Francesco Pontiggia 921085 10524642
Arianna Ricci 921275 10536266

Trackme RASD

REQUIREMENTS ANALYSIS AND SPECIFICATION DOMAIN (RASD)

Release date: 11/11/2018

Version: 3.0

Table of content

1. Introduction

1.1. Purpose

1.1.1 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 (most important of all) and 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.

1.1.2. Purpose of the project

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 from 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 over 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.

1.2. Scope

1.2.1 Analysis of the world and the shared phenomena

First of all, with Data4Help TrackMe 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). In the first case, we don't care about how data are transferred from the sensor to the smartphone, since we consider the users' devices external to our boundaries.

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 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' privacy occurs when a third party is in control of a user's data without 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 or 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. Regarding this, we consider it external to our boundaries: the system sends the alert and receives the ack, but it can't affect in any way the actual 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 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.

1.2.2. 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) (although it seems of no relevance, it's the main cause of the choice of providing data through a website and avoiding interfacing directly with the ERP of the third parties)
- Unique identification of clients (i.e. associating every operation to a client's profile)

1.2.3. Shared phenomena

[Controlled by the world]

- Detection of good health values (above thresholds)
- Detection of bad health values (below/over 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
- 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 one has been put here and not in machine phenomena because, in our modelling, it can be in some ways detected by the server]

- Sensor breakdown (not correct detection of data)

[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

1.2.4. Machine phenomena

- Data queries
- Data insertings
- 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.

1.2.5. 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

1.2.6. 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

1.2.7. 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)

1.3. Definitions, acronyms, Abbreviations

1.3.1. Definitions

- "***Health status***": when in the following parts we state "health status" we are meaning the following values:
  - Heart rate: it's an indicator of heart diseases (to detect heart attacks)
  - Blood pressure: 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;
- ***"Thresholds"***: specific values for the health status that clearly indicate a disease (minimum and maximum values within which each parameter of the state of health must stand so that the user is not considered in danger of life)
- ***"Active/ non-active":*** condition in which the application receives/does not receive information in real time
- ***"User":*** person who interfaces to the application via a wearable device that provides his/her data to the system
- ***"Third Party":*** entity that interfaces to the web application in order to request data or offer assistance
- ***"Client":*** everyone using the service, both the person who wears the user and the third party
- ***"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.

1.3.2. Acronyms

- RASD: Requirements analysis and specification domain
- API: Application Programming Interface

1.3.3. Abbreviations

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

1.4. Revision history

This is the third release of the document. We have fixed some minor errors.

1.5. Document structure

The RASD is divided in 6 sections:

1. The first section gives a general idea of the purpose of the document and the project. The problem given is presented, explained and analysed, in terms of phenomena and goals, that are listed. Some basic information useful for reading the document is also provided.

2. The second section gives a more detailed description of the problem and its modelling. It describes the specific functions of the system and the characteristics of the users. Eventually, it lists all the assumptions and restrictions that have been taken into account to model the system.

3. In the third section, the requirements of the application are listed and deepened. First, the interface requirements are illustrated, then the functional requirements, associated with their goals, eventually the non-functional requirements are explained.
For the analysis of the functional requirements, scenarios have been defined, from which the most relevant use cases have been obtained. Both are reported in the section together with different UML diagrams, created to make some important requirements and concepts clearer.

4. The fourth section is entirely dedicated to the Alloy modelling and results

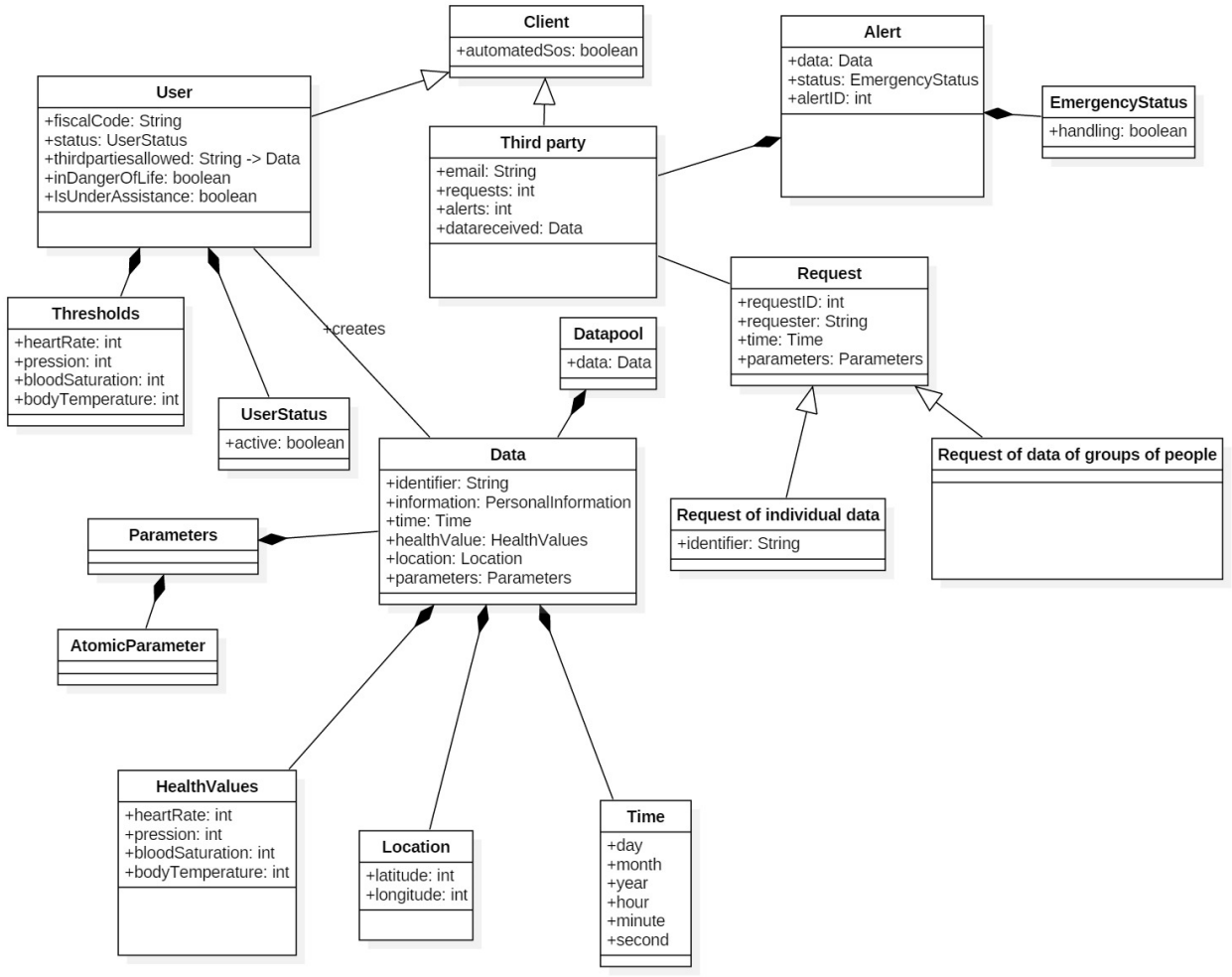
5. In the fifth section is reported the time we have spent in the writing of the RASD

6. In the sixth section we have inserted all the references to the tools used during the work

2. Overall description

2.1. Product perspective

2.1.1 Class Diagram



2.2. Product functions

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

2.2.1 Both Data4Help and AutomatedSos: Registration

After downloading the app, users must provide all their data and personal information (in Italy, we accept the fiscal code), through a registration form. In this way all the users can be uniquely 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.

2.2.2. 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 he accepts the system should be able to retrieve his specific data and send them without anonymizing.

2.2.3. 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 third parties should also have the possibility to express some preferences in data and to be informed by the system when there are produced: this means the system must provide several options.

2.2.4. 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.

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.

2.3 User characteristics

2.3.1. 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); it is also interested in offering assistance service to people.

2.4. Assumptions, dependencies and constraints

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

2.4.1. 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

2.4.2. 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 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 login *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

3. Specific requirements

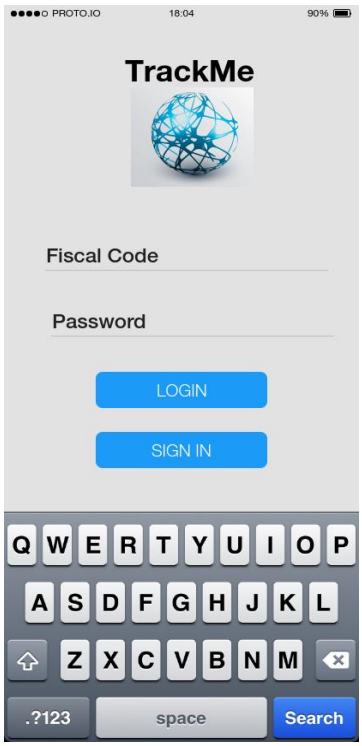
3.1 External Interface Requirements

3.1.1. User interfaces

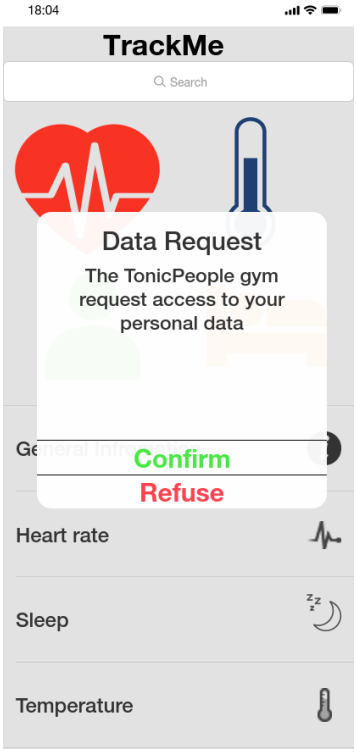
To give a generic idea of what the application will look like we deployed the following mock ups:

Data4Help

3.1.1.1. User Login to TrackMe's application



3.1.1.2. The application asks the User to access his/her data



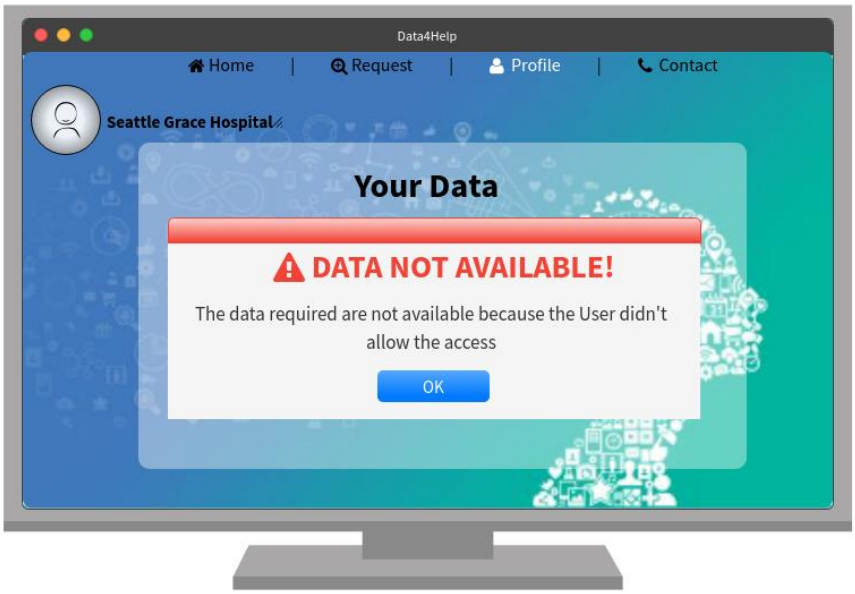
3.1.1.3. Third Party Login

3.1.1.3. Third Party home page

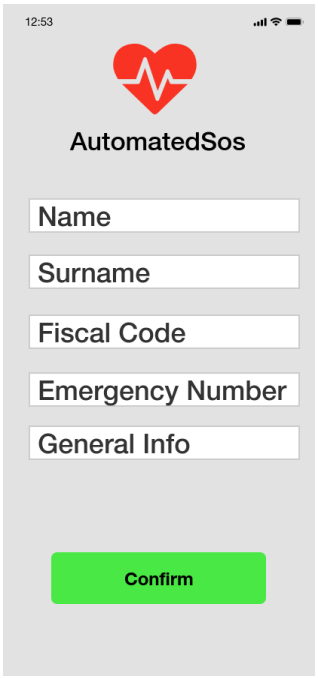
3.1.1.4. Third Party request of data

3.1.1.4. Third Party visualize and download the data

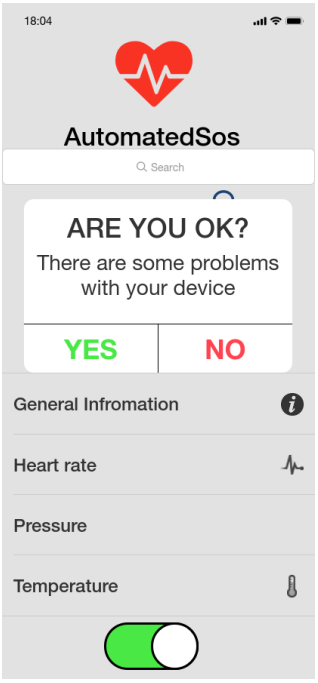
3.1.1.4. Third Party is notified that data are not available because the User didn't allow the access (same warning if data cannot be anonymized)



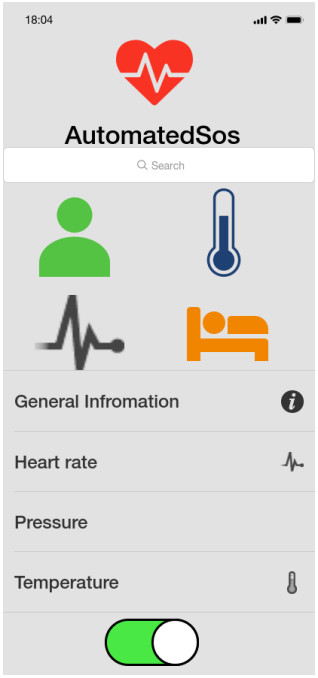
3.1.1.6. User registration



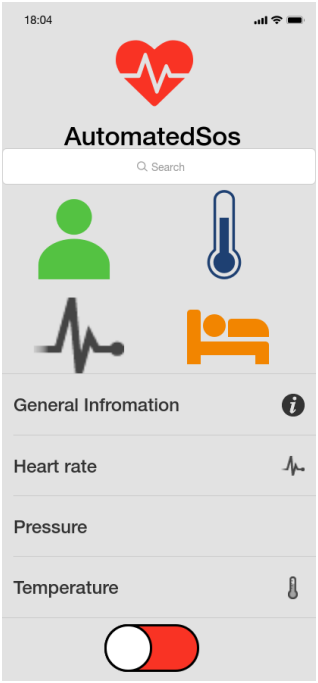
3.1.1.7. User notification of malfunction



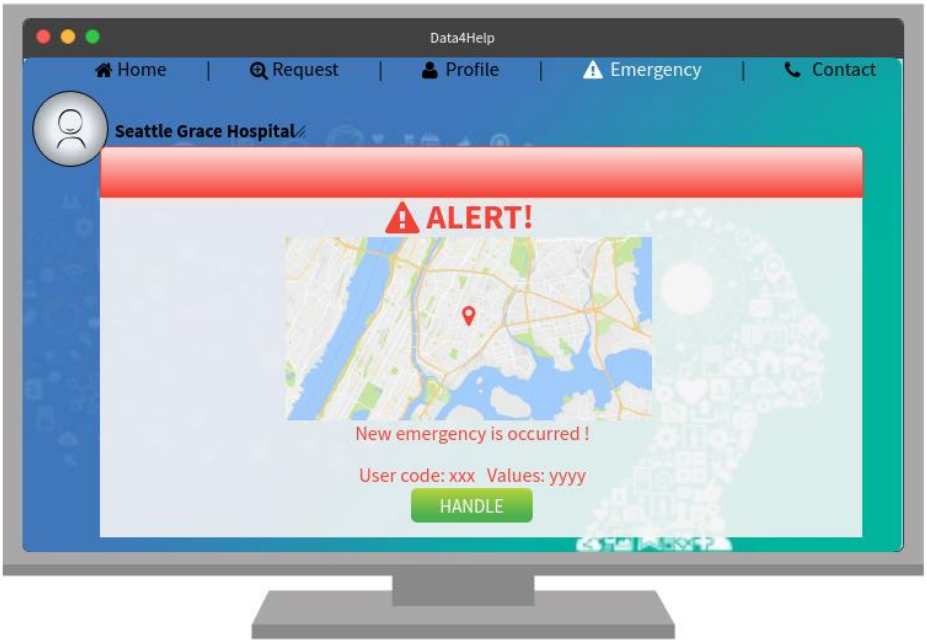
3.1.1.8. User Status On



3.1.1.9. User Status Off



3.1.1.10. Third Party alert notification of emergency



3.1.2. Hardware Interfaces

[in case the sensor is physically connected to 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.

3.1.2. 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*)

3.1.3. 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.

3.2 Functional Requirements

3.2.1. Functional requirements and goals

[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
- [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

- [D2] 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
- [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

- [R22] The system allows to specify a "non-active" state, where nobody is alerted in any case

3.2.2. Scenarios

3.2.2.1 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 alert's status changes in *"handling"*.

3.2.2.2 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.

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

Since they asked for statistical tools, the application shows a histogram indicating the number of people in every pressure band, to distinguish correctly the big number of people with normal pressure 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 fill the standard fields for the

research and then click on “request” button, 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.

3.2.2.4 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.

3.2.2.5 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.

3.2.3. Use cases descriptions

DATA4HELP

3.2.3.1. Sign up to Data4Help

Name	Sign up
Goals and Requirements	[G1] [G1.1] [R2] [R3] [R4]
Actor	User
Entry conditions	The User has installed the TrackMe's application on his/her device
Event flow	1. The User opens the app and clicks on "Sign in" button 2. The User fills the mandatory fields, providing his/her fiscal code, password and other necessary information 3. The User clicks on the "Confirm" button 4. The system receives and saves the data
Exit condition	The User is successfully registered to the application and he/she is able to use it on his/her device
Exceptions	1. The User provides inserts not valid data in one or more mandatory fields 2. The User's fiscal code is already associated with a password Both the exceptions are handled by notifying the User and taking him/her back to the point 1.

3.2.3.2 Sign up to Data4Help

Name	Sign up
Goals and Requirements	[G1] [G1.2] [R2] [R3] [R5]
Actor	Third Party
Entry conditions	The Third Party have installed Data4Help web application on his/her device
Event flow	1. The Third Party is on the homepage of the app and clicks on "Sign in" button 2. The Third Party fills the mandatory fields, providing his/her e-mail address, password and other necessary information

	3. The Third Party clicks on the “confirm” button 4. The system receives and saves the data
Exit condition	The Third Party is successfully registered to the application and he/she is able to use it on his/her laptop
Exceptions	1. The Third Party provides inserts not valid data in one or more mandatory fields 2. The Third Party’s e-mail address is already associated with a password Both the exceptions are handled by notifying the Third Party and taking him/her back to the point 1.

3.2.3.3 Login to Data4Help

Name	Login
Goals and Requirements	[G1] [G1.2] [R1]
Actor	Third Party
Entry conditions	The Third Party have installed Data4Help web application on his/her device and is already correctly signed up to Data4Help
Event flow	1. The Third Party opens the application on his/her laptop 2. The Third Party enters his/her credentials in “E-mail” and “Password” fields 3. The Third Party clicks on “Login” button 4. The Third Party is successfully logged, and he/she is redirected to the Data4Help’s homepage
Exit condition	The Third Party is successfully redirected to the Data4Help’s homepage
Exceptions	1. The Third Party enters not valid e-mail 2. The Third Party enters not valid password Both the exceptions are handled by notifying the User and taking him/her back to the point 2.

3.2.3.4. Login to Data4Help

Name	Login
Goals and Requirements	[G1] [G1.1] [R1]
Actor	User
Entry conditions	The User has installed the TrackMe's application on his/her device and is already correctly signed up to Data4Help
Event flow	1. The User opens the application on his/her device 2. The User enters his/her credentials in "Fiscal Code" and "Password" fields 3. The User clicks on "Login" button 4. The User is successfully logged, and he/she is redirected to the homepage of the TrackMe's application
Exit condition	The User is successfully redirected to the application homepage
Exceptions	1. The User enters not valid fiscal code 2. The User enters not valid password Both the exceptions are handled by notifying the User and taking him/her back to the point 2.

3.2.3.5. Obtain personal data of the User

Name	Obtain to personal data
Goals and Requirements	[G3] [R6] [R10]
Actor	User, Third Party
Entry conditions	1. The User is signed up and logged to the application 2. The Third Party is already signed up and logged to the application
Event flow	1. The Third Party enters the User's fiscal code on the research bar 2. The Third Party clicks on the "Request" button 3. The system receives the request 4. The system sends a message to the User with the information about the Third Party 5. The User receives the message and opens it 6. The User accepts the request and clicks on "ok" button 7. The system sends the data to the Third Party
Exit condition	The Third Party obtains personal data about a single User, with his/her permission
Exceptions	1. The User does not respond to the system's message This exception is handled by sending another message to the User for a maximum of 3 times. If the User never responds the system notifies the Third Party that the data is not available.

3.2.3.6. Obtain data of a group of people

Name	Obtain to data of group of people
Goals and Requirements	[G4] [R8] [R9]
Actor	Third Party
Entry conditions	1. The Third Party is already signed up and logged to the application

Event flow	1. The Third Party clicks on the “make a request” button of the main menu 2. The Third Party fills the fields about constrains and selects the button of the options 3. The Third Party clicks on the “Request” button 4. The system receives the request 5. The system anonymizes the data (removing the explicit references to the single User) 6. The system notifies the Third Party that the data are ready 7. The Third Party clicks on the “show” button 8. The Third Party sees the data on the screen
Exit condition	The Third Party obtains the data requested about a group of people
Exceptions	1. The data cannot be anonymized (involving less than 1000 people) The exception is handled sanding to the Third Party a warning communicating that data are not available because they cannot be anonymized (involving less than 1000 people)

3.2.3.7. Control the access to personal data

Name	Avoid access to personal data
Goals and Requirements	[G2] [R6] [R7]
Actor	User, Third Party
Entry conditions	1. The User is signed up and logged to the application 2. The Third Party is already signed up and logged to the application
Event flow	1. The Third Party enters the User’s fiscal code on the research bar 2. The Third Party clicks on the “Request” button 3. The system receives the request 4. The system sends a message to the User with the information about the Third Party 5. The User receives the message and opens it 6. The User accepts the request and clicks on “no” button 7. The system notifies the Third Party that User refuses the access

Exit condition	The User's personal data are hidden to the Third Party, according to his will
Exceptions	1. The User does not respond to the system's message This exception is handled sending another message to the User for a maximum of 3 times. If the User never responds the system notifies the Third Party that the data is not available.

3.2.3.8. Subscribe new data

Name	Subscribe new data
Goals and Requirements	[G5] [R13] [R14]
Actor	Third Party
Entry conditions	1. The Third Party is already signed up and logged to the application
Event flow	1. The Third Party clicks on "make a request" option of the main menu 2. The Third Party clicks on "subscribe new data" section 3. The Third Party fills the fields about constrains 4. The Third Party indicates some specials options for the data such as the time or the period during which collect the data 5. The Third Party clicks on the "Request" button 6. The system receives and saves request 7. When the data are ready the system sends a notification to the Third Party 8. The Third Party clicks on the notification and on "Show" button 9. The Third Party visualizes the data
Exit condition	The Third Party obtains new data after a certain period of time
Exceptions	1. The data collected by the system during the period of time requested cannot be anonymized

	The exception is handled sanding to the Third Party a warning communicating that data are not available because they cannot be anonymized (involving less than 1000 people)

3.2.3.9. Personalize requests

Name	Personalize requests
Goals and Requirements	[G3] [G6] [R11] [R15] [R16] [R17]
Actor	Third Party
Entry conditions	1. The Third Party is already signed up and logged to the application 2. The Third Party is making a request of data
Event flow	1. The Third Party expresses in which health's information it is interested, selecting by the menu 2. The Third Party sets position, age, gender and similar constrains, clicking on dedicated button 3. The Third Party chooses, if it wants to, from statistical tools 4. The Third Party clicks on the "Request" button 5. The system receives the request 6. The system manipulates the data 7. The system notifies the Third Party that the data are ready 8. The Third Party clicks on the "Show" button 9. The Third Party sees the data on its screen 10. The Third Party chooses, if it wants to, the more comfortable tools to view the data 11. The Third Party that wants to save the data, clicks on "Download" button 12. The system sends the data
Exit condition	The Third Party visualizes and obtains the data required
Exceptions	1. The Third Party gets wrong in clicking some buttons or in filling some fields This exception is handled notifying the error at the Third Party and bringing it back to the previous step

3.2.3.10. Sign up to AutomatedSos

Name	Sign up
Goals and Requirements	[G1] [R2] [R3] [R4]
Actor	The User
Entry conditions	1. The User has installed AutomatedSos his/her device
Event flow	1. The User opens the app and clicks on “Sign in” button 2. The User fills the mandatory fields, providing his/her fiscal code, password and other necessary information 3. The User provides a mandatory emergency number 4. The User clicks on the “confirm” button 5. The system receives data 6. The system elaborates data and calculates the threshold for the User 7. The system saves data
Exit condition	The User is successfully registered to the application
Exceptions	1. The User provides inserts not valid data in one or more mandatory fields 2. The User’s fiscal code is already associated with a password 3. The User is already registered to data4Help The exceptions 1 and 2 are handled by notifying the User and taking him/her back to the point 1. The exception number 3 is handled redirecting the User at point 3.

3.2.3.11. Handle an Emergency

Name	Handle an emergency

Goals and Requirements	[G7] [R18]
Actor	Third Party
Entry conditions	2. The Third Party is already signed up and logged to the application 3. The User is already signed up and logged to the application 4. The system has calculated the correct thresholds for the User 5. The GPS system and the connection work properly 6. The application status is “active”
Event flow	1. The system registers a value that is under threshold 2. The system sends an alarm through the application 3. On the Third Party’s desktop appears a warning that there is a new emergency, the warning also contains all the information about the emergency (location, values, basic info about the User) 4. The Third Party clicks on the “handle” button
Exit condition	The emergency status is *“handling”*
Exceptions	
Special Requirements	The warning must be sent within 5 seconds after the values goes under threshold (NB: dopo che il valore è registrato o dopo che va sotto soglia?)

3.2.3.12. Detect of malfunction

Name	Detect of malfuction
Goals and Requirements	[G8] [G7] [R20] [R21] [R19]
Actor	Third Party, The User
Entry conditions	1. The User is already signed up and logged to the application 2. The system has calculated the correct threshold’s values for the User 3. The internet connection and the GPS of the device work properly 4. The User communicates his values through the device

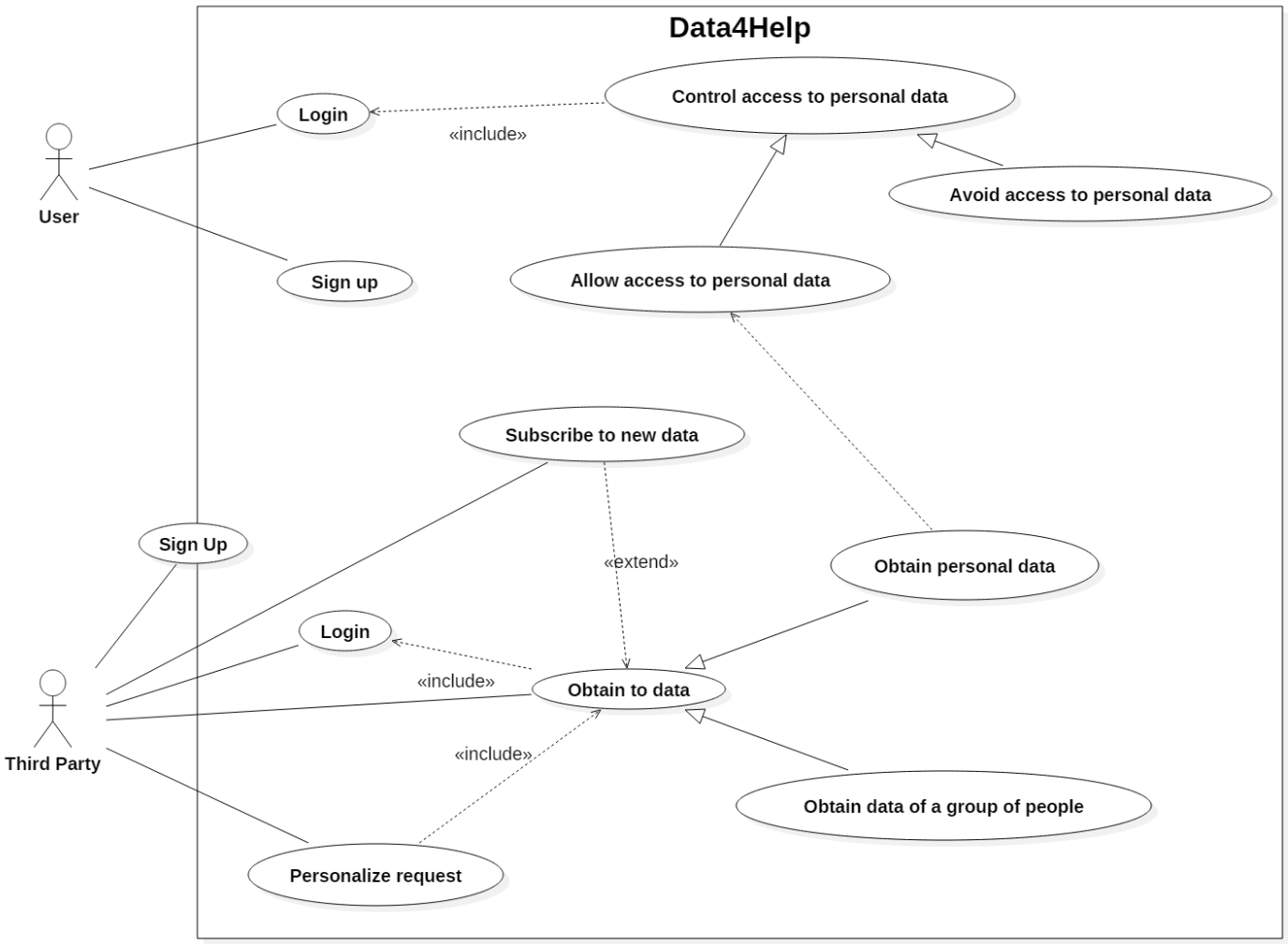
Event flow	1. The system doesn't receive new data from the device 2. The system sends a message to the User, asking if he/she is ok and notifying a malfunction 3. The User clicks on "ok" button
Exit condition	The User has been notified of the malfunction
Exceptions	1. The User clicks "no" button 2. The User does not respond after 5 minutes The management of the exception 1 is explain in the use case 11 (Handle an emergency) The exception 2 is handled by sending a message to the emergency number within an hour.
Special Requirement	The system receives new data every 500 ms

3.2.3.13. Set application's state

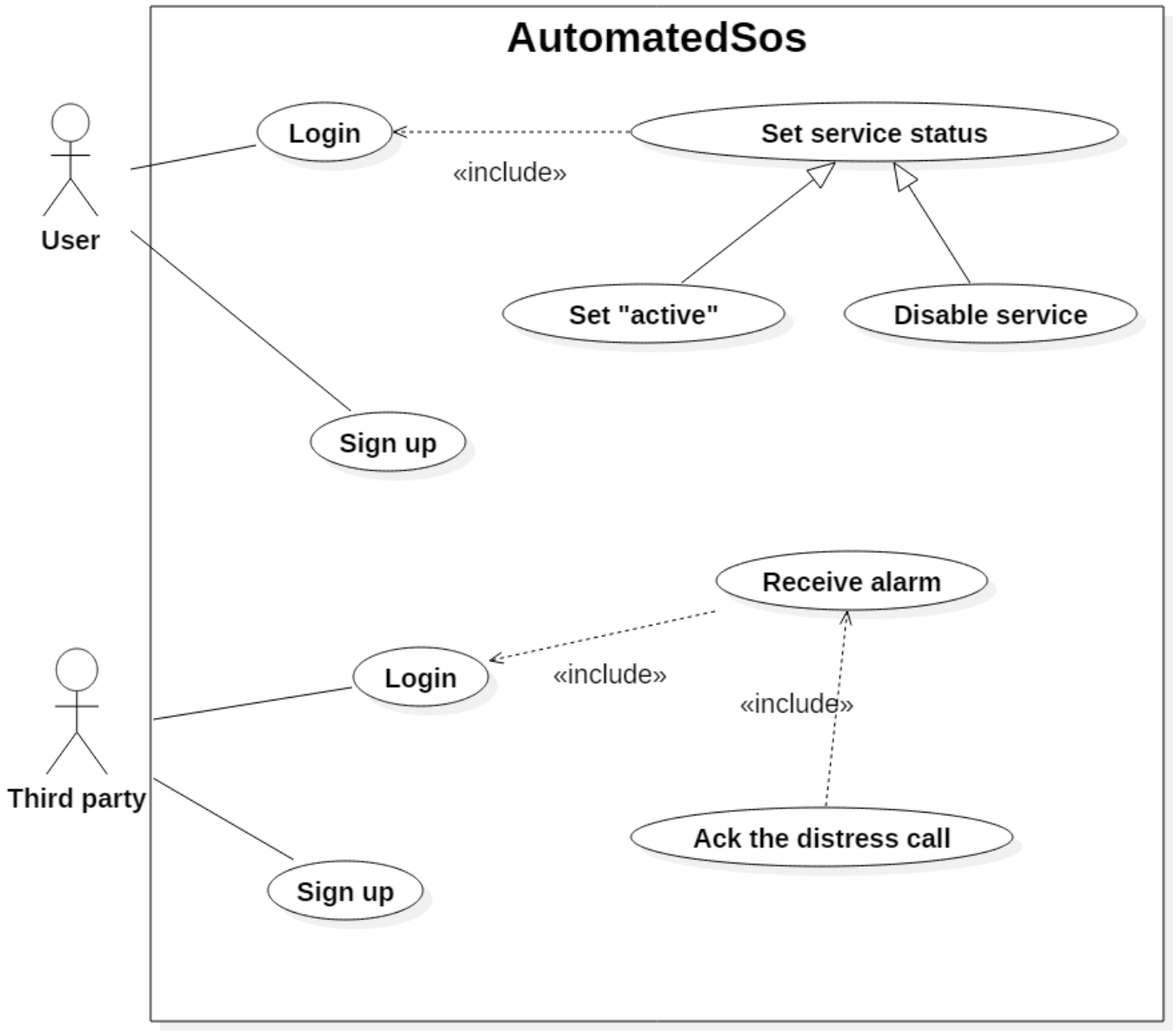
Name	Set the application's state
Goals and Requirements	[G9] [R22]
Actor	The User
Entry conditions	1. The User is already signed up and logged to the application
Event flow	1. The User opens the application on his smartphone/device 2. The User clicks on the "off"/ "on" button
Exit condition	The system status changes to "non-active"/ "active"
Exceptions	

3.2.4. Use case diagram

3.2.4.1. Data4Help

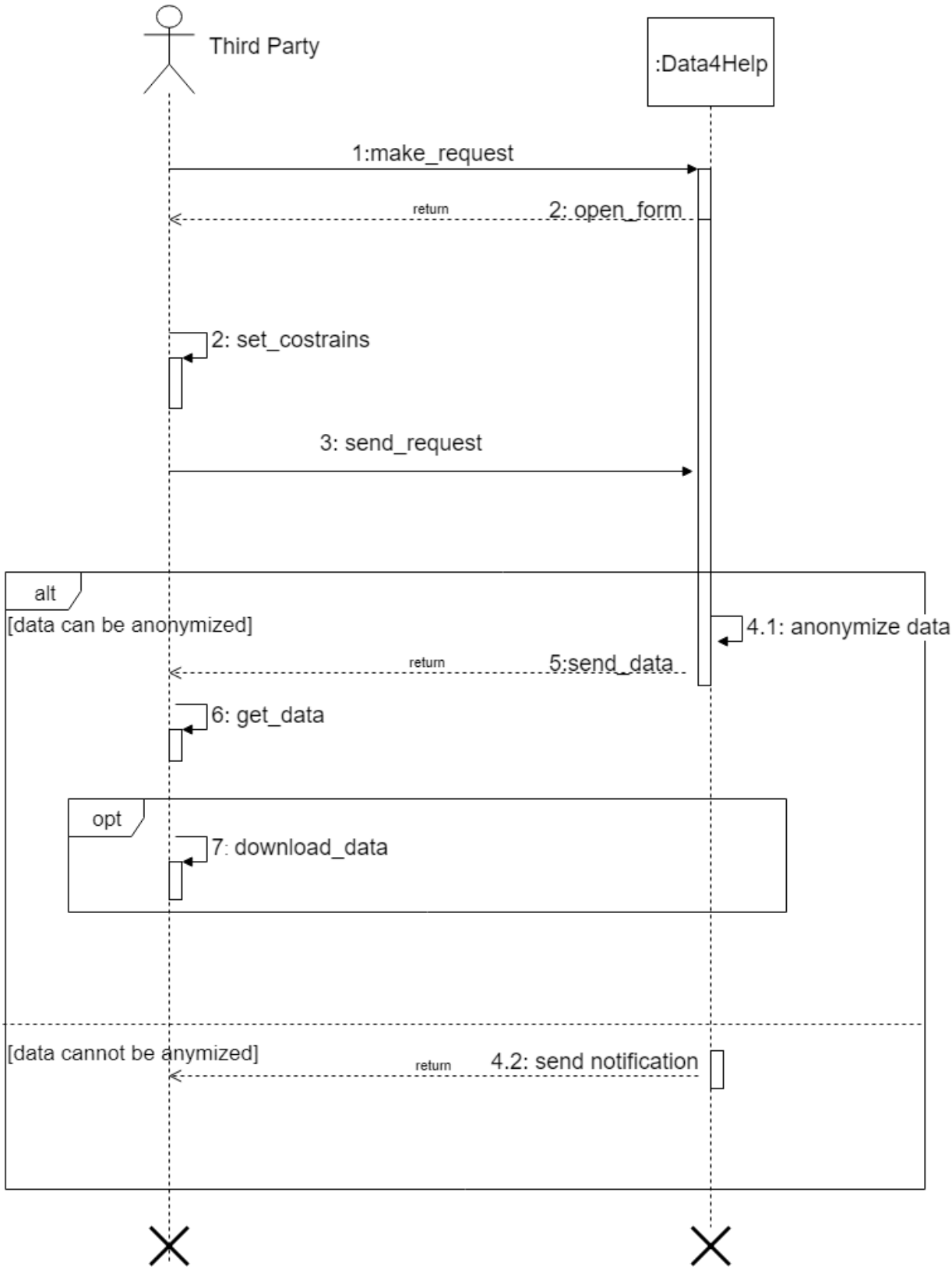


3.2.4.2. AutomatedSos

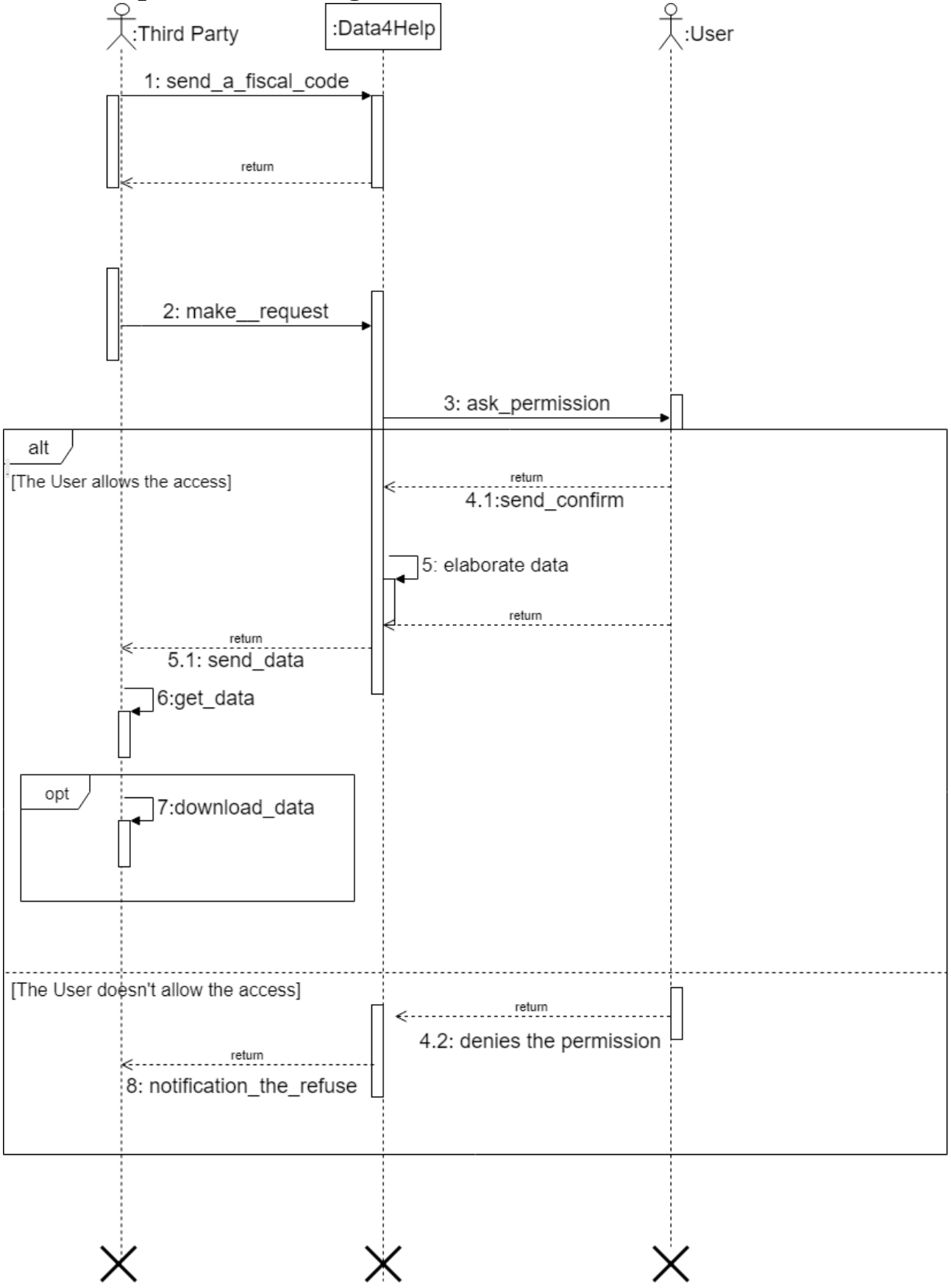


3.2.5. Sequence diagram

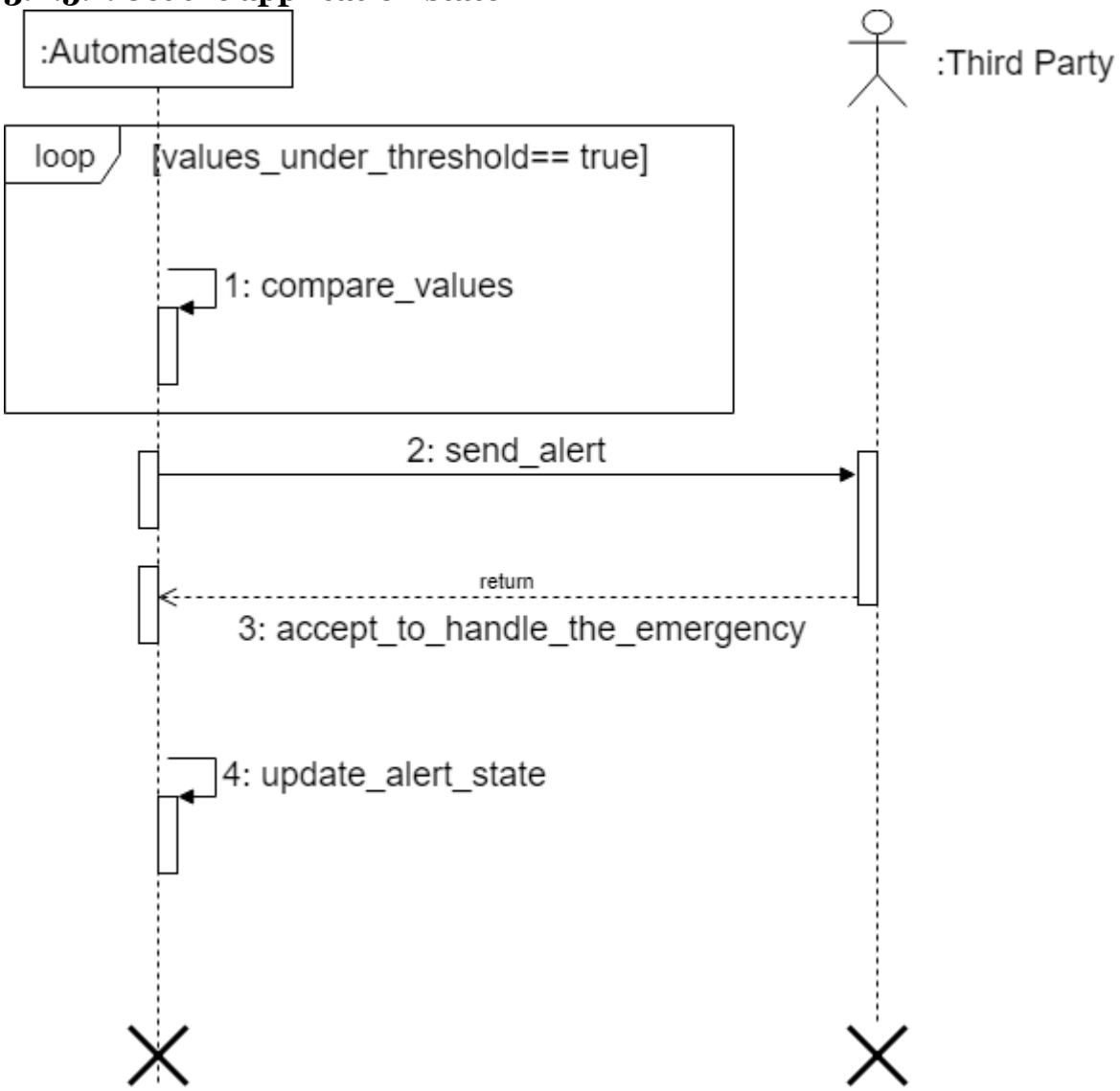
3.2.5.1. Request data of a group of people



3.2.5.2. Request data of a single User



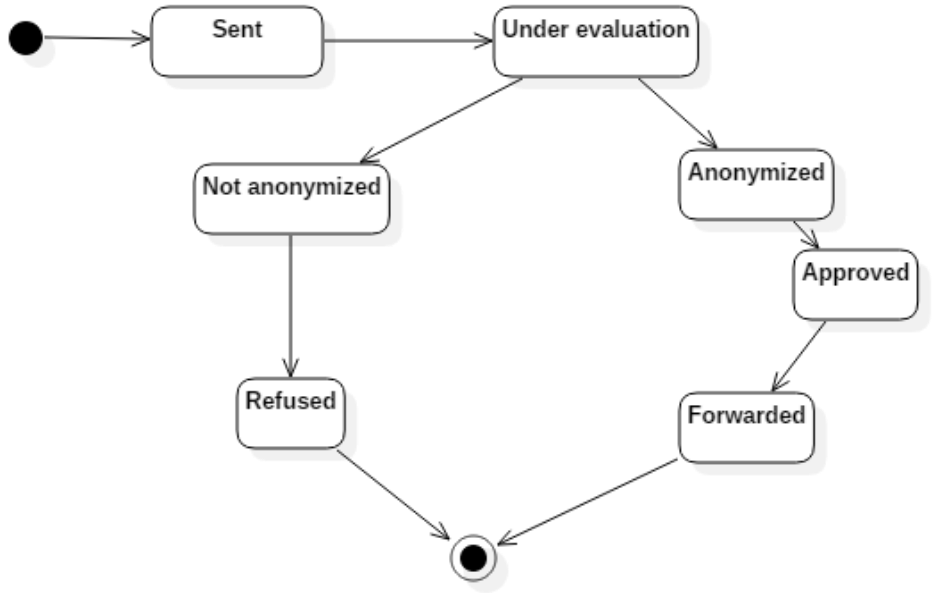
3.2.5.1. Set the application state



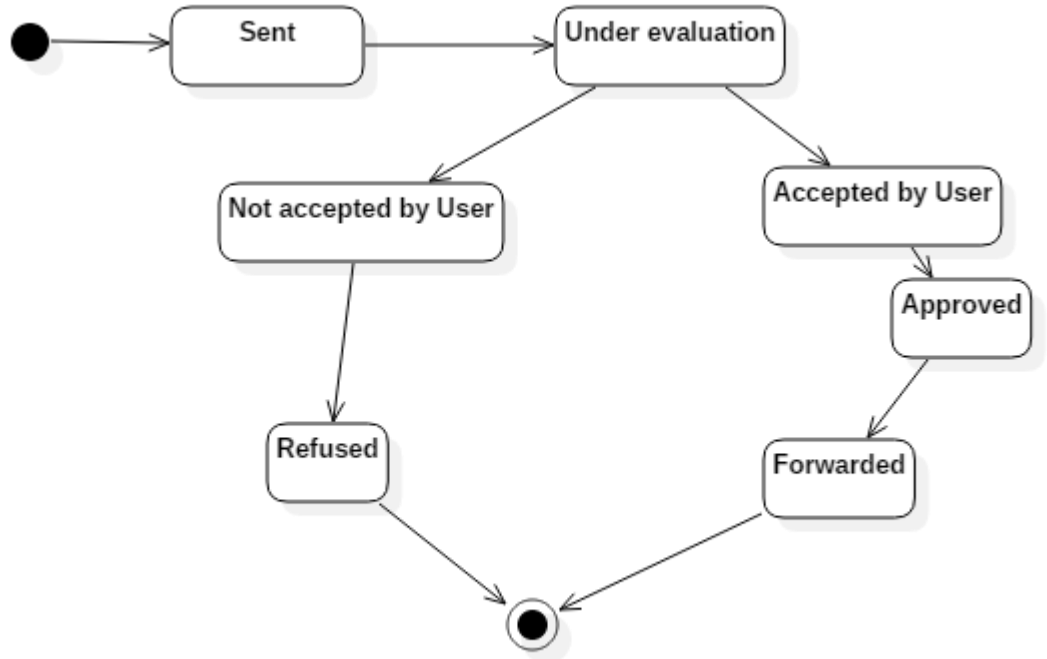
3.2.6. Statechart diagram

The following diagram shows how the status of the main "object" of the system change during the use cases.

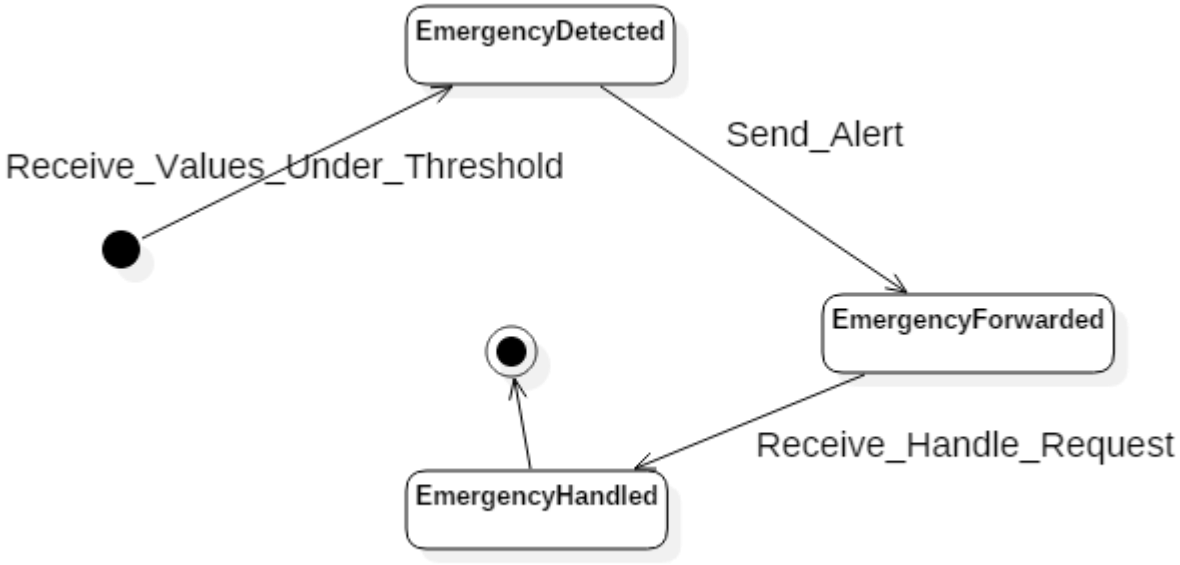
3.2.6.1. Request of data of a group of people



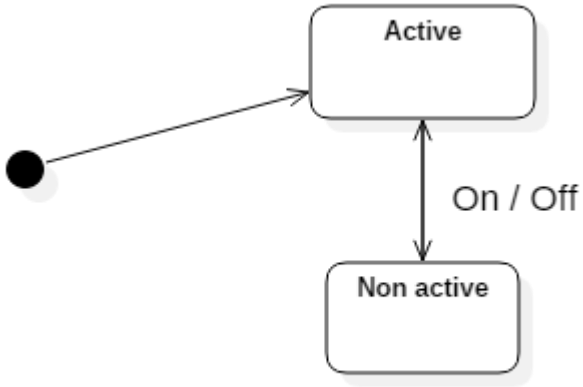
3.2.6.2. Request of data of personal data



3.2.6.3. Emergency alert



3.2.6.1. Application state



3.3. Non-functional Requirements

3.3.1. 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).

3.3.2. 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”

3.3.3. Security

Thinking of a market such as the American one, where health care is subject to negotiation between users 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.

3.3.4. 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.

3.3.5. 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.

3.4. Design Constraints

3.4.1. 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

4. Formal analysis using alloy

3.1. Alloy code

```
open util/integer
open util/boolean

//signatures
abstract sig Client{
automatedSos: one Bool
}
//all the cross product where the first element is a Int or a Time are intended to model the time instances ( both in User and ThirdParty)
sig User extends Client{
fiscalCode: one String,
status: one UserStatus,
thirdpartiesallowed: Int -> (ThirdParty -> Data),
inDangerOfLife : Int -> one Bool,
IsUnderAssistance: Int -> one Bool,
thresholds: one Int
}

sig UserStatus{
active: Bool
}
sig ThirdParty extends Client{
email: one String,
alerts: set Int,
datareceived: Int -> set Data,
groupeddatareceived: Int -> set Data
}

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: Int,
parameters: one Parameters,
accepted: one Bool
}
{
requestID > 0
}

sig IndividualRequest extends Request{
identifier: one String,
}
sig GroupRequest extends Request{
}

sig Data{
identifier: String,
parameters: one Parameters,
healthValues: one Int,
time: one Int,
}

sig Parameters{
}

// useful preds and functions
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]
}

fun ThePrevious[num:Int]: one Int{
{prev: Int | prev = num -1 }
}

//facts
//requests must regard subscribed users
fact IndividualRequestmustRegardAsubscribedUser{
  all r1: IndividualRequest |  IsSubscribedtoData4Help[r1.identifier]
}
```

```
//No two requests at the same time by the same third party
fact NoContemporary{
no disjoint req1, req2: Request | req1.requester = req2.requester and req1.time = req2.time
}

//data must regard subscribed users
fact DataMustRegardSubscribedUser{
all d1: Data |  IsSubscribedtoData4Help[d1.identifier]
}

//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
}

//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 NoSameTimeSameUserData{
no disj d1,d2: Data | d1.identifier = d2.identifier and d1.time = d2.time
}

//a user can give permission only for his data
fact OnlyMyData{
all u1:User, d1:Data, num:Int, t1:ThirdParty|  (d1 in u1.thirdpartiesallowed[num][t1] implies
d1.identifier = u1.fiscalCode)
}

//third parties e utenti non possono avere lo stesso identificativo
fact ThirdPartiesandUsersDosjointed{
all t1:ThirdParty, u1:User | t1.email != u1.fiscalCode
}

//TIME MODELLING
//a third party can't receive data from the future
```

```
fact NotFromTheFuture{
all t1:ThirdParty, d1:Data, num:Int | d1 in t1.datareceived[num] implies d1.time < num
}

fact GroupedNotFromTheFuture{
all t1:ThirdParty, d1:Data, num:Int | d1 in t1.groupeddatareceived[num] implies d1.time< num
}

//The user can't allow the third party to receive future data
fact NoAllowForTheFuture{
all u1:User, d1:Data, num:Int, s:ThirdParty|  (d1 in u1.thirdpartiesallowed[num][s] implies d1.time <
num)
}

//MODELLING REQUESTS
//if the third party has something, he has asked for the data before [done with a pred]
pred existsassociatedRequest[t1:ThirdParty, d:Data, num:Int, req:IndividualRequest, u1:User ]{
 (req.identifier = d.identifier and req.requester = t1.email and u1.fiscalCode = d.identifier and
req.time = ThePrevious[num] and isTrue[req.accepted] and req.parameters = d.parameters and
d.time <= req.time)
}
fact Onlyrequested{
all t1:ThirdParty, num:Int,  d:Data, u1:User |  (
        (num -> d in t1.datareceived )
        <=>
        (one req:IndividualRequest| existsassociatedRequest[t1,d, num,req,u1])
)
}

pred existsgroupedsassociatedRequest[t1:ThirdParty, d:Data, num:Int, req:GroupRequest]{
 (req.parameters = d.parameters and req.requester = t1.email and req.time = ThePrevious[num]
and isTrue[req.accepted] and d.time <= req.time)
}
fact groupedonlyifrequested{
all t1:ThirdParty,d1:Data,num:Int |(
        (num -> d1 in t1.groupeddatareceived)
        <=>
        (one req: GroupRequest| existsgroupedsassociatedRequest[t1,d1, num,req])
)
}

//if a user has given his permission, he has been asked for it before[done with a pred]
fact onlyifrequested{
all u1:User,  d:Data, t1: ThirdParty, num:Int |(
        (num ->t1 -> d in u1.thirdpartiesallowed )
        <=>
        ( one req: IndividualRequest|existsassociatedRequest[t1,d, num,req,u1])
)
```

```
}

//a grouped request is accepted only if the number of users involved in the request is more than 2
(arbitrary number for 1000)
fun GetTheUsers[req: GroupRequest]: set User{
{ u1:User | some d1:Data | d1.identifier = u1.fiscalCode and d1.parameters = req.parameters and
req.time >= d1.time}
}
fact OnlyifAnonymous{
all req1:GroupRequest | isTrue[req1.accepted] <=> #GetTheUsers[req1] > 2
}

//assertions checking that privacy is always respected
assert PrivacyIsProtected{
all t1:ThirdParty, num:Int,  d1: Data | ( (num->d1 in t1.datareceived) <=> (one u1:User | (num -> (t1
-> d1) in u1.thirdpartiesallowed)))
}

// interesting predicates: we want to be sure that the third parties are able to receive data in our
modelling
pred AllowThirdPartiesToGetGroupedData{
some t1:ThirdParty | some num:Int | (t1.groupeddatareceived[num] != none)
}

pred AllowThirdPartiesToGetData{
some t1:ThirdParty | some num:Int | (t1.datareceived[num] != none)
}

//AutomatedSos
//facts
//a user with no automatedSos service  is never in danger of life ( in our modelling)
fact DangerOfLifeonlyisAutomatedSos{
all u1:User, t1: Int | ( (t1 -> True in  u1.inDangerOfLife) implies isTrue[u1.automatedSos]
)}

//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 can't be  two alerts with the same id
fact AlertsIDAreunique{
no disjoint al1, al2 : Alert | al1.alertID = al2.alertID
}

//third parties must have ids in their relation corresponding to unique alerts (consistency of alertID)
fact CorrectAlertsID{
all t1: ThirdParty | all number: t1.alerts | one al: Alert | number = al.alertID
```

```
}

//there are alerts only if the user is active (again: an assumption of our modeling)
fact IfNonActiveThereareNoEmergencies{
all al:Alert | IsActive[al.data.identifier]
}

//there can't be two 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
}

//whenever a user is in danger of life, health values go below thresholds (just for that time)
fact DangerOfLife{
all t1:Int, u1:User | ( (t1->True in u1.inDangerOfLife)  <=> (one d1:Data | (d1.time = t1 and
d1.identifier = u1.fiscalCode and d1.healthValues < u1.thresholds)))
}

//an alert is created only if healthvalues are under thresholds
fact AlertCreation{
all al1: Alert |  one u1:User | al1.data.healthValues < u1.thresholds
}

//an alert is created if healthvalues are under thresholds
fact AlertCreation2{
all d1:Data | (  (one u1:User | (u1.fiscalCode = d1.identifier and d1.healthValues < u1.thresholds))
implies  (one al1:Alert | al1.data = d1))
}


// an alert is marked as "handling" if and only if there is exactly one third party which is handling it
fact ExactlyOneHandling{
all al1: Alert | (isTrue[al1.status.handling] <=>  (one t1: ThirdParty |  al1.alertID in t1.alerts))
}


//there are no alerts marked as not handling (which means, there is always a third party which
handles the request)
fact AllAlertsAreHandled{
all al1:Alert | isTrue[al1.status.handling]
}

//when a third party handles an alert, the user is provided with medical assistance (we don't care
how, it's out of our control)
fact ProvideMedicalAssistance{
all t1:ThirdParty, al1:Alert | ( (al1.alertID in t1.alerts) implies (all u1:User | ((u1.fiscalCode =
al1.data.identifier) implies ( al1.data.time -> True in u1.IsUnderAssistance))))
}
```

```
//the main goal of AutomatedSos
assert HandleEmergency{
all t1:Int, u1:User | ( ( (t1->True) in u1.inDangerOfLife) implies (t1 -> True in u1.IsUnderAssistance))
}

pred CanHandle{

some t1: ThirdParty| some num: Int | num in t1.alerts
}
```

3.2. Results

Here there are the commands and the results.

```
//commands
check PrivacyIsProtected  for 5 but  exactly 5 String, 5 Int
check HandleEmergency for 5 but exactly 5 String, 5 Int
run CanHandle for 2 but  exactly 2 String,  2  Int
run AllowThirdPartiesToGetGroupedData for 2 but  exactly 2 String,  2  Int
run AllowThirdPartiesToGetData for 2 but  exactly 2 String,  2  Int
```

Warning: JNI-based SAT solver does not work on this platform.
This is okay, since you can still use SAT4J as the solver.
For more information, please visit http://alloy.mit.edu/alloy4/

Executing "Check PrivacyIsProtected for 5 but 5 int, exactly 5 String"
Solver=sat4j Bitwidth=5 MaxSeq=5 SkolemDepth=1 Symmetry=20
194106 vars. 7737 primary vars. 548579 clauses. 16240ms.
No counterexample found. Assertion may be valid. 1180ms.

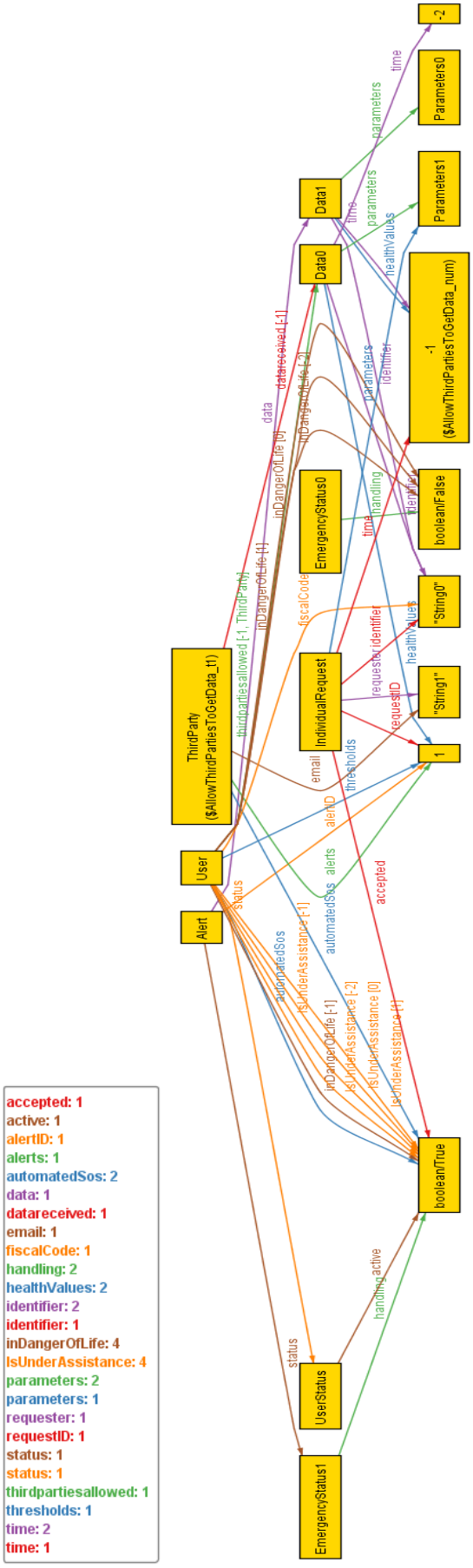
Executing "Check HandleEmergency for 5 but 5 int, exactly 5 String"
Solver=sat4j Bitwidth=5 MaxSeq=5 SkolemDepth=1 Symmetry=20
188779 vars. 7732 primary vars. 526314 clauses. 8954ms.
No counterexample found. Assertion may be valid. 5862ms.

Executing "Run CanHandle for 2 but 2 int, exactly 2 String"
Solver=sat4j Bitwidth=2 MaxSeq=1 SkolemDepth=1 Symmetry=20
3029 vars. 232 primary vars. 6357 clauses. 39ms.
Instance found. Predicate is consistent. 49ms.

Executing "Run AllowThirdPartiesToGetGroupedData for 2 but 2 int, exactly 2 String"
Solver=sat4j Bitwidth=2 MaxSeq=1 SkolemDepth=1 Symmetry=20
3047 vars. 232 primary vars. 6388 clauses. 200ms.
Instance found. Predicate is consistent. 119ms.

Executing "Run AllowThirdPartiesToGetData for 2 but 2 int, exactly 2 String"
Solver=sat4j Bitwidth=2 MaxSeq=1 SkolemDepth=1 Symmetry=20
3047 vars. 232 primary vars. 6388 clauses. 135ms.
Instance found. Predicate is consistent. 77ms.

3.3 World generated



5. Effort spent

The following table aims at summarizing the effort spent by each component of the group, in discussing, examining solutions and writing them. After working together in the first phase to define the core of the project, everyone focused on some specific topic. The data are expressed in hour.

	Francesco Pontiggia	Arianna Ricci
Purpose and Scope	12	8
Requirements	10	10
Diagrams	3	14
Use Cases	1	10
Alloy	12	2
Revision	5	5

6. References

1. Star UML 2.81
2. Draw.io
3. Proto.io
4. Alloy Analizer 4.2
5. Specification document “Mandatory Project Assignment AY 2018-2019”