Health, demographic change and wellbeing

Personalising health and care: Advancing active and healthy ageing

H2020-PHC-19-2014

Research and Innovation Action

ACANTO

A CyberphysicAl social NeTwOrk using robot friends

Deliverable 2.4

User, activity and environmental description: Final release models for user, activity and environment

Deliverable due date: 1.2.2017	**Actual submission date: 13.3.2017**
Start date of project: February 1, 2015	**Duration: 42 months**
Lead beneficiary for this deliverable: ATOS	**Revision: —-**
Authors: Ivo Ramos (ATOS), Ingo Brauckhoff (ATOS), Paolo Bevilacqua (UNITN), Luigi Palopoli (UNITN), Pashalis Padeleris(FORTH)	
Internal reviewer: Daniele Fontanelli (UNITN)	

The research leading to these results has received funding from the European Union's H2020 Research and Innovation Programme - Societal Challenge 1 (DG CONNECT/H) under grant agreement n°643644

Dissemination Level		
PU	Public	**X**
CO	Confidential, only for members of the consortium (including the Commission Services)	

Contents

Executive Summary

This deliverable describes the models that present the User Profile, Activities and Environment. It reflects the results of the work made with the models used for the user profiles, activities and environment. These models will support the development of the CPSN and, in particular, of the following components: activity recommendation (**ActivityGenerator**), activity planning (**ActivityPlanner**), activity execution, monitoring and evaluation.

The models in the deliverable are on:

1. The user profile, i.e., how each user is described along with her/his requirements and preferences;
2. User Circles, i.e., how groups of users with similar interest and requirements can be modeled;
3. Activities, i.e., how the possible activities can be characterized and proposed to the users;
4. Environment, i.e., how the environment where the users live and the activities are executed is represented.

These models are at the heart of the design of the Cyber Physical Social Network (covered in WP4), and the Activity Planning (covered in WP5). The document also has an important link with **D2.5/D2.6**, where we discuss formal languages and technologies to generate plans that make an activity executable. Another important link is with the **API** covered in D7.1, where we discuss how plans and activities are concretely processed and executed. Finally, the activity and environment models are also Input for Task 5.2 in the planning work-package.

The models presented in this deliverable have now reached a good level of maturity and they are currently used in different parts of the implementation of the ACANTO Infrastructure. We expect that a few changes could be required during the validation and testing phases of the different modules. In this sense, the document is also “alive” and it is subject to possible updates until the end of the project.

1 Introduction

This document presents the final version of the User, Activity and Environment models used as input and output for the CPSN, Activity Generation (preferences of the user, state of the user and Environment opportunities), Activity Planner and Execution. It reflects the update of the models presented in the previous deliverable *D2.3 – "User, activity and environmental description (preliminary)"* [2]. The models illustrated in this deliverable have been developed using the Unified Modelling Language (UML) [1], explicitly the Class and Sequential diagrams. Due to the absence of specific diagrams for graphical databases, we took the liberty to create our own diagrams, identified as "proprietary diagrams".

The **User Profile** model is the general description of the data representation and relationships for the users of the system. It intends to centralize data coming from different sources, mainly personal information, preferences, interests, mobility constrains, previous activities, etc.

The **Activity** model shows the relationship of the activity with the user that executes it and the activity evaluation provided by the user.

The **Social Activity** models are related with group activities, like walking in the park with someone else, going shopping or to an art exhibition. It is crucial to understand the possible characteristics of social activities that could be relevant for a better recommendation and organization of activities.

The **Circles** define the concept of "group", necessary to define activities with a social dimension. Essentially, they will be generated between people with compatible profiles and with common interests.

The **Environment** models represent the definition of the "life zones", i.e., the region inside the environment in which prominently the users carry out their daily activities.

1.1 Use cases

In order to make a clear connection between the models presented in this deliverable and the general framework of ACANTO, we will use throughout this document three use cases, as narrative examples, related to typical scenarios of interest as defined from the work carried out in WP1.

First Use Case

Isabel is an 82 year old woman who has lived alone for the past two years. She lives in a flat by herself in Newcastle. She no longer goes out very often and has become very physically inactive. Her daughter brings her groceries once a week. While Isabel used to enjoy going for walks in the local park, she no longer has anyone to go walking with. She recalls the times she spent walking with fondness and wishes that she had someone to go walking with. One day, she receives an invitation by mail to try out the new *FriTab* and *FriWalk* system. After receiving the system when a researcher visits her, she tells the system about her background and interests. She tells the system that she used to enjoy walking. Later that day, the *FriTab* suggests that she meet a lady in the next street, Martha, who likes to visit the local shopping mall and has similar mobility problems. The system has noticed that Isabel does not have many friends and believes that if she had a friend who also enjoyed going for walks, she might go there again. Isabel is hesitant at first but then agrees to meet up and try out the *FriWalk*. The *FriTab* tells Isabel to meet Martha the following Wednesday at 10am to enjoy a morning together at the shopping mall. Once she arrives to the shopping mall, the *FriWalk* shows the directions to get in touch with Martha at the prescribed time. Since Martha has a similar *FriWalk*, the two ladies meet with any problem. Isabel and Martha go for a walk in the mall and decide to buy some groceries. The *FriWalk* suggests the route and monitor the execution of the activity, which is based on medical prescriptions. Both Isabel and Martha wear the anklets, which set a comfortable pace for them to walk at. During the walk, the *FriTab* realises that Martha feels a little bit tired and suggests to interrupt with the planned activity. The *FriTab* suggests Isabel and Martha to have lunch in a local cafe. The *FriWalk* devices guides them to it where they have a pleasant lunch and agree to meet up again.

Second Use Case

Michael is a 72 year old man who lives alone in Felling, Gateshead. For the past few years, he has found mobility very difficult and he is waiting for a hip operation. Consequently, he doesn't get out much. He used to enjoy visiting museums and now fulfils his passion for natural history by watching documentaries on TV. He would like to be able to get out to visit the museums in Newcastle.
A researcher visits Michael one day and shows him the *FriTab*. Michael explains to the researcher that he has mobility problems so he can't walk for long periods of time. But the researcher explains the *FriWalk* to him which is owned by several shops, galleries and museums in the area. So Michael enters his details into the system and tells it that he has mobility problems.
The next day, the *FriTab* shows Michael that a tour is being organised at the Hancock Museum. It invites him to attend and tells him that the museum owns *FriWalk* devices that can help him. Michael is hesitant but agrees to give it a try. So he tells the system that he will attend.
The *FriTab* tells him when the tour starts and where in the museum it will start from. When he arrives at the museum, he is given a *FriWalk* which helps him to walk with the rest of the tour group. After the tour is over, the *FriWalk* even suggests a guided tour of its own that Michael can do alone. Michael is tired but decides to come back and try the guided tour another day. The *FriTab* forwards the activity log file to the network for user profile update.

Third Use Case

Dorothy is a 69 year old woman who lives alone in Blaydon. Dorothy uses a walker to get around because she finds that it gives her confidence after her fall one year ago. Dorothy loves shopping and likes it when her friend occasionally takes her shopping at the MetroCentre. While she likes the MetroCentre, she is nervous about going there alone and worries that she would get lost. But she would like to go there more often.
Dorothy is shown the *FriTab* and told that it clips onto *FriWalk* devices which are available at the MetroCentre. She decides to try out the *FriTab* system. Several days later, the system suggests that Dorothy visit the MetroCentre to enjoy some shopping. The system has noticed that Dorothy has stayed indoors for several days and believes that she would benefit from getting out. Dorothy thinks that it would be a good idea and asks the *FriTab* for more information. The *FriTab* suggests that she get the 2:15 bus from the nearby bus stop which will take her to the MetroCentre. It tells her that it will give her directions to the MetroCentre and will help her find her way around inside.
She gets the bus and travels to the MetroCentre. When she gets there she swaps her walker for a *FriWalk* and clips in her *FriTab*. The *FriTab* shows her that several shops

have sales and gives her directions. When she starts feeling tired, she presses a button on the *FriTab* and it directs her back to where her walker is. She unclips her *FriTab* and it tells her where to get the bus home.

Chapter 2

2 Accessory Packages

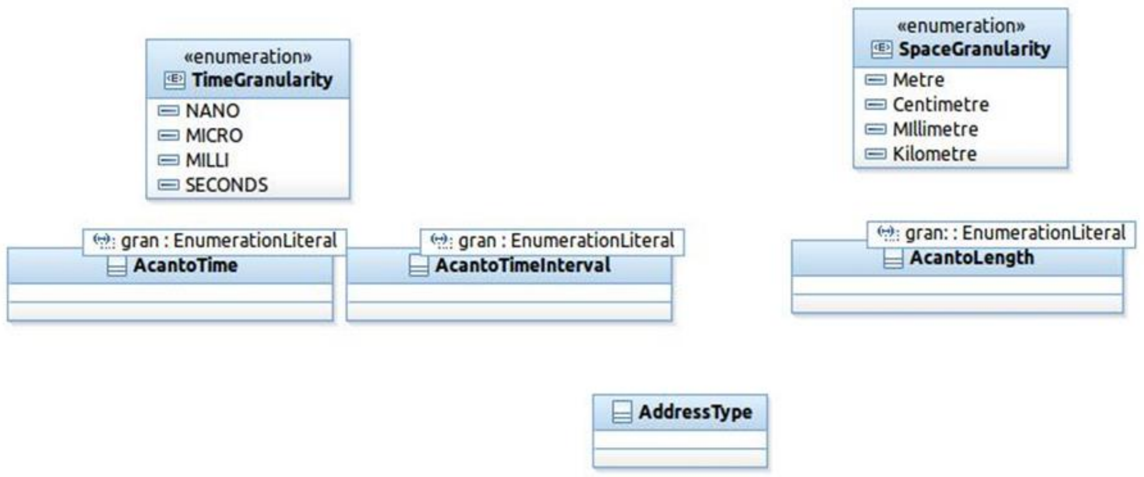


Figure 1: General Types

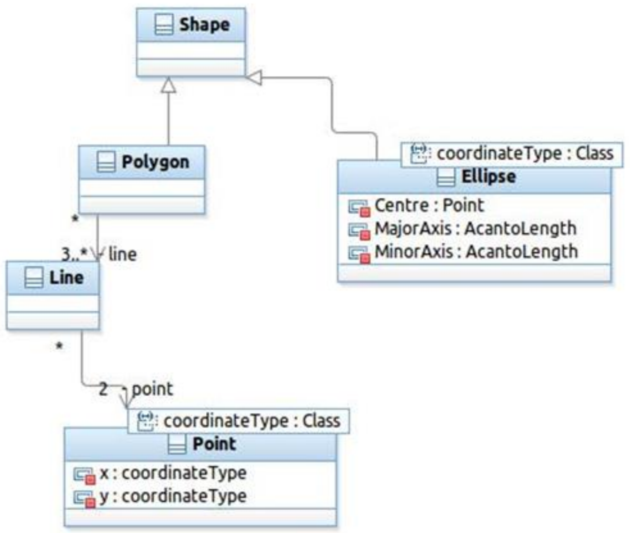


Figure 2: Geometric Entities

Two packages contain a set of classes' instrumental to the models' construction. The first one is **GeneralTypes**, which contains a number of classes used for the definition of objects of general applicability. Its content is evolving while newer functionalities become part of the project. A snapshot of its current state is presented in Figure 1. The classes **AcantoTime** and **AcantoInterval** are used to represent time. The first one represents a time instant. Adopting the well-known standard POSIX convention, the definition of a time instant refers to the beginning of the UNIX era (January, 1st, 1970). The second one is used to express intervals between two time points. In the same way, the **AcantoLength** class is used to express metric distances. Both times and distances are modelled as template classes of a granularity enumerated type, which for the time ranges in the set MILLI, NANO, MICRO, SECONDS and for the length ranges in the set Metre, Centimetre, Millimetre, Kilometre. Templates are naturally bound to a C++/Java

language construction and are very useful to avoid insidious programming errors, such as confusing measurement units. For instance, if a method is defined with a formal parameter expressed in Milliseconds, it simply cannot be bound onto an actual parameter expressed in Seconds, unless an explicit cast is made.
The other package contains the different geometric entities used, is shown in Figure 2: Geometric Entities. The geometric package is based on the notion of **point**, which is modelled as a template class. The template parameter is the coordinate type (which can be a double or any other scalar type). Points can be composed to form **lines** and **polygons**, which are one of the possible geometric shapes that can be used. Another type of shape is an **ellipse**, which is defined by the centre and the axis.

Chapter 3

3 User Profile

The user profile aims to collect data coming from different sources such as personal information, preferences, interests, mobility constraints and previous activities. The user profile model is the general description of the data representation and relationship for the users of the system. The main parts of the user profiles are the personal information (first name, family name, complete name, birthday, civil status, the social profile (likes, dislikes and preferences), and the mobility record (constraints and prescriptions). The **¡Error! No se encuentra el origen de la referencia.**, illustrates the main blocks of information to present the user at the ACANTO system together with the user circles and location. This also reflects the update of the user profile models presented previously in the document *D2.3 – "User, activity and environmental description (preliminary)"* [1]. The diagram uses a style similar to the UML class diagram, but actually reflects the graph database schema. The similarity is useful since *orientDB* [3] describes its schema with classes as well. The color-key to the diagram is:

- light blue = vertices
- green = edges,
- light red = embedded documents
- purple = indices

The initial user profile will be generated after answering to an initial questionnaire and will be enlarged and developed subsequently with information provided by formal caregivers, relatives and medical doctors.

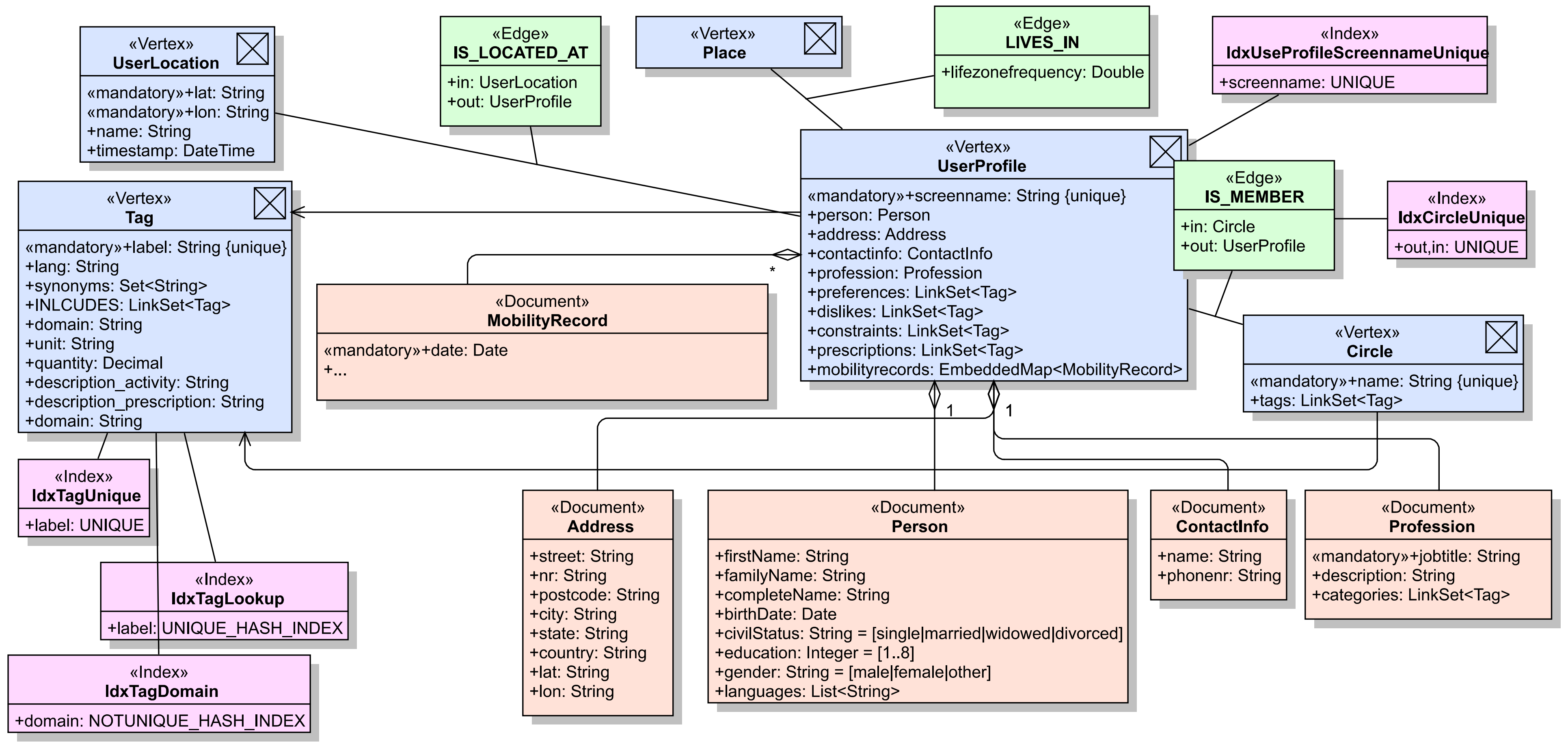


Figure 3: User profile schema

Graph Representation of the User Profile: as mentioned above, the graph representation is another mode to present information in a database, which is used in graph-databases such as *OrientDB.* [3] We use the graph representation because a social network is a natural use case for these systems. The representation is based on the graph theory, which characterizes 3 main elements: **vertices**, **edges**, and **properties**. The **vertices** are the main entities, the **edges** are the relationships within nodes, and the **properties** are characteristics that describe these two element types.

The user profile decomposes in the following way: a Vertex of type (or class) ***UserProfile*** with properties resembling embedded documents that are individually specified by their respective classes: *profession*, *person*, *address*, *contactinfo*, etc. *OrientDb* allows not only to store properties with a primitive type, like integer values or strings, but also whole embedded documents, links, sets, etc. (see *OrientDB* documentation).

It is worth mentioning the ***MobilityRecord*** structure, which is embedded into the *UserProfile* as a Map (*mobilityrecords* property) whose keys state the date of the day the record is related to in '**`yyyymmdd`**'-format. The linked document of class *MobilityRecord* has a property *date* which matches the key of the Map. The *MobilityRecord* will hold the data from the sensors and will be used to derive some statistics from these.

Circles are used to group *UserProfile*s and state only the name of the group as a property. The grouping is realized by connecting one or more *UserProfile* nodes to the *Circle* node through ***IS_MEMBER*** edges. A *UserProfile* can be part of a *Circle* only once – this is guaranteed by the unique index ***IdxCircleUnique***, which is generated on the *in* and *out* properties of the *IS_MEMBER* edges.

Associated to the *UserProfile* via the *IS_LOCATED_AT* edge is the *UserLocation*. It holds geolocation information *lat* (latitude) and *lon* (longitude) as well as a timestamp. This type of nodes will be useful to track the current location of a user.

Figure 4: User profile relations shows the relations, marked by edges in the graph, of the *UserProfile* nodes between instances of the same type. We defined the following types of edges:

- ***FOLLOWER_OF*** and ***CONNECTED_TO*** describe the network of followers and connections of the *UserProfile*. While *FOLLOWER_OF* is a unilateral relation, *CONNECTED_TO* resembles a bilateral or 'friendship'/'family-member' relation.

- ***CAREGIVER_OF*** will be used to give special rights to view and/or alter the contents of this connected *UserProfiles*. *CAREGIVER_OF* edges originate from *UserProfile*s which take the role of a doctor, therapist or nurse, and the amount of data that can be revealed this way will be adjusted accordingly. A doctor for example will have the most ample view on the collected data, while a simple caregiver might just be allowed to query the current location of the patient.

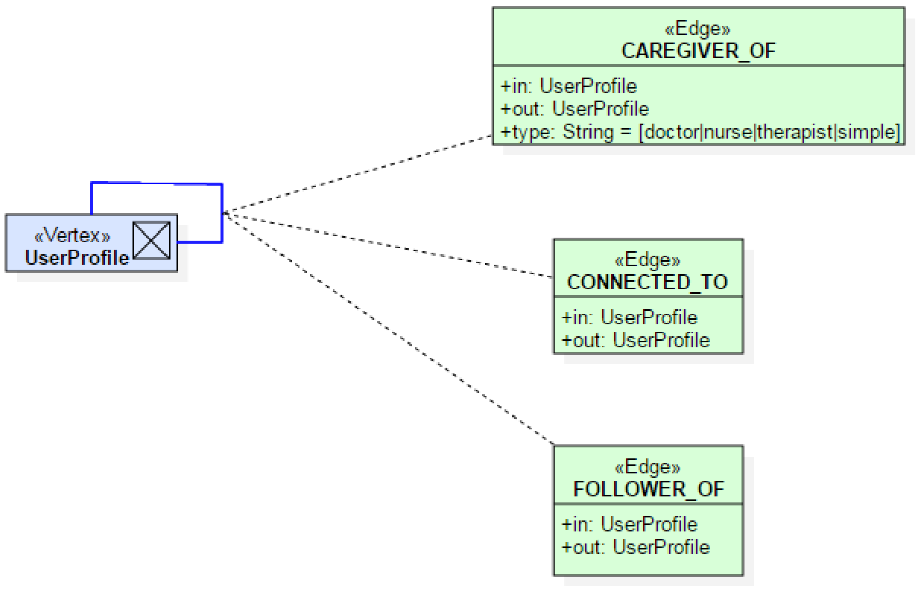


Figure 4: User profile relations

Tag* nodes:** throughout the system we will use tags to classify or categorize the nodes they are linked to, and therefore tags provide a semantic layer. They will be used by the recommender systems to match comparable ***UserProfiles, group them into ***Circles*** and to find appropriate ***Activities*** for a given user. The important feature about them is their concise way to describe an activity or a person with just one or two words, easily to understand and interpret and most importantly, to limit the amount of different descriptions used to a smaller and controllable vocabulary.
The properties of the *Tag* nodes are described in the following table:

Label	`String`	the name of the tag, semantic purpose
Lang	`String`	the language in which the label is given; optional
synonyms	`Set<String>`	collection of labels with the same meaning
domain	`String`	a domain name to separate different applications of tags, defaults to "" (empty) as free domain and can take value "mobility" for use with the mobility profiling system.
description_activity	`String`	***mobility* domain only**: an explanatory description for the tag, in contrast to the concise way of the label. This description will be used when tagging *Activity* nodes.
description_prescription	`String`	***mobility* domain only**: an explanatory description for the tag, in contrast to the concise way of the label. This description will be used when tagging *UserProfile* nodes.

Table 1: Tag node properties

KNOWLEDGE BASE

The Knowledge Base aggregates data from user profiles, circles, tags, evaluations, environments and activities, as shown in the Figure 5: Knowledgebase diagram. This package is used to differentiate the classes used for the recommendation system. The Knowledge Base will be accessible to the other components of the ACANTO information system through a Representational State Transfer Application Programming Interface (REST API) [4] or native Application Programming Interface (API) for a variety of programming environments [5], allowing easy re-use, scalability and support to a number of evolving services.

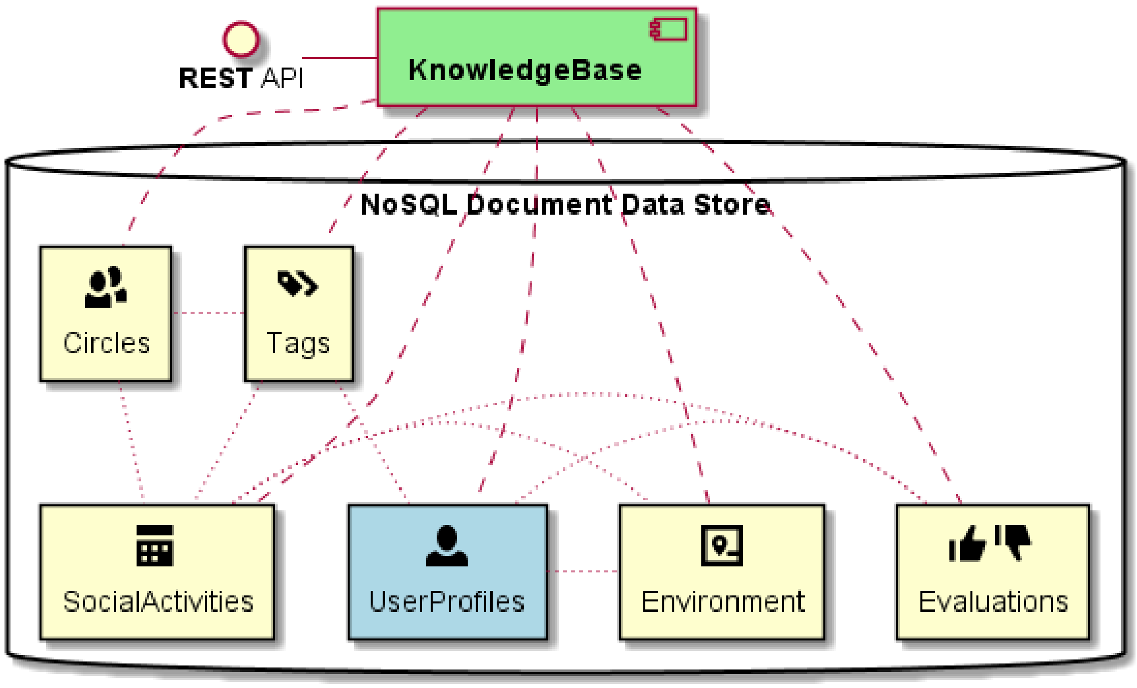


Figure 5: Knowledgebase diagram

3.1 Circles

The **Circle** is a specific concept of “group”, necessary for planning activities with a social dimension. They are generated between people with compatible profiles and with common interests.

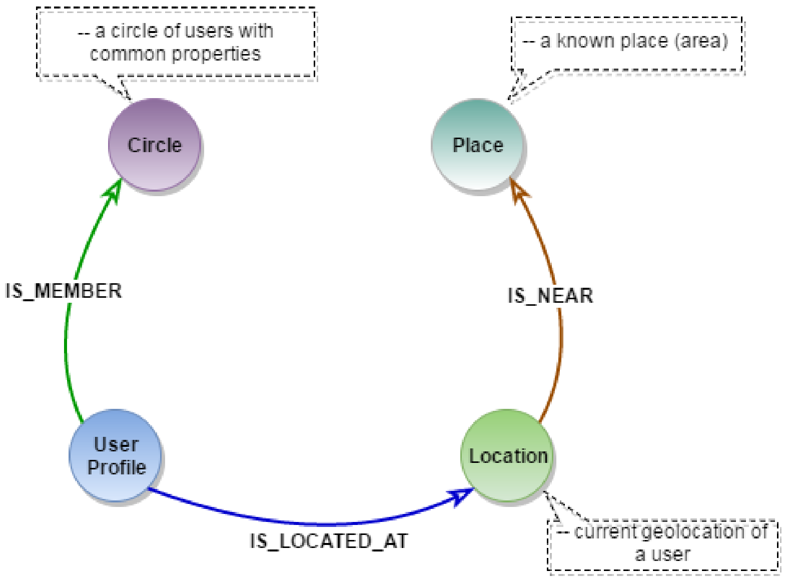


Figure 6: Vertex relations of the User profile, Circle, Location and Place

Figure 6: Vertex relations of the User profile, Circle, Location and Place, is a proprietary diagram which shows the vertex relations between ***UserProfile***, ***Circle***, ***Location*** and ***Place***. *Circles* are nodes specifically used to group *UserProfiles*. A *UserProfile* can be a member of one or more circles (declared by IS_MEMBER typed edge) and is associated with a Location node that holds information about the current geolocation of a user. The *synonyms* property collects *Tags* with the same semantic meaning, for example in another language. A *language* property states the language the tag is labelled in. The activity recommender system developed in Task 4.4 uses this information of *UserProfiles* and *Circles* to generate its recommendations.

Example: Considering the first Use Case, when Isabel first connects to the CPSN, she has to fill in a questionnaire to complete her personal data and preferences. The questionnaire also wants information about her profession and whether she has any experience with foreign languages.
After her sign up, her physician completes the user profile of Isabel with the necessary prescriptions and health constraints. He does this by adding some of the available mobility tags: 'needs to have toilets nearby', 'Needs to walk longer distances (>1km)'.
The recommender system afterwards uses this information to find the right group of people and recommends joining a Circle of 'walking' people. From there she gets recommendations to activities of walking every day. She also gets a personal recommendation to visit the mall, since there is a 'black-friday' event announced for the following week.

Chapter 4

4 Environment Description

The environment description encompasses both static and dynamic information about the world.

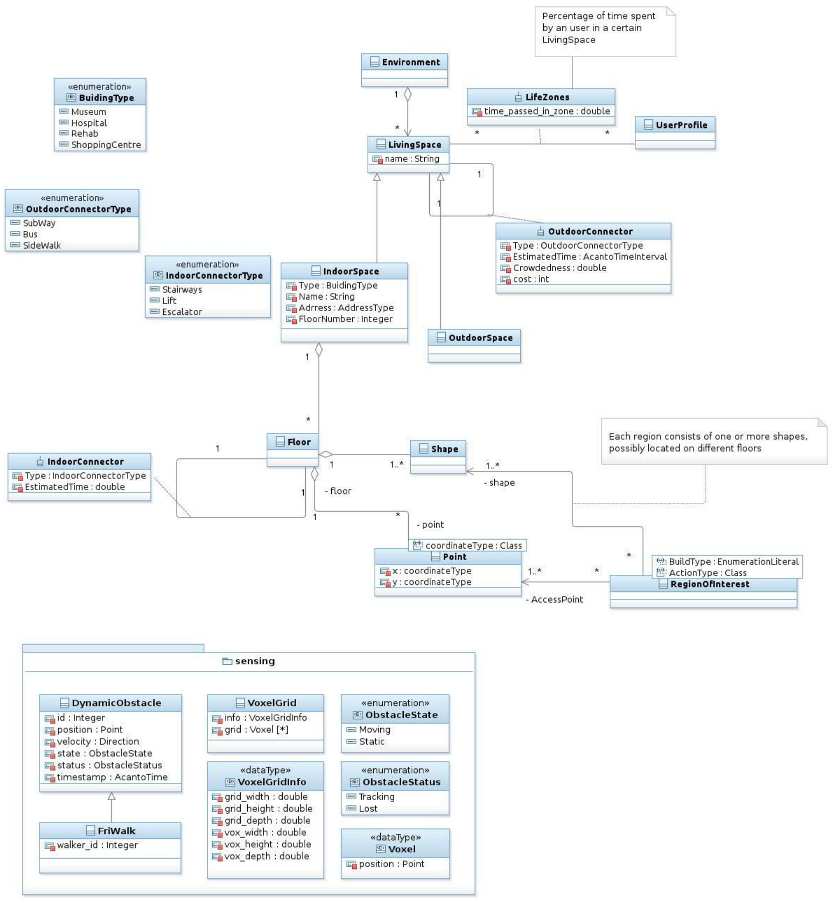


Figure 7: Class diagram offering an overview of the environment description

The static information refers to knowledge about the world that does not change or does not change regularly. This includes both metric information (required to produce detailed plans) and information on the “semantics” of the different places. The static environment is defined as a collection of **LivingSpaces** (Figure 7). Each one of them is defined as a place where the user can carry out activities. Examples are the user’s residence, the shopping mall, the park, the museum, etc. Living spaces are connected by **OutdoorConnectors**. Each connector has a type (**SubWay**, **Bus**, **Sidewalk**), an estimated time to travel, an average level of crowdedness, and a cost (e.g., the bus ticket). Living spaces can be of two types: **OutdoorSpaces** and **IndoorSpaces. IndoorSpaces** have a type (**Museum**, **Hospital**, **Rehab and Shopping-Centre**), a name (e.g., “Royal Gallery”), an address, and a number of floors. Indeed, each indoor space is a collection of floors which are connected by an **IndoorConnector**. An **IndoorConnector** has a type (**Stairways**, **Lift** and **Escalator)**. Within each floor, it is possible to define points and shapes (see Chapter 0), which are associated with a **RegionOfInterest**. A **RegionOfInterest** spans over one or multiple shapes (possibly located on different floors) and has one or more access points. The former models the portion of the space allocated to the region, while the latter models doors or other types of entry/exit points. For instance, a shop covers a well-defined area within a mall and has a clearly identified entry point (typically a door). Less intuitive is the case of a museum. In this case, the space in front of an art work (where people typically stand or sit to admire it) define a region of interest. In this case, the entry points could be set on the boundary of this area (e.g., taking into account for the presence of any obstacles or sitting places).

The dynamic information refers to knowledge about moving objects (i.e. obstacles, people, and other *FriWalk* walkers) in the direct vicinity of the walker. As the *FriWalk* operates in a dynamic environment, the activity and path planning modules need to take decisions based on up-to-date information about the surroundings of the walker. Towards this end, the environment model is augmented with real time information produced by the onboard sensing components (perception package of T3.2) as well as real-time information from the cloud when available. This information is modeled in the environment class diagram of Figure 7 by the classes grouped under "sensing". The **DynamicObstacle** class encodes information about humans and moving obstacles**.** Each obstacle has an id and fields such as position, direction and velocity of movement associated with it. This information is updated in real time as objects are detected and tracked using the front facing RGB-D sensor of the walker. Obstacle information is consumed by the short term planning module in order to suggest a safe route to the user. The **FriWalk** class is a subclass of DynamicOstacle and encodes information about other walkers in the area. In contrast to the generic dynamic obstacle, information about walkers is made available through the cloud. Each walker that has a connection to the ACANTO cloud services uploads its telemetry information (id, current position, speed etc.). The information can subsequently be made available to other connected devices. The **VoxelGrid** class encodes lower level information about the objects in front of the walker. The space is discretised into cubes (or voxels), of a fixed size and the occupied voxels are sent to the environment model. Each voxel encodes its position in the global coordinates system. The VoxelGrid information is consumed in real time by the planning modules.

4.1 Regions of Interest

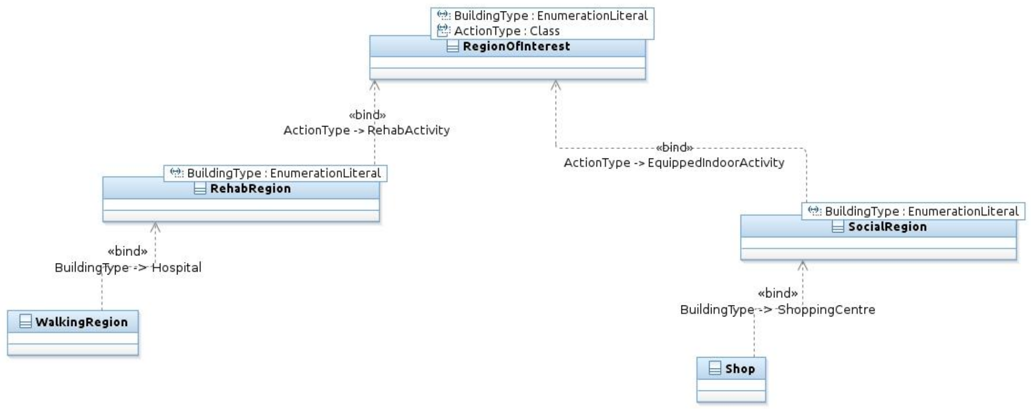

BuildingType : EnumerationLiteral
ActionType : Class
RegionOfInterest
«bind»
ActionType -> RehabActivity
«bind»
ActionType -> EquippedIndoorActivity
BuildingType : EnumerationLiteral
RehabRegion
BuildingType : EnumerationLiteral
SocialRegion
«bind»
BuildingType -> Hospital
WalkingRegion
«bind»
BuildingType -> ShoppingCentre
Shop

Figure 8: Examples of how different region types are instantiated binding the parameters

A **RegionOfInterest** is defined as a parametric type, where parameters can be the type of the building and the type of activity that can be organised using the region. An example of how the different parameters can be bound generating a hierarchy of regions is shown in Figure 8. The parameter **ActionType** (see Chapter 5) can be bound to **RehabActivity** to generate a family of **RehabRegion**, while setting it to **EquippedIndoorActivity** generates a family of **SocialRegion**. By further binding the **BuildinigType** parameter to **Hospital**, it is possible to generate a **WalkingRegion**, where the user performs walking exercises in a controlled environment. Likewise, binding the **BuildType** of **SocialRegion** to **ShoppingCentre** we can generate a **Shop**.

4.2 Life zones

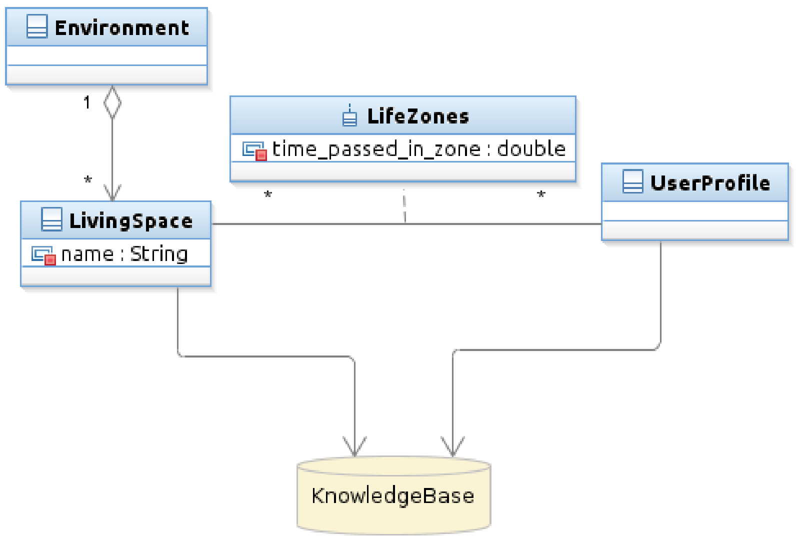


Figure 9: Life zones, users profile and their connection with the knowledge base

The life zones are defined as the region inside the environment in which prominently the user carries out his/her daily duties (see *T4.1 – User Profiles harvesting*).
The life zone is defined when the user profile is created and the environment is set for a particular user, storing the information into the Knowledge base.
As an **example**, in the case of the Use Case of Dorothy, the living spaces could be: home (85% of her time), minimarket (10% of her time), MetroCentre (only 2% of her time, because of her concerns).

Chapter 5

5 Models for Activity Generation

5.1 Activities

The following activity models are part of the subsystem *Conception of social activities* and they partially form part of the input of tasks T4.4 – “Social Activities Recommendations” and T5.1 – “Activity Planner”. Inside this subsystem, the *User Profiler*, collects relevant information on the user from different sources and consolidates it with the past observations to create and update the profile (see Figure 10 - Conception of Social Activities subsystem). It is composed by a “medical” part (which is strictly confidential) and a “social” part, which derives from previous observations and from the participation to the social network. The user *preferences* are what the user enjoys. The *constraints* are the mobility limitations and the *health prescriptions* contain specific social interaction and physical exercise. The *Activity Generator* is the module where the recommended Activities are chosen considering the *preferences of other users* that belong to the same circle, the *state of the user* and the *opportunities suggested by the environment* (e.g., special events planned for the day etc.). **In the case of the clinical scenarios** the decision of the activity can be directly made by formal caregivers. The *Activity Monitor* (T5.3) [6] monitors the progress of an activity (e.g., the number of planned locations that have been visited) and decides changes to the activity in case of significant deviations (e.g., the user is too tired to proceed) or of unpredicted events (e.g., one of the locations is not available or is overcrowded).

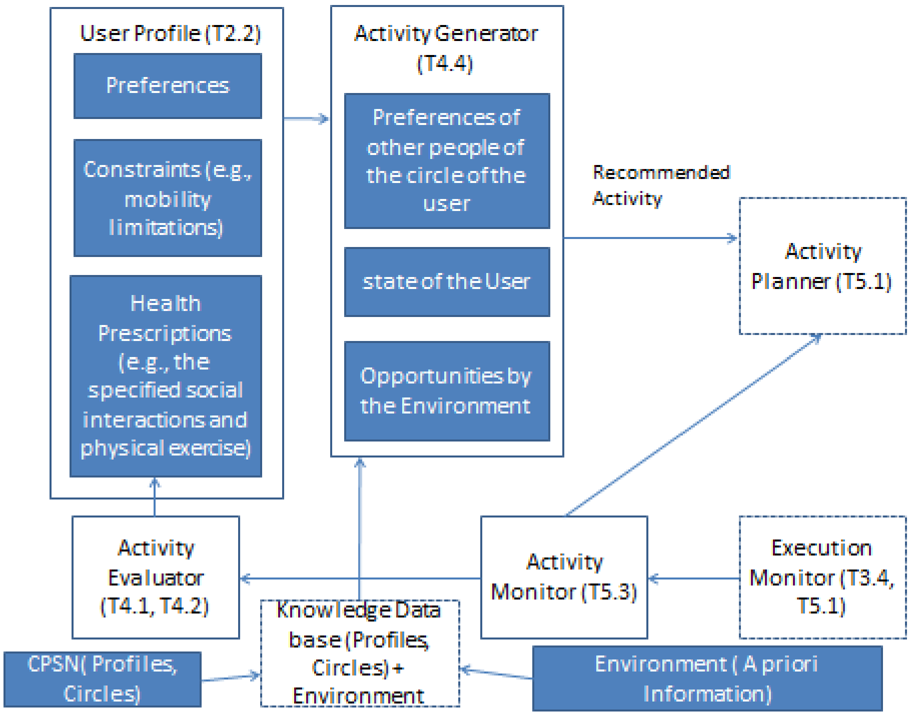

User Profile (T2.2)
Preferences
Constraints (e.g., mobility limitations)
Health Prescriptions (e.g., the specified social interactions and physical exercise)
Activity Generator (T4.4)
Preferences of other people of the circle of the user
state of the User
Opportunities by the Environment
Recommended Activity
Activity Planner (T5.1)
Activity Evaluator (T4.1, T4.2)
Activity Monitor (T5.3)
Execution Monitor (T3.4, T5.1)
Knowledge Data base (Profiles, Circles) + Environment
CPSN (Profiles, Circles)
Environment (A priori Information)

Figure 10 - Conception of Social Activities subsystem

The initial Activity attributes are provided by the user and integrated by caregivers and relatives. They will be continuously and automatically updated using the harvested information from the *Activity Evaluator* (to be developed in the task 4.1 and 4.2), which will be due at the end of an activity. It is used to measure the therapeutic efficacy of the activity, the user’s satisfaction, her/his physical and emotional state observed during the different phases and the quality of social interactions inside the circle (if the activity involves more than one user). This information is used to refine the profile and evaluate the possibility of the creation of new circles. Once an activity is matched with the users' profile and becomes a recommendation, it will be added with extra information related with the user. The *Activity Planner* generates an executable plan for the activity (see task 5.1) [6].

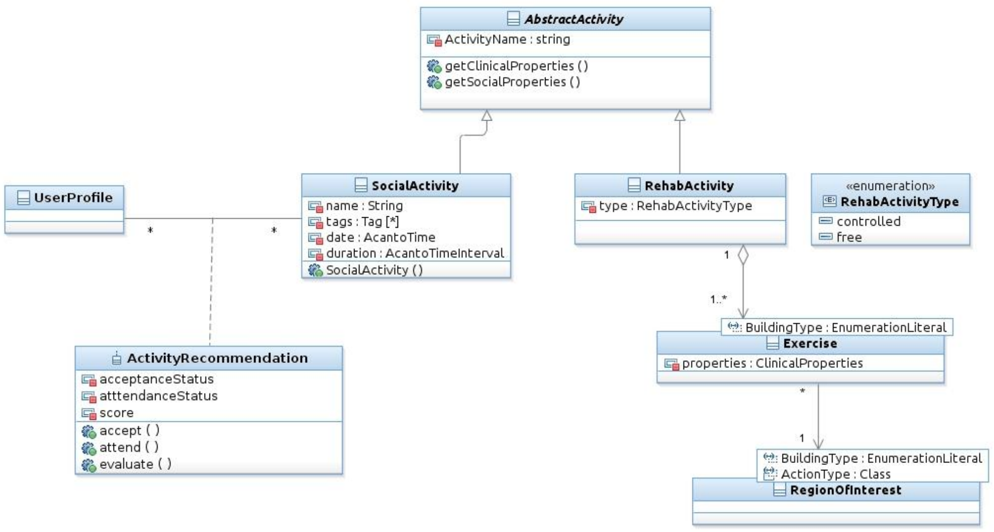


Figure 11: Activity class diagram

5.1.1 Rehabilitation Activities

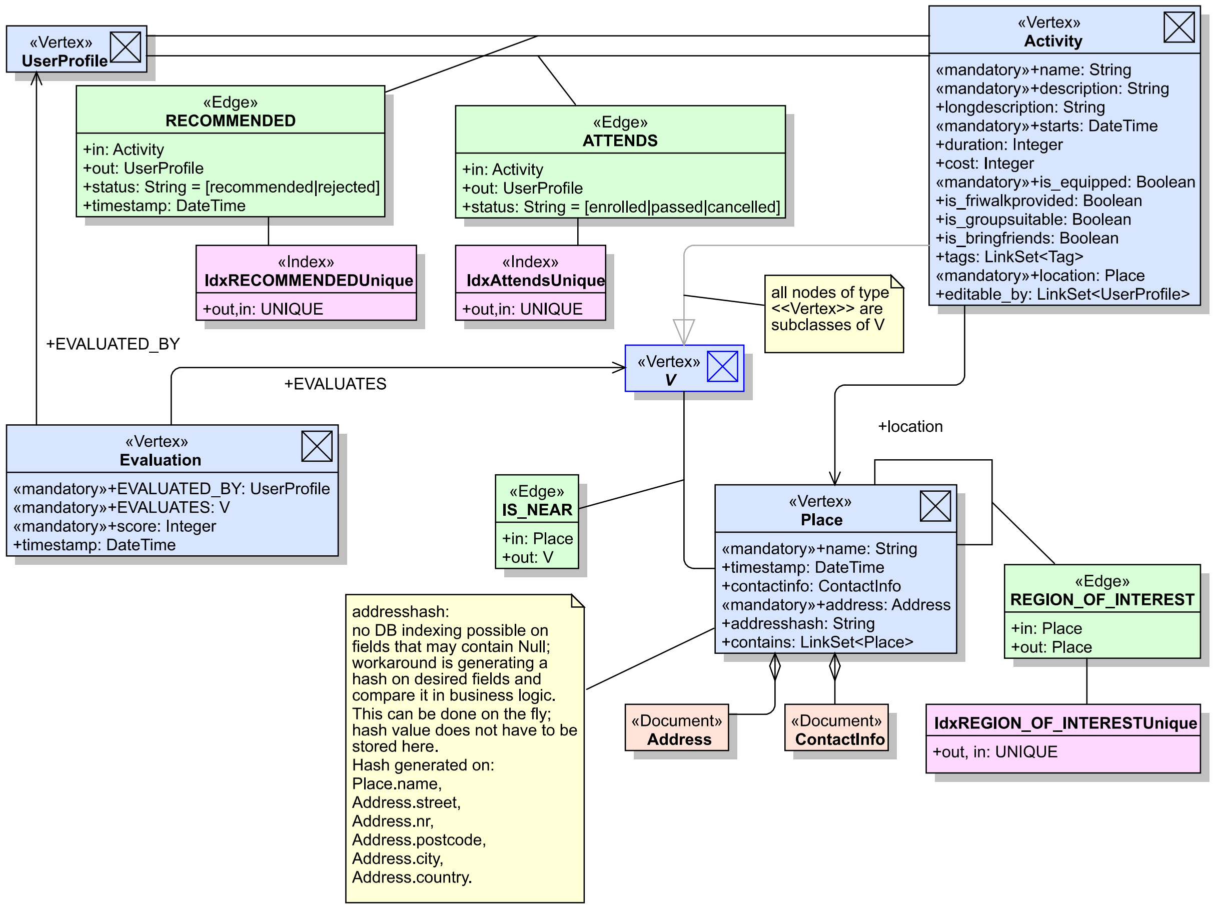
A **RehabActivity** can be of two types: **controlled** and **free**. Controlled activities are executed with the direct presence of a therapist, while free activities are executed autonomously by the user following the prescription and the indication of the therapist. Whatever the type, an activity is a sequence of exercises, each with its own critical properties. Exercises are carried out in a region of interest. Exercises are parametric types, where the parameter is given by the type of the building (e.g., walking exercises can be associated with "hospital" or "nursing house"). The region of interest where the activity is supposed to take place has to be of the same type. This guarantees a correct–by–construction definition and binding of regions of interest with exercises, relieving the tools from any type of run–time checks. Exercises have some clinical properties. The call to getClinicalProperties on the activity aggregates the properties of all the exercises composing the activity.

Example: Suppose the case of Dorothy (Use Case 3) undertakes a prevention programme for fall avoidance. She goes to the local hospital and she is administered a full diagnostic test in order to decide the appropriate treatment. The full diagnostic test is an instance of RehabActivity composed of three exercises: "stand–up Exercise" (class StandUpExercise), "controlled walk" (class ControlledWalkExercise) and "free walk" (class FreeWalkExercise). Each exercise has its ClinicalProperties hence, the properties of the test will be the union of the properties of each exercise.
Consider now, as an example, the StandUpExercise. It will specialise the template parameter BuildType to Hospital and it will be linked to a RegionOfInteres defining the area where the test is usually administered. The latter will have in its turn BuildType

specialised to Hospital and the ActionType parameter specialised to Exercise<Hospital>.

5.1.2 Social Activities

This section reflects the updates to the models previously developed for the deliverable D2.3 [1] and that are related to the Social Activity models.



all nodes of type <<Vertex>> are subclasses of V

addresshash:
no DB indexing possible on fields that may contain Null; workaround is generating a hash on desired fields and compare it in business logic.
This can be done on the fly; hash value does not have to be stored here.
Hash generated on:
Place.name,
Address.street,
Address.nr,
Address.postcode,
Address.city,
Address.country.

Figure 12: Social Activity schema

Figure 12: Social Activity schema depicts the structure of an ***Activity*** node in the *KnowledgeBase* and its relations. Every vertex of type *Activity* has a series of properties to describe the nature of a social event, and especially if it is suitable for group visits, or can be used with the *FriWalk* and whether the *FriWalk*s are available on site. The location of an *Activity* points to a ***Place*** node, which holds information about address, name of the place and its geolocation (latitude and longitude) as optional data.
The relationships between Activities and *UserProfile*s are described using edges of type **ATTENDS** and **RECOMMENDED**. The latter of which is being created by the recommender system. Once a user interacts with a recommended Activity an ATTENDS edge is created, in case he enrolls to attend the activity, or the RECOMMENDED edge's property *status* is set to 'rejected' if the user chose to reject the proposed activity.
EVALUATES edges are created once the related activity has passed and the user has confirmed its satisfaction about the event giving a ***score*** from one to five stars, being one star the least and five stars the most satisfactory experience. These edges will be used by the recommender system to update the recommendations.
The **IS_NEAR** edges are available to mark the proximity of another object; the type of

the pointed to (out) vertex is the generic class V and allows therefore to connect all subtypes of V. The only node types of interest are Place (say a Toilet or Bus stop), *UserProfile* (another person) and maybe *Circle* to indicate the proximity of a group of people. The SocialActivity schema is the general description of the data representation and the relationships of activities.

5.2 Activity Plans

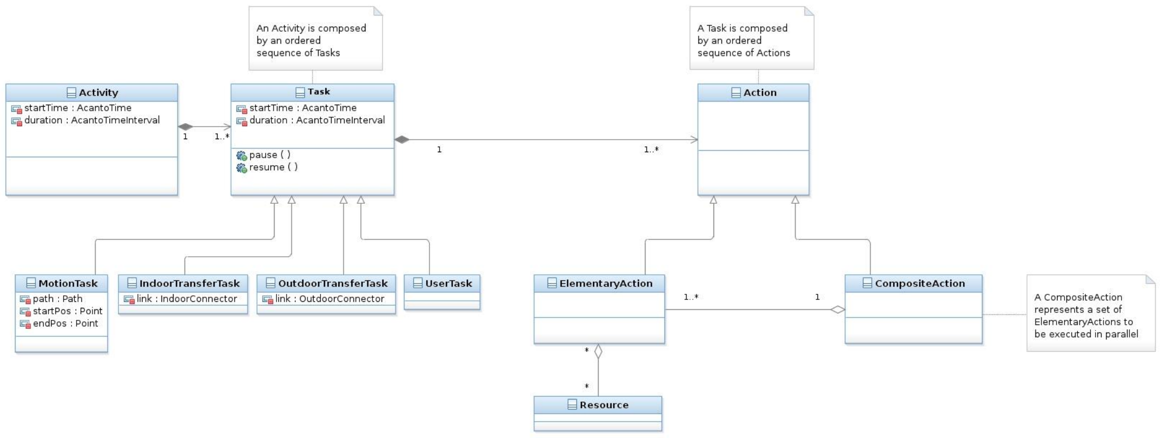


Figure 13: Activity Plan Class Diagram

The activity planning is covered in other deliverables (e.g., D2.5, D5.3), where appropriate languages are introduced for expressing plans, and tools are discussed that could proficiently generate them. In this context, we are merely interested to the interactions and relations between the Activity Planner and the other components of the system, its output and supporting data structures.
The aim of the Activity Planner is to refine an activity suggested by the Activity Generator, and to determine and fill in all the missing parameters to render it "executable".
Specifically, each activity is expanded into an ordered sequence of Tasks. The Activity Planner is responsible to determine for each Activity a sequence of Tasks implementing it, considering the preferences, prescriptions and requirements of the user and the information regarding the environment.

Specifically, a Task can wrap different kind of elements:

- A *UserTask*: a task that is performed by the user (e.g. buying some food at the grocery);
- A *MotionTask*: a motion between two *RegionOfInterest* on the same floor, which happens when walking between the target and the source location, which can be itself an activity with some clinical relevance;
- An *IndoorTransferTask*: a transfer between two Floors (using an *IndoorConnector*);
- An *OutdoorTransferTask*: a transfer between two *LivingSpaces* (using an *OutdoorConnector*).

For each *Task*, a starting time and a duration are determined, based on the overall duration of the *Activity* and data regarding the *Environment*. Some slight deviations from these timings can be tolerated, but in the case of large deviations, corrective actions may be required, and the plan may be updated.

Each Task is composed by an ordered sequence of one or more *Actions*. An Action is a wrapper for two kinds of items:

- An *ElementaryAction*, that can be associated with some *Resources* required to perform it (e.g. some hardware components);
- A *CompositeAction*, representing a set of *ElementaryActions* to be executed in parallel.

While the sequence of *Tasks* composing an *Activity* is chosen dynamically by the Planner, the sequence of *Actions* composing each *Task* is predetermined, and depends only on the type of the *Task*. At runtime, during the generation and execution of the plan, the dynamic parameters associated with the different actions are set (e.g. the path to follow for an *Action* representing a *Motion* of the walker).

5.3 Generation Monitoring and Evaluation of Activities

In this section we will present the sequence diagrams for the generation, monitoring and evaluation of activities.

5.3.1 Activity generation as it happens

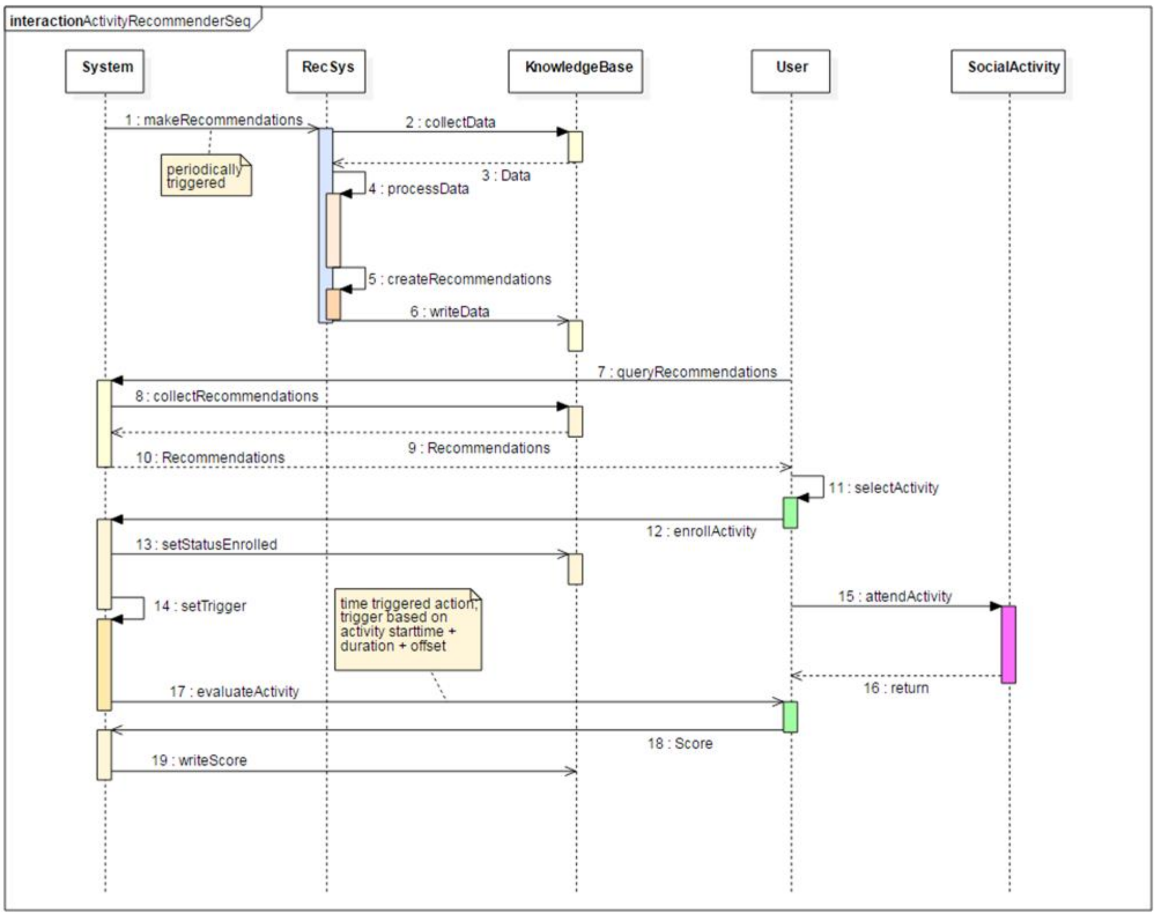


Figure 14: Activity recommender sequence diagram

The figure above exemplifies the activity recommender sequence diagram. The sequence diagram is used primarily to show the interactions between objects in sequential order. The *System* periodically triggers a batch process to generate recommendations. The recommender system (*RecSys*) collects all necessary data from the *KnowlegdeBase*, which it processes, creates recommendations from it and writes the data back to the *KnowledgeBase*. The *User* requests recommendations from the *System*. The *System* will query the *KnowledgeBase* and pass the collected recommendations to the *User*. The *User* chooses a recommended activity and the *System* sets the Activity status to “enrolled” in the *KnowledgeBase*. Once the activity has completed, i.e. *start time* + *duration* + some offset have passed, the *User* is prompted to attribute a score to evaluate his experience, which is then also written to the *KnowledgeBase*.

5.3.2 Activity Planning and Execution

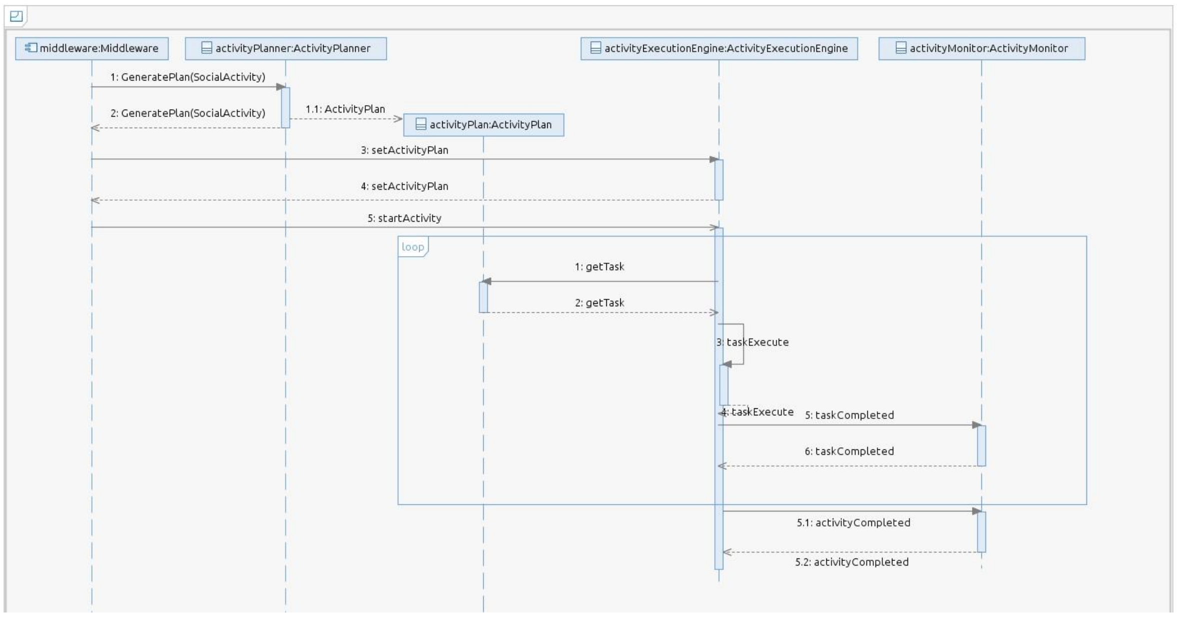


Figure 15: Sequence chart for activity planning and execution

The sequence diagram above illustrates the flow of messages when an Activity gets accepted and needs to be executed. The different components involved in the process are:

- The *System* middleware: responsible for the invocation of the different components involved at the different stages of planning and execution;
- The Activity Planner: responsible for the generation of a plan composed of a sequence of *Tasks* implementing the chosen *Activity*;
- The *ActivityExecutionEngine*: responsible for the actual execution of the *Actions* composing the *Activity*;
- The *ActivityMonitor*: component that monitors the progress of an *Activity*, the deviations from the original plan and the physical and emotional state of the participant during the execution of each *Task*.

The first step is the invocation of the Activity Planner to refine the suggested *Activity* and render it executable. Based on the information regarding the *UserProfiles* and the *Environment*, the most appropriate sequence of *Tasks* implementing the *Activity* is chosen, and filled with all the required data (e.g. the timings for each *Task*). Once the plan has been generated, it is passed to the *ActivityExecutionEngine* that is responsible for the execution of each of the *Tasks*. After the completion of each *Task*, all the data related to its execution and the physical and psychological state of the user are sent to the Activity Monitor. After all the *Tasks* have been completed, the level of satisfaction is being collected from the user (see Figure 15: *Activity* recommender sequence diagram). When the evaluation stage is completed, the *KnowledgeBase* is updated with the new score.

5.3.3 Class diagrams for activity generation, monitoring and evaluation

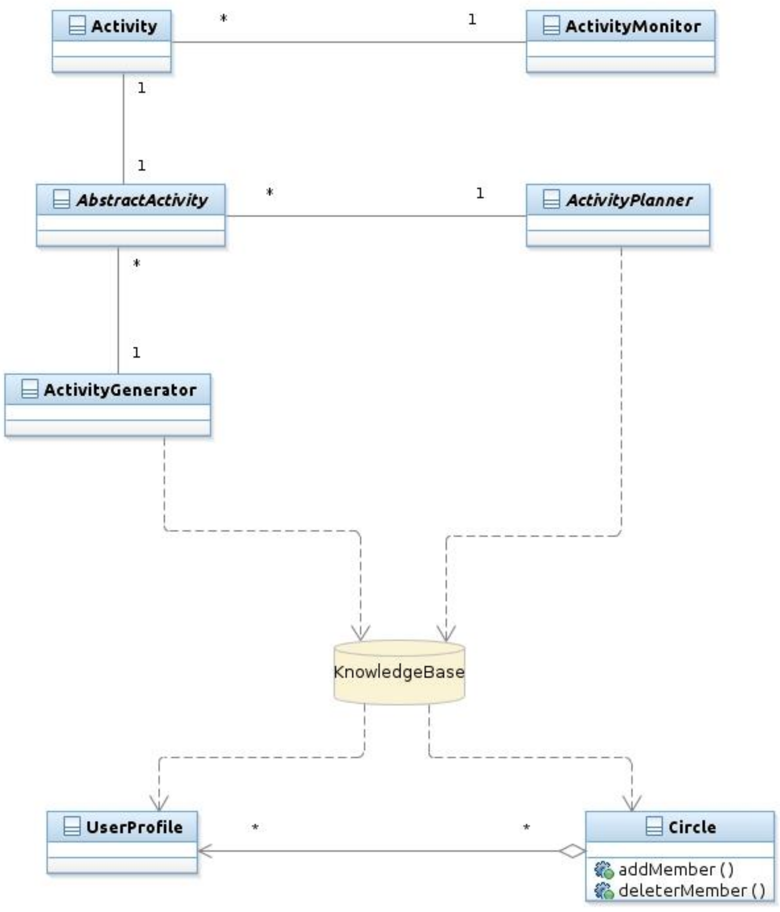


Figure 16: Class diagram for activity generation, monitoring and evaluation

The interaction described in the previous sequence diagrams suggests the diagram for classes and relation in Figure 13: Activity Plan Class Diagram. The **ActivityGenerator** is associated with the instances of **AbstractActivity** that it generates (so the relation is one–to–many). The **ActivityMonitor** needs visibility of both the Activity instances that it is required to monitor and the plan required for their execution. Finally all components need access to the KnowledgeBase to read and update the data.

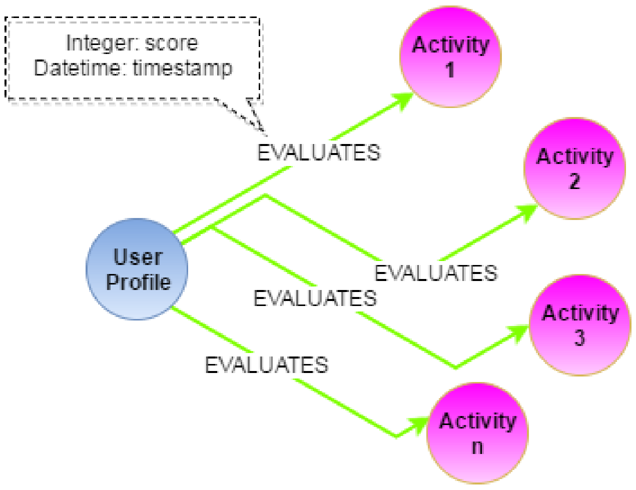


Figure 17: EVALUATES relation

Evaluation of social activities
The evaluation is the main element used by the recommender system, which uses the data to improve on the continuous task of recommending new *Activities* to the user. After the *Activity* is completed, the evaluation is collected from the user, e.g. by prompting with a dialogue from the User Interface (UI), and stored as EVALUATES edges between his/her *UserProfile* and the passed *Activity*. The edges carry the integer score and a timestamp as properties. If a user changes his/her mind about the given score, the *Activity* can be reevaluated and the updated score will be participating in the next cycle of recommendations.

Items and Activities for Evaluation
The *Tag* nodes are the categorization of different attributes that are part of the *UserProfile* or an *Activity*. *Tags* are used by the following properties: **preferences**, **dislikes**, **prescriptions** and **profession.categories** from the *UserProfile*, and tags property from *Activity*. The *Activity* nodes include a name, description and long description of the activity, date and time, duration, cost of admission, location, category tags and some booleans to indicate if the *FriWalk* are required and whether they are available on site. Also if a user can bring his friends and general if the activity is suitable for visits in groups. Once the activity is passed, an evaluation will be collected from the user.

The user profile is composed by the social profile, which contains their preferences and the mobility profile to inform about the requirements for the exercises to be assigned to the user and his/her constraints.

UserProfile and *Activity* vertices are connected by edges of type ATTENDS, which have a property *status* to reflect the state of the relation:
enrolled – when the user just signed up for an activity
passed – when the activity has passed and the user participated
cancelled – when the user made up her/his mind and chose not to go

See Figure 18 - Activity enrolment state diagram for the state diagram.

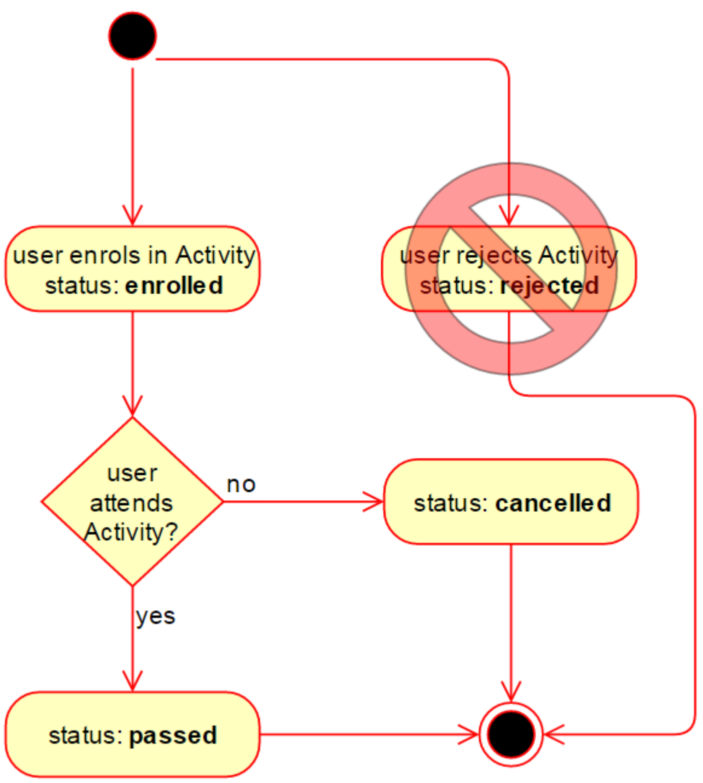


Figure 18 - Activity enrolment state diagram

The Tag class will be used to semantically connect the domains of the *UserProfile*'s preferences to the features of a social *Activity*: for example if an *Activity* is given a Tag '*outdoor*', then this would match against the same Tag describing a user's preference for outdoor activities. It is crucial for the recommender system that these tags come from a vocabulary common to both, *Activities* and *UserProfiles*. While the vocabulary for describing *Activity* features and *UserProfile* preferences is free (new *Tags* can be created when changing the UserProfile property '*preferences*' or when creating or modifying an *Activity)*, the vocabulary for *Tags* of domain ***mobility*** is restricted. The *mobility* domain is reserved for use with the ***mobility profile*** of *UserProfiles* and *Activities*. The mobility profile will help to join a medical practitioner's requirements (or "prescriptions") and *Activity* categories.

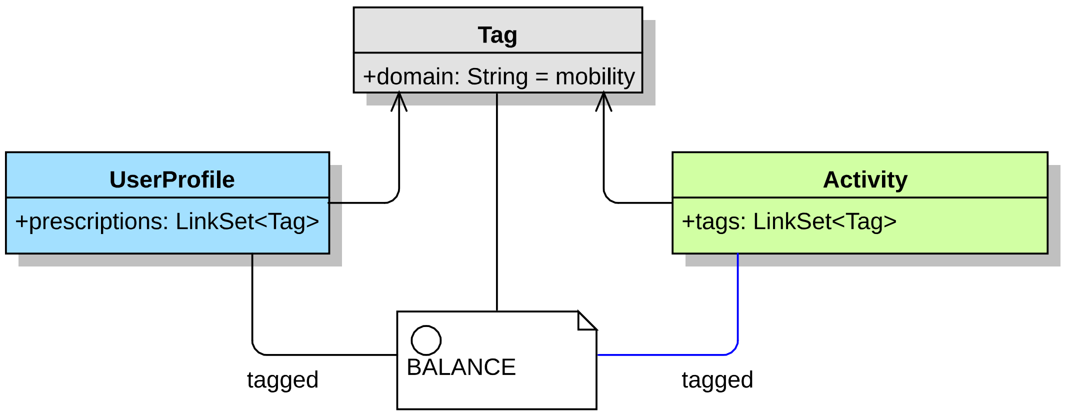


Figure 19: Mobility profile - tag - matching

These mobility *Tags* will be given two different descriptions, one to be used by the medical professionals and the other by the authors of social activities. An example can be seen in the following table.

label	activity description	prescription description
balance	involves balance activities	needs more balance activities
suitable visually impaired	suitable for visually impaired	poor eyesight
toilets available	toilets available nearby	needs to have toilets nearby
build muscle strength	helps to build muscle strength	needs to build muscle strength
suitable blind people	suitable for blind people	blind
wheelchair accessible	wheelchair accessible	wheelchair user
walk longer distances	involves lots of walking (more than 1km)	needs to walk longer distances (more than 1km)
suitable hard-of-hearing	suitable for those with bad hearing	poor hearing
FriWalker accessible	FriWalker accessible	FriWalker user
no walk longer distances	does not involve long distance walking (less than 1km of walking)	cannot walk long distances (must walk less than 1km)
suitable deaf people	suitable for deaf people	deaf

Table 2: Mobility Tag examples

Example: George, 47, is event manager for the local Mayor's office. To promote an activity conceived for the exercise of older adult's memory he enters into his CPSN account, where he is registered as *'Activity Creator'* and opens the page with the *Activity Creator portlet*. He quickly adds name, descriptions and time schedule for his new activity of 'improvisational comedy'. He then tags the activity with **'theatre'**, **'memory'**, **'improvisation'**, **'indoors'** and **'fun'** to enable the social categories. Also he adds mobility tags **'*FriWalk* accessible'**, **'suitable hard-of-hearing'**, **'toilets available'**, **'balance'** and **'no walk longer distances'** to the mobility features section. After selecting the Theatre from the list of known places he is ready to store the activity into the system.

6 Links to other work packages

The work of this document is related with WP4 and will serve as input and output language for the Activity Generator (T4.4), their attributes are initially provided by the user and constantly updated using the information from the activity evaluator implemented at T4.1 – *"User profiles harvesting".* The user profile model will also be the base for its evolution towards the inclusion of user's information coming from dynamic sources (i.e. sensing data coming from the cyber physical network based in the techniques developed in WP3, at task 3.2 – "Perception of the environment" as well as from the evolving social data coming from directly from the users' social network). It is also related with the WP5, namely the tasks 5.1 – *"Activity planning*", 5.2 – "Reactive planning" and 5.3 – "*Monitoring the execution of activities*".

7 Bibliography

[1] Addison-Wesley, Unified Modeling Language User Guide, The (2 ed.). A, ISBN 0321267974., 2005, p. 496.

[2] I. Ramos, L. Palopoli, D2.3 User, activity and environmental description (preliminary), 2016.

[3] OrientDB, «OrientDB,» 2017. [En línea]. Available: http://orientdb.com.

[4] OrientDB, «OrientDB REST,» 2017. [En línea]. Available: http://orientdb.com/docs/orientdb-rest.html.

[5] OrientDB, «OrientDB programming language bindings,» 2017. [En línea]. Available: http://orientdb.com/docs/last/programming-language-bindings.html..

[6] A. Consortium, ACANTO Project, 2015.