition service that can monitor at least heartbeat and GPS location
ex. *an athlet that wants to monitor his/her physical status during sport activity, an old person that suffers from heart diseases or a sedentary worker that wants to keep track of his health parameters in the working hours*

**Third Party** Entity interested in the `Data4Help` **data sharing** service; it is required to register to the `Data4Help` service in order to exploit its functionalities; according to the scope of its requests, it may be interested on the data of a particular user or in aggregated chunks of data
ex. *the physician of a person that suffers from heart diseases or a statistical institute*

## 2.4 Assumptions, dependencies, constraints

### 2.4.1 Domain assumptions

#### World

`D.W1` Signals processed by wearables are encoded correctly and represent the status of the *world*

`D.W2` Given certain health parameters[1], it is possible to decide if a person is in health danger just by checking wether the parameters are above or below certain thresholds

#### Existing systems

`D.E1` In the *world* already exists a SOS system that is able to dispatch ambulances and accepts emergency calls through an API

`D.E2` In the *world* already exists a payment handler that is able to deliver money payments and accepts calls through an API

`D.E3` In the *world* already exist wearables that encode signals for health status; these encoded signals are accessible from the software side

#### Legal constraints

`D.L1` Acquired data can be sold to third parties

### 2.4.2 Dependencies

`Data4Help` relies on

- **payment handler**, to deal with payment of third parties'requests
- **wearables**, to encode users'data

`AutomatedSOS` relies on

- **wearables**, to encode GPS signals and users'parameters
- **SOS system**, for ambulance dispatching

[1] TBD

### 2.4.3 Constraints

`AutomatedSOS` time performances are critical: the emergency call must be delivered no more than 5 seconds after the health danger detection, in order to have an ambulance dispatched from the SOS system as soon as possible. It is important to note again that `AutomatedSOS` is in charge only of the emergency call.

`Data4Help` acquires data through external devices owned by other companies, so it must be legally authorized to sell the data it acquires. In this document we assume that there are no legal issues for the selling activity, as TrackMe will develop contracts with the wearables companies in order to solve this issue.

# 3 Specific requirements

## 3.1 Scenarios:

## 3.2 External interface requirements

### 3.2.1 User interfaces

### 3.2.2 Hardware interfaces

### 3.2.3 Software interfaces

### 3.2.4 Communication interfaces

## 3.3 Functional requirements

## 3.4 Performance requirements

## 3.5 Design constraints

### 3.5.1 Standards compliance

### 3.5.2 Hardware limitations

### 3.5.3 Any other constraint

## 3.6 Software system attributes

### 3.6.1 Reliability

### 3.6.2 Availability

### 3.6.3 Security

### 3.6.4 Mantainability

### 3.6.5 Portability

# 4 Formal analysis using Alloy

```
sig Name, Addr {}

sig Book {
        addr: Name -> lone Addr
}

pred show {}

run show for 3 but 1 Book
```

# 5 Effort spent

# References


[1] Mandatory Project Assignment AY 2018-2019

[2] IEEE 830-1993 - IEEE Recommended Practice for Software Requirements Specifications

[3] ISO/IEC/IEEE 29148 - Systems and software engineering — Life cycle processes — Requirements engineering

[4] Collection and Processing of Data from Wrist Wearable Devices in Heterogeneous and Multiple-User Scenarios
`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038811/`

[5] Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices
`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5334130/`

[6] Google Fit API
`https://developers.google.com/fit/overview`

[7] PayPal API
`https://developer.paypal.com/docs/`

[8] RapidSOS Emergency API
`https://info.rapidsos.com/blog/product-spotlight-rapidsos-emergency-api`

[9] Slides of the course by Prof. Di Nitto
`https://beep.metid.polimi.it/`

[10] LaTeXtemplates
`http://www.latextemplates.com/`

[11] Alloy Latex Highlighting
`https://github.com/Angtrim/alloy-latex-highlighting`