Funded by the 7th Framework Programme
of the European Union

Project Acronym: RAPP
Project Full Title: Robotic Applications for Delivering Smart User Empowering Applications
Call Identifier: FP7-ICT-2013-10
Grant Agreement: 610947
Funding Scheme: Collaborative Project
Project Duration: 36 months
Starting Date: 01/12/2013

D1.2 RAPP Ontology

Deliverable status: Final
File Name: RAPP_D1.2_V1.1_30012015.pdf
Due Date: November 30, 2014
Submission Date: November 30, 2014
Updated Version Date: January 30, 2015
Dissemination Level: Public
Task Leader: 1 – CERTH/ITI
Author: Emmanouil Tsardoulias

The RAPP project consortium is composed of:

CERTH	Centre for Research and Technology Hellas	Greece
INRIA	Institut National de Recherche en Informatique et en Automatique	France
WUT	Politechnika Warszawska	Poland
SO	Sigma Orionis SA	France
Ortelio	Ortelio LTD	United Kingdom
ORMYLIA	Idryma Ormylia	Greece
MATIA	Fundacion Instituto Gerontologico Matia - Ingema	Spain
AUTH	Aristotle University of Thessaloniki	Greece

Revision Control

VERSION	AUTHOR	DATE	STATUS
0.1	Emmanouil Tsardoulias (CERTH/ITI)	September 29, 2014	Initial Draft
0.5	Włodzimierz Kasprzak Cezary Zieliński	October, 12, 2014	Addition of an example of processing and communication procedure
1.0	Emmanouil Tsardoulias	October, 24, 2014	Incorporated the provided changes.
1.1	Emmanouil Tsardoulias	January, 20, 2015	Included visual upper taxonomy examples regarding the two ontologies

Project Abstract

The RAPP project will provide an open-source software platform to support the creation and delivery of Robotic Applications (RApps), which, in turn, are expected to increase the versatility and utility of robots. These applications will enable robots to provide physical assistance to people at risk of exclusion, especially the elderly, to function as a companion or to adopt the role of a friendly tutor for people who want to partake in the electronic feast but don't know where to start.

The RAPP partnership counts on seven partners in five European countries (Greece, France, United Kingdom, Spain and Poland), including research institutes, universities, industries and SMEs, all pioneers in the fields of Assistive Robotics, Machine Learning and Data Analysis, Motion Planning and Image Recognition, Software Development and Integration, and Excluded People. RAPP partners are committed to identify the best ways to train and adapt robots to serve and assist people with special needs.

To achieve these goals, over three years, the RAPP project will implement the following actions:

- Provide an infrastructure for developers of robotic applications, so they can easily build and include machine learning and personalization techniques to their applications.
- Create a repository, from which robots can download Robotic Applications (RApps) and upload useful monitoring information.
- Develop a methodology for knowledge representation and reasoning in robotics and automation, which will allow unambiguous knowledge transfer and reuse among groups of humans, robots, and other artificial systems.
- Create RApps based on adaptation to individuals and taking into account the special needs of elderly people, while respecting their autonomy and privacy.

- Validate this approach by deploying appropriate pilot cases to demonstrate the use of robots for health and motion monitoring, and for assisting technologically illiterate people or people with mild memory loss.

The RAPP project will help to enable and promote the adoption of small home robots and service robots as companions to our lives. RAPP partners are committed to identify the best ways to train and adapt robots to serve and assist people with special needs. Eventually, our aspired success will be to open and widen a new 'inclusion market' segment in Europe.

Table of Contents

List of Abbreviations

Abbreviation	Definition
AAL	Ambient Assisted Living
ICT	Information and Communication Technology
OWL	Web Ontology Language
RApp	RAPP Application
RDF	Resource Description Framework
ROS	Robot Operating System
SIFT	Scale-invariant feature transform
SURF	Speeded Up Robust Features

Executive summary

The present document is a deliverable of the RAPP project, funded by the European Commission's Directorate-General for Communications Networks, Content & Technology (DG CONNECT), under its 7th EU Framework Programme for Research and Technological Development (FP7).

As our societies are affected by a dramatic demographic change, in the near future elderly and people requiring support in their daily life will increase and caregivers will not be enough to assist and support them. Socially interactive robots can help to confront this situation not only by physically assisting people but also functioning as a companion. The increasing sales figures of robots are pointing that we are in front of a trend break for robotics. To lower the cost for developers and to increase their interest on developing robotic applications, the RAPP introduces the idea of robots as platforms. RAPP project will provide a software platform in order to support the creation and delivery of robotic applications (RApps) targeted to people at risk of exclusion, especially older people. In the context of the RAPP project, WP1 is responsible for describing the RAPP Platform Design, i.e. the basic upon which the RAPP Platform will be developed. This document is the second deliverable of WP1 (D1.2) and provides the description of the semantic robotic description schemes. The current document aims to provide the basis and specifications for the upcoming RAPP Platform development and utilization.

1. Introduction

The purpose of the first RAPP work package is to create the basis on which the entirety of the RAPP platform will be developed, i.e. to provide the tools and their specifications to be utilized during the platform's construction. These tools include the involved user requirements (T1.1), the specification of a language for representation and query (T1.2), RAPP's ontology design and development (T1.3), RAPP platform requirements and specifications (T1.4), RAPP key performance indicators (T1.5) and ethical legislations and guidelines (T1.6).

The current document is dedicated to the presentation of the adopted ontology schemes that will be utilized in the RAPP platform. One of RAPP's central aspects (and aspirations) is to contain a means of storing, utilizing and transferring information among different robotic devices. It is a fact that the most promising trend in the future robotics is the "collective robotic intelligence", i.e. creating ways for the robotic devices to drain knowledge from a central (or distributed) repository, as well as provide new concepts derived from their everyday functionality. Nowadays there are numerous projects that begun to investigate this problem, some of which are RoboEarth [1], RoboBrain[1] and DAvinCi [2] and as often referred as *Cloud Robotics*.

These systems are able to store and distribute knowledge, as well as derive and infer information from many sources. These may be:

- Stimuli observed by robots such as an object in a household, certain chain of motions performed by humans to achieve a task (i.e. that opening a door requires to grasp the handle and perform a downward arc movement) or even information provided by vocal means, when for example a person gives commands or information to the robot.
- The internet, which is a vast e-storage for structured and unstructured information. As far as structured information is concerned, many websites exist that provide information in a way that a computer (and hence a robot) can utilize, such as ImageNet, 3D Warehouse, YouTube videos, WikiHow[2], WordNet[3] and several other websites where categorized and tagged information exists. On the other hand unstructured information can be found literally anywhere, as they exist in any text written by humans like articles in Wikipedia, e-books, e-newspapers etc. A robot (or a computer) could in theory "study" these sources and infer more and more complex information as the NLP abilities enhance.
- Predefined knowledge databases which were manually encoded in order for the robots to be able to employ them. Some of them are the KnowRob ontology and the OMCS (Open Mind Common Sense project)[4].

So why we perform all this effort to provide the robots the means of learning the world instead of letting them collect that knowledge by themselves? The truth is that in terms of computation and information inference a human being is by far more advanced in comparison to a robot. This means that tasks which are trivial to humans need a lot of detail and effort for a robot to be able to perform them. An example could be the procedure of fetching a glass of orange juice. If this command is given to a human, he/she would go to the kitchen, open the fridge, find the orange juice bottle, serve some juice in a glass and bring the glass to the person who did the request.

The same procedure is extremely complex for a robot, as it does not have the knowledge and experience of a human being, thus many things are not assumed. First of all, the robot receives a vocal command that has to "translate" into its own "language" and understand that it needs to fetch a glass full or orange juice to the person that gave the command. Then it should search for the "glass" and "orange juice" concepts and find out that they exist in the "kitchen". Of course the robot needs a map in order to determine where the kitchen is, or to understand that a room is the kitchen by observing the things in it. When the robot reaches the kitchen, it should find a glass. Here indeterminacy is inserted, as

[1] http://robobrain.me/

[2] http://www.wikihow.com/

[3] http://wordnet.princeton.edu/

[4] http://web.media.mit.edu/~push/Kurzweil.html

the robot could have a model of a glass (image of 3D cad) but the specific house's glasses to be different in size, colour etc. Apart from this, the robot should have the information of where are the glasses usually stored (may be the drawers or another kitchen furniture). Let's assume that a glass exists on top of the table and is easy to reach. Now a manipulation problem is present. The robot must know that the glass is fragile (thus it should not apply too much pressure) and that glasses are usually grasped from their bottom part.

The second act is the orange juice serving. Again the robot should know that the orange juice exists in the fridge, which has a special handle to open and that handle must be grasped and pulled towards the robot. Then the information must exist that the orange juice is in a cardboard box, because it is liquid. Thus the cardboard box should be handled with care as it is a transformable object (not rigid) and in order to be opened a cup should be twisted in a rotational movement. When this is achieved the juice should be poured in the glass. Now the glass should be kept in the correct position relatively to the juice box and the box to have a specific angle to the ground in order for the juice to flow. Of course we assume that there is actually juice in the box, and is enough to fill the glass. During the filling, the robot must execute image processing algorithms to understand when the glass is full in order to return the cardboard box in a vertical position, or else the juice will be spilled to the floor. Then, after the actions of sealing the juice box, returning it to the fridge and closing the fridge, the robot must bring the glass to the human. During its space traversal it must know that in no case must it change the orientation of the glass, regardless of the external stimuli (other vocal commands or dynamic objects in the house).

It is apparent that robots need a large amount of details in order to be able to perform even minor tasks and that is the reason behind the latest boost in the construction of "collective robotic intelligence" or "cloud robotics" projects.

The current document is structured as follows:

- **Chapter 2: State of the art**. Here the most important ontology schemes regarding robotics and social concepts are described.
- **Chapter 3: Ontology design**. The description of the actual ontologies we are going to use in the RAPP project will be presented, their main classes and hierarchy, as well as examples of their properties.
- **Chapter 4**: **RAPP Ontology**. Here, some technical details on the ontology setup, access and utilization will exist.

Next, the state of the art is presented.

2. State of the Art

The current chapter's aim is to provide the basic information about the available ontologies and knowledge repositories that exist and are utilized by robots or computers in general. Initially, some general purpose ontologies will be described, then the robot and AAL oriented ontologies and finally a few online general information sources / repositories will be presented.

2.1. General purpose ontologies

The first project to describe is called **Cyc**, a name that derived from the word "encyclopedia". This project initiated in 1984 as part of Microelectronics and Computer Technology Corporation[5] and its ultimate goal was to create a repository of human common sense knowledge in a form that could be utilized by machines. In 1994 the Cyc project was transferred to Cycorp, Inc[6]. Originally, the knowledge database was proprietary but then **OpenCyc**[7] was released under Apatche licence (and currently under OpenCyc licence). Of course OpenCyc did not contain the entirety of information that existed in Cyc, but a small basic portion of it. OpenCyc's domain is all of human consensus reality and the current release (v4.0) comprises almost 250.000 terms and over 2.000.000 triples. Lately, the whole Cyc relational database was released in a special distribution called **ResearchCyc**, under the *ResearchCyc* licence, making available to the scientific community the entirety of the gathered knowledge[8]. Furthermore, a subset of OpenCyc is **UMBEL** (Upper Mapping and Binding Exchange Layer)[9] that contains over 28.000 concepts and provides extra tools for semantic search and query.

Another general purpose ontology is **SUMO**[10], whose name derives from Suggested Upper Merged Ontology. SUMO, as stated in its website "forms the largest formal public ontology in existence today". Its main characteristics is that it is the only ontology that has been mapped to the WordNet lexicon (will be described in chapter 2.4), is free and owned by IEEE. Furthermore, the ontologies that extend SUMO are also open source under the GNU GPL licence. Its size is about 25.000 terms and 80.000 axioms and contains many other ontologies (except from SUMO) such as MILO (Mid-Level Ontology) and ontologies for a wide range of heterogeneous aspects.

Finally, **YAGO** [11] is a large semantic knowledge base which combines concepts from Wikipedia, WordNet and GeoNames (will be described later). Its current version is *YAGO2s* and contains more than 10.000.000 entities and over 120 million facts about these entities. YAGO is free to use and can be accessed by a web interface, by a SPARQL[12] interface or by downloading it and loading it offline into an RDF[13] triple tool.

2.2. Robotic ontologies

As described in the introduction, most of the knowledge that exists in the web is not directly applicable b a robotic device, as several spatiotemporal and conceptual details are missing. This is the reason behind several efforts on creating an ontology that a robot can actually utilize, without the need of manually breaking down certain tasks by the user.

[5] http://en.wikipedia.org/wiki/Cyc
[6] http://www.cyc.com/
[7] http://www.cyc.com/platform/opencyc
[8] http://www.cyc.com/platform/researchcyc
[9] http://umbel.org/
[10] http://www.ontologyportal.org/
[11] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
[12] http://www.w3.org/TR/sparql11-overview/
[13] http://www.w3.org/RDF/

One of the most important projects that aimed towards this goal is **RoboEarth**[14] and the robotic knowledge that contains, **KnowRob**[15]. KnowRob is described as "Knowledge processing for robots" and combines heterogeneous knowledge sources for enriching its database, as well as numerous tools to process and manipulate this knowledge, such as deterministic and probabilistic reasoning mechanisms, clustering, classification and segmentation algorithms, as well as query interfaces and visualization tools.

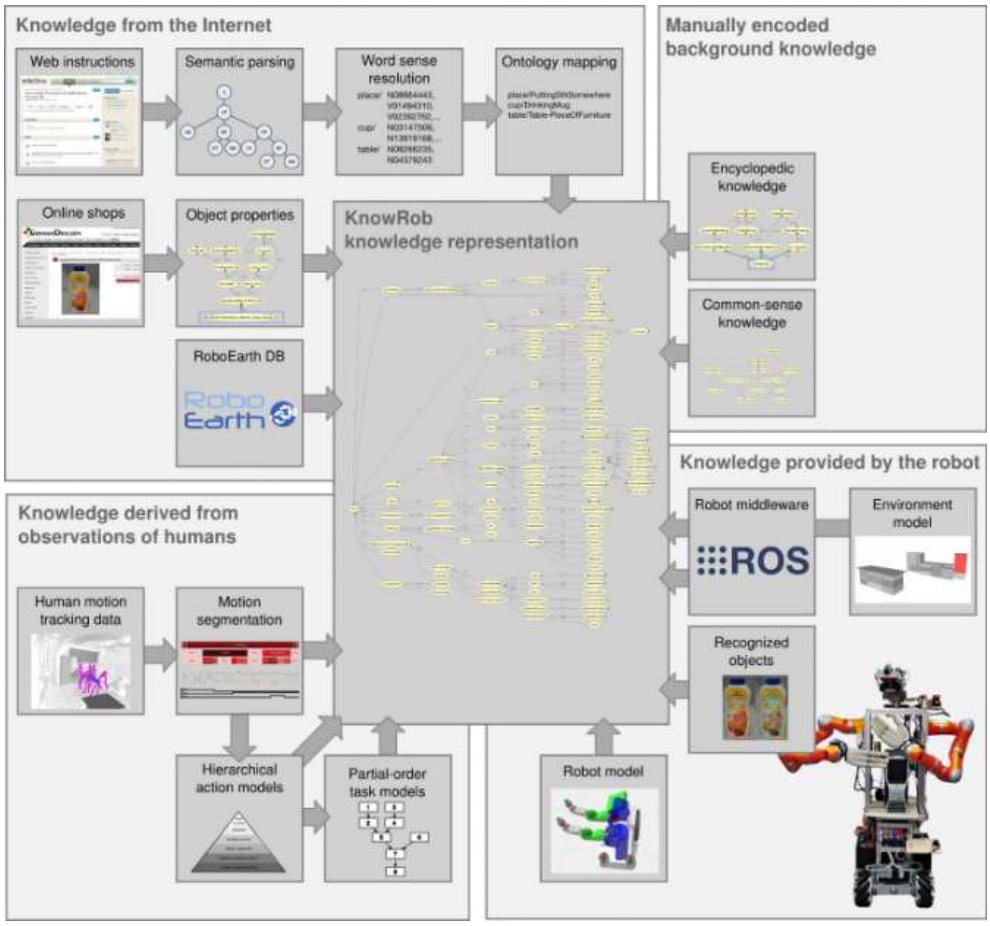


Figure 1: KnowRob information sources

As visible in Figure 1, KnowRob extracts knowledge from four distinct sources [3]:

- The upper right part is a static, manually inserted knowledge database, which contains basic encyclopaedic knowledge about the types of things and common-sense knowledge about the household. This is considered the fundamental basis for any robot to operate.
- The upper left part provides the knowledge extracted from various internet sources. As mentioned, a lot of these sources are designed for the human understanding, thus the robot has to acquire this information and process it, in order to create a machine-friendly version of it.
- Another source of information derives from robot observation of humans. It is obvious that humans are much more dexterous in everyday tasks, from pressing buttons to unscrewing bottle caps, as robots need both complex perception systems and spatiotemporal object information to be able to perform such motions. This fact make the knowledge derived from observation of humans an interesting concept, as a robot can learn how a task is performed by experience transference.

[14] http://roboearth.org/

[15] http://www.knowrob.org/

- Finally, the robot creates knowledge as it interacts with its surrounding environment. When a new piece of knowledge becomes available to the robot, it can contribute and upload it to the RoboEarth platform, in order to be able to circulate among other robotic devices.

All knowledge is represented in OWL (Web Ontology Language)[16] and KnowRob is implemented in SWI Prolog[17]. Finally, KnowRob is available as ROS packages[18], fact that is convenient to our case, as RAPP will utilize ROS both in the robots and in RAPP cloud.

Another robots-oriented ontology is **ORO** – The OpenRobots Common Sense Ontology[19] [4]. This ontology is mainly based on the OpenCyc ontology and shares most of its concepts with the KnowRob ontology. As mentioned in the ORO dedicated website, this project focuses on the creation of a framework for robots to share a common representation of concepts of the world. Some of ORO's capabilities are the employment of queries based on SPARQL, classification and first order logic based on the *Open World Assumption,* using the Pellet library[20], event system and assignment of memory profiles to statements (for example the robot is able to forget some facts after some amount of time).

Finally, the **Proteus** project ("Plateforme pour la Robotique Organisant les Transferts Entre Utilisateurs et Scientifiques" or Robotic Platform to facilitate transfer between Industries and academics) has as goal to bridge the French robotic community to the industry, in order to enable transfer of knowledge between the two different fields. The created ontologies were oriented towards fully defining operational and functional scenarios for humanoid and air / land robotics. Some of the aspects described were the mission description, the functional goals, the robots, the robots associated objects, the technical robots capabilities to perform tasks, their constraints, the environment description, as well as the dynamic aspects of this environment (events, triggers etc). The resulting ontology is available for downloading in OWL / RDF files.

2.3. AAL / ICT ontologies

The need for organizing the knowledge in a way that is computer comprehensible is apparent in almost every scientific field. The same applies for the social / assisted living paradigms. Their aim is to formally specify the **context** of things and actions found in AAL, in order to make robots and machines context-aware in order to sense and react based on their environment. In [5] context may be characterized as:

- **Static or dynamic:** Static context describes unaltered facts, such as a person's birthdate. A dynamic paradigm may be a person's age, as in increases with time.
- **Imperfect:** The information stored may be altered in short notice or the context sources may provide erroneous data.
- **Alternative representation:** This describes the ambiguity between multiple representations for a single concept.
- **Highly interrelated:** The contexts of entities have strict (or relaxed) relations between them, something that holds information as well.

Next, some examples of ALL/ICT ontologies are briefly described.

[16] http://en.wikipedia.org/wiki/Web_Ontology_Language
[17] http://www.swi-prolog.org/
[18] http://www.ros.org/
[19] http://www.openrobots.org/wiki/oro-server
[20] http://clarkparsia.com/pellet

The first case is **SOPRANO (Service-Oriented Programmable Smart Environments for Older Europeans)** [6]. SOPRANO is not just an ontology, but a combination of ontology-based techniques and a service oriented architecture. SOPRANO comprises three main parts:

- The SOPRANO ontology, whose lower part contains the means to semantically describe services and the operational state of the supported devices and the higher part contains the entities that help to semantically describe the environment.
- The service-oriented infrastructure, which implements the different component inter-communication, based on the OSGi-framework[21].
- The SOPRANO ambient middleware, which provides functionalities such as semantic service matchmaking, access and manipulation of ontology data etc.

Another Ambient Assisted Living project is **SINDI** [7], whose name derives from the words Secure and Independent Living. This is an intelligent home healthcare system that contains semantic, inference and reasoning mechanisms, in conjunction to a wireless sensor network. Its main functionality contains the constant acquisition of new semantic data. After their categorization and storage in the system, these are utilized to detect fluctuations in the social and mental status of a patient, to predict possible hazardous situations for his / her health, as well as predict cases and actions that may worsen the patient's mental or physical health. Technically speaking, data from different sources are gathered, where reasoning applies and tries to interpret inconsistent, erroneous or simply incomplete information by correlation rules provided by clinicians.

VAALID (Accessibility and Usability Validation Framework for AAL Interaction Design process)[22] is an FP7 - ICT, 2007.7.2 project, which provides a set of ontologies aiming at modelling an AAL environment, as well as its contents (actors, spaces, devices). Though, no reasoning techniques that will help to extract or infer knowledge are described..

ELDeR [8] is an ontology, which aims in facilitating the procedure of creating ambient intelligent systems (AmI) and contains context information relative to the elder's daily activities. Additionally, its inspiration is to enable developers to create systems which can infer potential future risks.

Finally, **OpenAAL**[23] [9] (the open-source semantic middleware for ambient assisted living) is an open source software based on the SAM (SOPRANO Ambient Middleware) architecture and implementation and is oriented towards AAL scenarios. Its main characteristics is that it provides a powerful user context management system, allows for context augmentation (abstraction and inference from low level information, i.e. measurements from sensors), comprises rule-based approaches, Bayesian networks and others.

2.4. General information sources

Finally, apart from pure ontology-based systems there is a number of other forms of collected information. Some indicative ones follow:

- **Mindpixel**[24] was a web-based artificial intelligence project, whose target was to create a large knowledge database, based on manually answered true or false statements (or probabilistic sentences) from humans. This project was discontinued in 2005.
- **DBpedia**[25] aims at making the Wikipedia contents semantically available, i.e. to extract structured information from Wikipedia pages. As stated in its official website, the current DBpedia size is 4.58 million things, out of

[21] http://www.osgi.org/Main/HomePage

[22] http://www.vaalid-project.org/project-details.html

[23] http://openaal.org/

[24] http://en.wikipedia.org/wiki/Mindpixel

which 4.22 million things are stored in an ontology. The strength of DBpedia is that it has a huge source of information and is automatically updated, as Wikipedia pages update. Additionally, it is multilingual and allows for complex quering.

- **Freebase**[26] provides access to the Google's Knowledge Graph either by an online API or by downloading the complete graph as an RDF file.
- **BabelNet**[27] is dedicated in the lexicographic science, as it is described as "A very large multilingual encyclopaedic dictionary and semantic network". It contains terminology from over 50 languages and a semantic network that connects the entities with relations.
- **GeoNames**[28] is a geography-dedicated database, which contains over 10 million geographical names, along with information such as longitude and latitude, elevation, population etc.
- **Never-ending Language Learning**[29] is a system that tries to "learn how to read the web". Initially it tries to extract facts from webpages and secondly it tries to improve its learning capabilities.

Next, the RAPP ontology design selections are going to be presented and some analytical examples of their basic entities will be described.

[25] http://dbpedia.org/About

[26] https://www.freebase.com/

[27] http://babelnet.org/

[28] http://en.wikipedia.org/wiki/GeoNames

[29] http://rtw.ml.cmu.edu/rtw/

3. Ontology design

Since RAPP is a multidisciplinary project that contains elements both from robotics and assisted living, it is crucial to employ ontologies that comprise semantical information from these two fields. Some of the prerequisites we had in mind before actually selecting the ontologies were:

- The tools should have ROS support, as the entire RAPP platform will be ROS-based. This will boost the modules' interconnection capabilities, as every node will have seamless access to the knowledge database.
- Both ontologies should have a common file format for the information to be easily joined.
- The ontologies should be state of-the-art, in order to fully take advantage of their capabilities and to make sure that they will have support from the community.

3.1. KnowRob

Taking these prerequisites under consideration, we selected KnowRob as the robotic ontology we are going to utilize. As stated before, KnowRob is part of the RoboEarth FP7 project and was specially designed in order to be used by robots. Specifically, it extends classical ontologies by containing information vital for a robot to operate, such as spatial / mathematical data (coordinates, matrixes, vectors etc), as well as temporal (events, timeslots and others).

The main reason that KnowRob was selected was that it is considered the state-of-the-art in robotic ontologies. Additionally, it is a software product of the RoboEarth project which is in many parts similar to RAPP, like cloud robotics, knowledge storage and distribution and knowledge extraction from different sources. We made this choice having in mind that utilizing already existent and tested tools is the appropriate thing to do, as it boosts performance and promotes collaboration between diverse scientific groups. Furthermore, an obvious advantage in utilizing KnowRob is that is compliant with ROS[30], as any developer is able to query the ontology by a ROS node. This is very helpful in keeping simple the overall system architecture we are proposing.

As far as support is concerned, KnowRob is utilized by many scientific teams and its creator is currently updating this software, making it employable even in the newest Ubuntu and ROS versions (14.04 LTS and Indigo respectively). It must be mentioned that in KnowRob all knowledge is implemented in OWL (Web Ontology Language)[31]. OWL files are XML-like and allow to formally describe relational knowledge, i.e. concepts that semantically interconnected. Also, as stated earlier, KnowRob is implemented in SWI-Prolog. Prolog is used to load store and reason on the knowledge contained in the OWL files. The ROS wrapper was created by utilizing the JPL interface (Java / Prolog bidirectional Interface)[32] and the experimental (up until now) *rosjava*[33] package. For a more detailed KnowRob documentation, many examples / tutorials can be found in the official KnowRob website[34].

[30] http://wiki.ros.org/knowrob

[31] http://www.w3.org/TR/2004/REC-owl-guide-20040210/

[32] http://www.swi-prolog.org/packages/jpl/

[33] http://wiki.ros.org/rosjava

[34] http://www.knowrob.org/doc

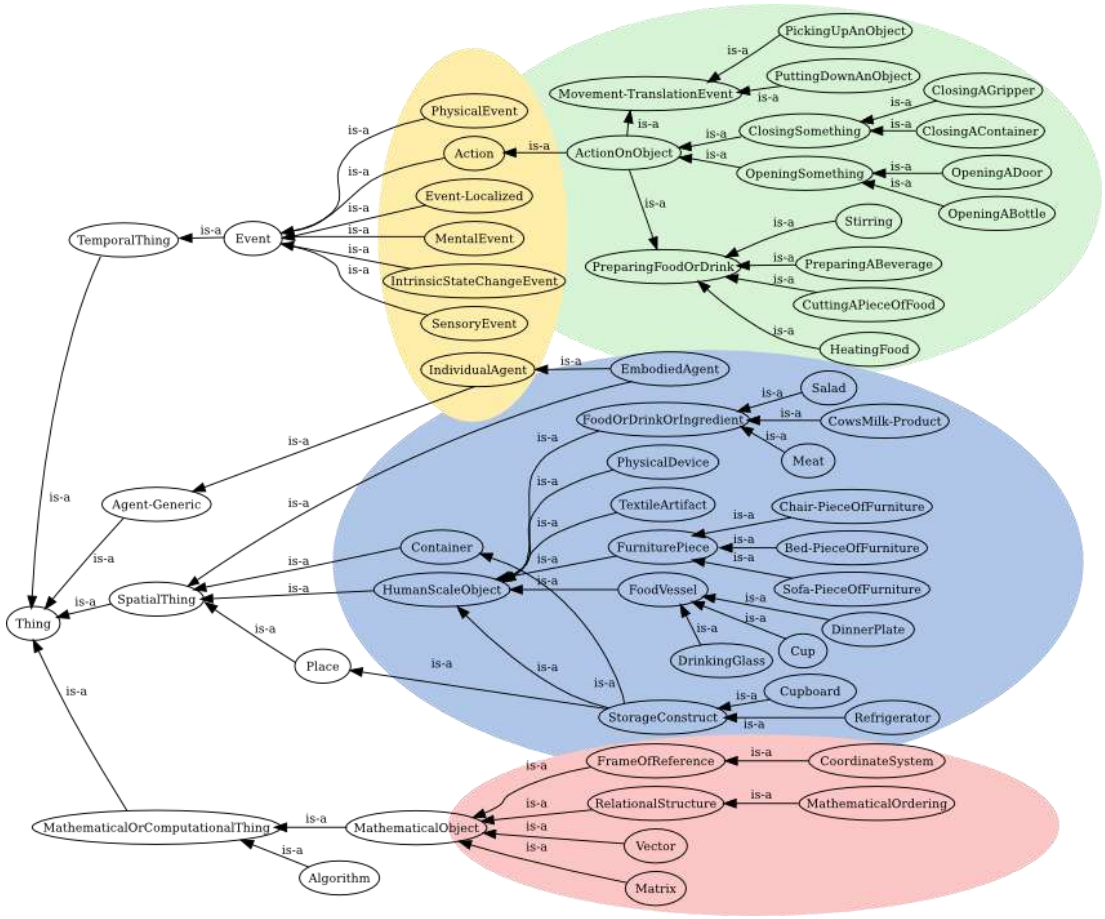


Figure 2: Upper KnowRob taxonomy

An image that presents the upper KnowRob ontology and taxonomy is Figure 2. There, the following concepts can be found:

- **Thing:** The most basic entity of the ontology. It is the most abstract concept and all other classes derive from it, i.e. everything is a “thing”. Its four subclasses are:
  - **TemporalThing:** Describes all elements that have to do with time. Its subclasses are
    - **TimeInterval:** (for example time points, dates, time of the day)
    - **Situation:** For example a posture or a grasp
    - **Event:** A general concept of events. This may contain:
      - **SensoryEvent** (e.g. perception of an object by camera or distance sensors)
      - **StateChangeEvent** (e.g. heating, freezing, vaporization)
      - **Action:** Elements that have to do with time and performing an action. Some examples are:
        - **Perceiving** (like the SensoryEvent but in the action context)
        - **VoluntaryBodyMovement** (Reaching / Releasing / Grasping)
        - **ActionOnObject** like
          - **ControllingSomething** (electrical devices for example)
          - **HoldingAnObject** (no movement, grasping involved)
          - **RemovingSomething**
          - **OpeningSomething**
          - **ClosingSomething**
  - **Agent-Generic**: Used to describe robots as agents.

- **SpatialThing**: Describes things that exist physically in an environment (or space)
  - **Map** of an environment
  - **Point**
  - **Place:** A relevant place in the environment (e.g. kitchen)
  - **Trajectory** of an object motion (or a robot motion)
  - **SurfaceRegion**: Different sides of objects (right side, front, back)
  - **EnduringThing-Localized:** All objects that can be assigned a location, These can be:
    - **EmbodiedAgent**
    - **PhysicalDevice:** e.g. tools
    - **Connection-Physical**: Any joint
    - **HumanScaleObject**: Objects related to humans. These may be:
      - **ConstructionArtifact**: Walls, stairs, doors, windows etc.
      - **FurniturePiece**: All furniture (tables, chairs etc.)
      - **AnimalBodyPart**: Arms, legs, torso, fingers etc.
      - **FoodOrDrink**: Edible and drinkable stuff
- **MathematicalOrComputationalThing**: Mathematical concepts
  - **Units**
  - **MathematicalObjects** e.g. Vectors, Matrices etc.

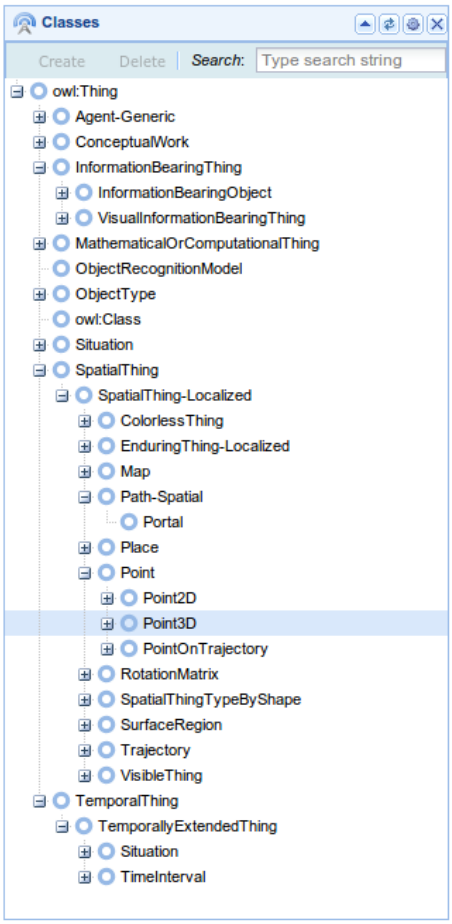


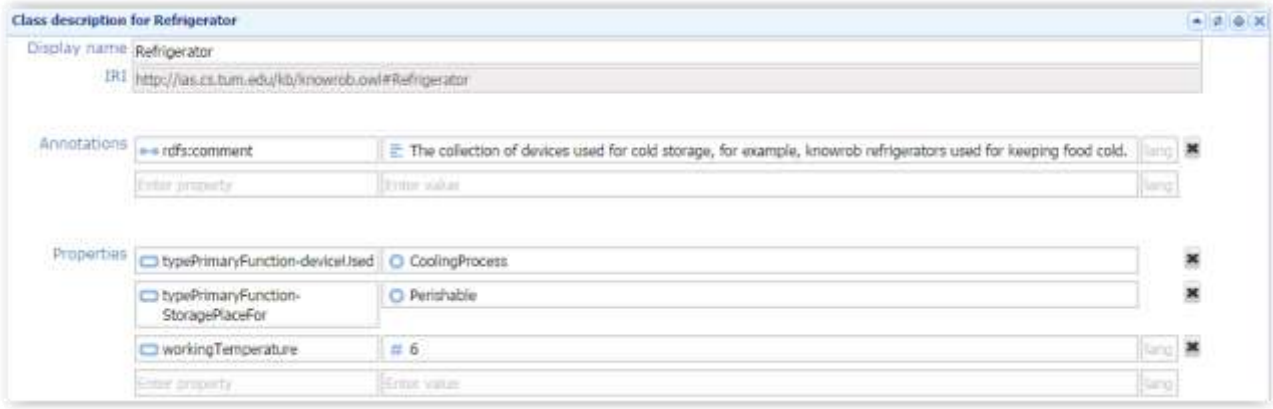
Figure 3: Example of the upper KnowRob taxonomy

A visual example of the upper KnowRob taxonomy, using Web Protégé[35] is depicted in figure 3.

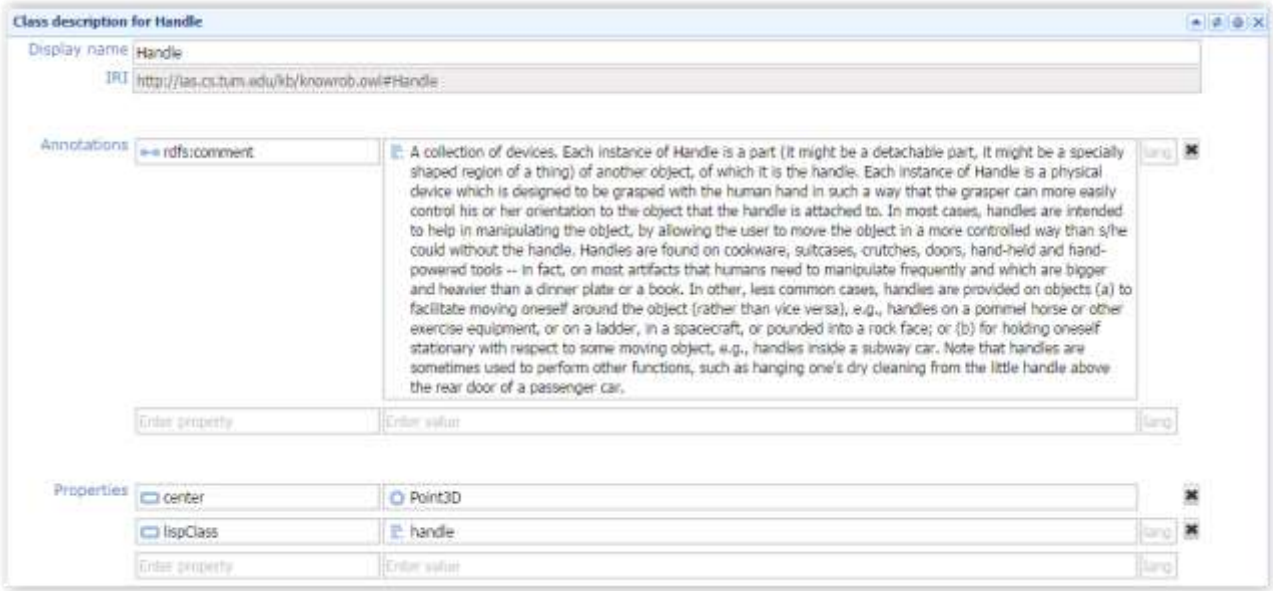
Let's see some examples of properties that specific classes have:

*Description of **cupboard** class*

*Description of **refrigerator** class*

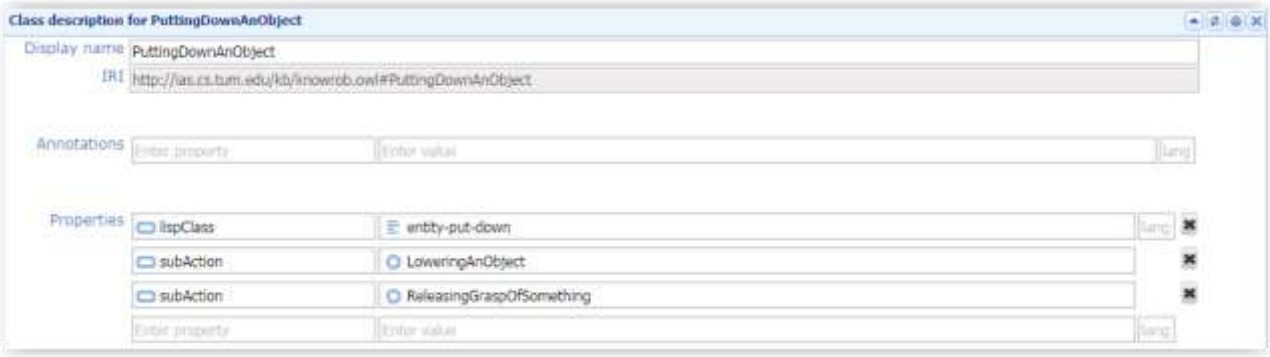


*Description of **handle** class*



[35] http://webprotege.stanford.edu/#List:coll=Home;

*Description of **PuttingDownAnObject** class*



It must be noted that KnowRob provides a basis for the ontology a robots needs to operate in a household and specifically in the context of the "assistive kitchen project", as stated in the Moritz Tenorth's dissertation [3], who was the main KnowRob creator and contributor. Thus, it is understandable that we will develop any extensions needed to be seamlessly employed in a household environment in general and specifically for the user scenarios described.

3.2. OpenAAL

In order to assess the problem of the social ontology, OpenAAL will be employed. As stated, OpenAAL is the result of the SOPRANO integrated project and it represents a flexible and powerful middleware for AAL scenarios. OpenAAL provides a full integrated platform for ambient assisted living, as it contains not only semantic storage and inference, but many tools for user context management, multi-paradigm context augmentation, context aware behaviours and others. Though, for this project's needs only the ontology will be needed. The high level ontology is contained in an owl file, fact convenient to our causes, as it can easily be merged to the KnowRob ontology. An example of the high level ontology tree is presented below:

- **OpenAALThing:** The top level concept. Every other concept / class is an OpenAALThing.
  - **Activity:** Describes a person's activity
  - **Acute-State:** Describes a "dangerous" state for AAL, such as **fall**.
  - **Appointment:** A person's appointments. Its subclass is the **Personal-Appointment**.
  - **Device-State:** Describes the state of a device (electronic). This class contains:
    - **Open-State:** Refers to the device's physical state; e.g. if an oven is open or closed.
    - **Power-State:** Refers to the device's electrical state; e.g. if an oven is on, off or standby.
  - **Due-State:** Due state can either be Due or Not-Due and can be related to an appointment to define whether the appointment is due or not due.
  - **Entity-With-Coordinates:** Every object that can be found in a household; e.g. a door.
  - **Language:** Self-explainable class that contains information about different languages (English, German etc).
  - **Locatable-Entity:** Every object that can be located in a household. This class contains:
    - **Device:** Examples are doors, lamps, TV, couch, oven etc.
    - **Low-Level-Thing:** Describes actuators and sensors. Though these do not refer to the robotics science:
      - **Actuator:** Class that contains Device-State-Actuators (e.g. remote control) and Message-Actuators.
      - **Sensor:** Class that contains Device-State-Sensors, Fall-Sensors and Message-Sensors.
    - **Person:** Class that differentiates between AP (Assisted Person) and Carers.

- **Location:** Contains Rooms and Room-Parts. Rooms comprise the Bathroom, Kitchen and Unspecified-Room classes.
- **Message:** An abstract class that contains:
  - **Answer:** Its subclasses are Confirmation, Yes and No.
  - **Information:** Contains Alarms (like Fall-Detected-Message), Notifications and Reminders.
  - **Question:** Its only subclass is Yes-No-Question
- **Message-Modality:** Contains messages that need to be assessed. Its two subclasses are Sound-Message and Text-Message.
- **Ontology-Statement**
- **OntologyALL-Datatype:** Variable types such as OpenAAL-Boolean, OpenAAL-DateTime and OpenAAL-String.

An example of the OpenAAL upper taxonomy is presented in figure 4.

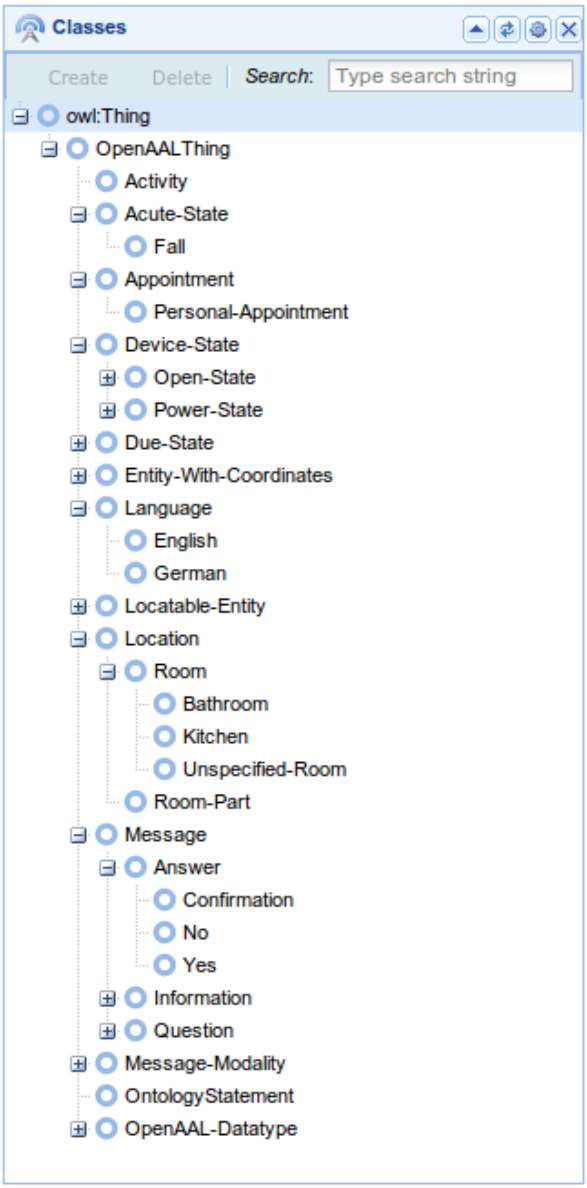


Figure 4: Upper OpenAAL taxonomy

It is apparent that the OpenAAL ontology provides the basis for semantically describe AAL scenarios. Though one observation is that it is not complete, i.e. it does not contain every class that can be found in a household environment, such as the difference elective appliances, locations or devices. Thus, this ontology will be enriched with the necessary classes and instances needed to test the user scenarios we prescribed in the RAPP project.

The second fact that should be mentioned is that the OpenAAL and KnowRob have some classes that are similar to a certain extend. Some examples are the *KnowRob:Thing* and *OpenAAL:OpenAALThing*, the *KnowRob:PhysicalDevice* and *OpenAAL:Device* etc. Taking this fact under consideration we decided to conceptually merge the two ontologies where possible and afterwards perform the necessary extensions.

Next, the actual ontology implementation and employment within the RAPP system will be described.

4. RAPP Ontology employment

4.1. Setup

As evident from the KnowRob description, its primary goal is to semantically integrate the ensemble of possible operations that can occur in a household with the capabilities and perception of robotic devices that operate within it. This fact suggests that the “natural environment” for the KnowRob to operate is a single household, where the spatiotemporal information of the robots and objects in it are kept and tracked. On a different basis, this suggests that a centralized system in which KnowRob will be installed will exist within the household where the robots operate. Unfortunately, this is not the case in RAPP.

As described, our main concern is to not perform important modifications in a household, and the absolutely needed modifications not to burden financially the end users. So, simple modifications like room and device tagging using QR tags can be applied, but installing a personal computer in order to enable heavy duty local operations is out of the question. Having said that, the only place where KnowRob could be installed is in the robotic devices. Thus, the main problem with this decision is that we intend to create an, as more as possible, globally applicable solution, i.e. for every robotic device that runs a Linux operating system. Of course not all robots have large storage and computational capabilities and NAO is an example of these robots.

Our way to overcome these problems is to install KnowRob and OpenAAL on the RAPP cloud part. This decision holds some advantages and disadvantages; the main disadvantage is that KnowRob isn’t utilized the way it was created for, meaning that the entirety of its capabilities will remain dormant. Of course this infers that we must “invent” a new way to employ the ontology, a way that will be described in chapter 4.3. On the other hand, this solution gives us the chance to create a centralized knowledge repository, from which all the robots can benefit, as described in the RAPP Description of Work.

4.2. Access

Both OWL files (KnowRob-common and OpenAAL ontologies) will be merged and loaded in the KnowRob software, in order to be able to perform semantic queries. As aforementioned, KnowRob is implemented in SWI-Prolog, thus it is hard to access semantic information from a C++ ROS node. Fortunately, the developers of KnowRob provide a KnowRob ROS stack that enables the queries to the ontology via a ROS message. As described in the deliverable D1.3, the operations that will be performed in the RAPP Platform (that contains the RAPP Store, the RIC – RAPP Improvement Centre and the HOP services) will be developed in the form of ROS nodes, as this is a tested way to create a closed system with intra-process communication. Each ROS node that provides services to the outside world, will be wrapped with a thin HOP layer, which in turn will uptake the task of communicating with the Robotic Applications (RApps) that are developed in HOP and exist in the robots. Thus each robot can have access to the knowledge repository by utilizing the provided HOP services.

4.3. Utilization

This section will describe the means of the ontology utilization within the RAPP boundaries. In a few words, after the ontology is expanded with the necessary classes, it will be used as inference tool for the semantic information, by creating anonymized class instances from the objects and actions observed by each robot and performing machine learning algorithms on them. On the other hand, the named (non-anonymous) information will be stored in a secure database, where a ROS node will have access. The concept behind these decisions will be made apparent by presenting a RApp example.

The following actors are distinguished:

1. **UserA** – the human.
2. **CoreAgent** - located on the "Robot platform" (i.e. "the robot"). The main robot controller.
3. **DynamicAgent** - current RApp running on the "Robot platform".
4. **HOPserviceRobot** - communicates the robot (both agents) with the RAPP platform. Part of the CoreAgent.
5. **HOPserviceRAPP** - communicates the RAPP Platform with the robot.
6. **RAPPplatform** – the cloud of RAPP resources.
7. **RAPPStore** – The App Store part of RAPP.
8. **KnowRob** – manages an Ontology database with a SWI-Prolog-program – allows a communication with the Ontology – located on the "RAPP platform".
9. **RobotDatabase** (Ensemble of files) - stores data, such as navigation map, needed object models and own state.
10. **SpeechRecognition** – a service application on the RAPP platform.

Those actors take part in processing and the communication associated with responding to the user's request. The below described procedure is based on the system architecture as defined in the deliverable D1.3.

Example

Let's assume that a robot gets a command from the user A, who asks it to "**search for hazards**".

The above sentence corresponds to the following sequence of transmission steps:

1. **UserA**: (voiced command "search for hazards") → **CoreAgent**, which turns it into a wav audio file: "command.wav"
2. **CoreAgent**: (command.wav) → **HOPserviceRobot** that packs it into a message "message.xml"
3. **HOPserviceRobot**: (message.xml) → **HOPserviceRAPP** at the RAPP platform that receives and unpacks the message
4. **HOPserviceRAPP**: (command.wav) → **RAPPplatform** that recognizes it is an audio file and calls a Speech Recognition service application from the RAPP platform
5. **RAPPplatform**: (command.wav) → **SpeechRecognition**, that is more than a sentence recognizer, as it needs to generate an appropriate action command corresponding to the sentence.
6. **SpeechRecognition**: ("hazards identification") → **RAPPplatform**
7. **RAPPstore**: ("hazardRApp") → **HOPserviceRAPP**, packs the application identifier and creates a "message.xml"
8. **HOPserviceRAPP**: (message.xml)→**HOPserviceRobot** at the Robot platform – unpacks the message
9. **HOPserviceRobot**: ("hazard RApp") → **CoreAgent**
10. **CoreAgent**: ("hazardRApp", download) → **HOPserviceRobot**
11. **HOPserviceRobot** → **HOPserviceRAPP** (pass the call)
12. **HOPserviceRAPP** → **RAPPplatform** (pass the call)
13. **RAPPplatform** → **RAPPstore** (pass the call)
14. **RAPPstore**: (hazardRApp_Nao executable) → **HOPserviceRAPP**

a. Eventually Step 14a: **RAPPstore** installs and **RAPPplatform** initializes the application part: hazardRApp_RAPP on the RAPP platform.

15. **HOPserviceRAPP** → **HOPserviceRobot**
16. **HOPserviceRobot** → **CoreAgent**, installs hazardRApp_Nao executable as the DynamicAgent and executes it

The robot enables (or downloads) the **Hazards Identification** RApp and immediately evokes a HOP service searching for "**hazard**" in the ontology.

17. **DynamicAgent**:("query", "hazard") → **HOPserviceRobot**
18. **HOPserviceRobot**:(message.xml) →**HOPserviceRAPP**
19. **HOPserviceRAPP**:("query", "hazard") → **KnowRob**; a query passed to the Ontology
20. **KnowRob**: ("oven.open") → **HOPserviceRAPP**
21. **HOPserviceRAPP**: (message.xml) → **HOPserviceRobot**
22. **HOPserviceRobot**: ("oven.open") → **DynamicAgent**

The service returns e.g. **open oven**, in order for the robot to search for it. Initially, the robot tries to identify if it has knowledge of the specific user's oven by securely querying the database, though no information is stored there.

23. **DynamicAgent**: ("query", "oven") → **RobotDatabase**
24. IF **RobotDatabase**: (model("oven")) == YES THEN GOTO Step 72
25. **RobotDatabase**: (empty) → **DynamicAgent**

Of course the robot does not know what an **oven** is, so another HOP service is called that queries for information about how an oven looks like (for example by getting SIFT [10] of SURF [11] features). Unfortunately, it is the first time that a robot tries to find the oven-related information, so there is none existent.

26. **DynamicAgent**:("query", "oven") → **HOPserviceRobot**
27. **HOPserviceRobot**:(message.xml) →**HOPserviceRAPP**
28. **HOPserviceRAPP**:("query", "oven") → **KnowRob**
29. **KnowRob**: ("empty") → **HOPserviceRAPP**
30. **HOPserviceRAPP**: (message.xml) → **HOPserviceRobot**
31. **HOPserviceRobot**: ("oven", "empty") → **DynamicAgent**

Then, the robot could ask the user to show it the oven and it captures an image of it. This image is uploaded to the RAPP Platform via a service and the features of the oven are extracted.

32. **DynamicAgent**:(speaker, "show the oven.open and oven.closed") → **CoreAgent**, synthesizes the speech signal
33. **CoreAgent**:("asking.wav") → **UserA**

From now on we need a lot of steps dealing with image processing – human pose detection, human arm tracking, human following, understanding the voice command (a human saying something like: "this is an oven.open") or hand gesture in the image!

34. **DynamicAgent**:("humanTrackRApp", download) → **HOPserviceRobot**
35. **HOPserviceRobot**: (message.xml) → **HOPserviceRAPP**
36. **HOPserviceRAPP** :("humanTrackRApp", download) → **RAPPplatform**

37. **RAPPplatform** (humanTrackRApp_Nao.exe) → **HOPserviceRAPP**
    a. Eventually Step 41a: **RAPPplatform** installs and initializes the application: humanTrackRApp_RAPP on the RAPP platform

38. **HOPserviceRAPP**(message.xml) → **HOPserviceRobot**
39. **HOPserviceRobot** (humanTrackRApp_Nao) → **DynamicAgent**, installs humanTrackRApp_Nao as the DynamicAgent and executes it
40. **DynamicAgent**: ("get map") → **RobotDatabase**
41. **RobotDatabase**: (map, ownposition) → **DynamicAgent**
42. **DynamicAgent**: ("get image") → **CoreAgent**
43. **CoreAgent**: (image.bmp) → **DynamicAgent**, an image to detect the human pose
44. **DynamicAgent**: ("detect_human", image.bmp) → **RobotDatabase**.RappFunction()
45. **RobotDatabase**: (objectstate, "human") → **DynamicAgent**
46. **DynamicAgent**: (map, ownstate, objectstate, "get move") →**RobotDatabase**.RappFunction()
47. **RobotDatabase**: (move) → **DynamicAgent**
48. **DynamicAgent**: (map, move) → **CoreAgent**, executes the move to follow the human
49. **CoreAgent**: (done) → **DynamicAgent**
50. **DynamicAgent**:(newposition) → **RobotDatabase**
51. IF **DynamicAgent**: ("humanpose != "shows object" ") THEN GOTO Step **42**"
52. **DynamicAgent**: (image.bmp, "detect pointed object") → **RobotDatabase**.RappFunction()
53. **RobotDatabase**: (objectstate) → **DynamicAgent**
54. **DynamicAgent**: (map, ownstate, objectstate, "get move") →**RobotDatabase**.RappFunction()
55. **DynamicAgent**: (move) → **CoreAgent**
56. **CoreAgent**: (done) → **DynamicAgent**
57. **DynamicAgent**: ("get image") → **CoreAgent**
58. **CoreAgent**: (image.bmp) → **DynamicAgent**

Human tracking is finished, as the oven (open, closed) is shown, and the image is captured:

59. **DynamicAgent**: ("humanTrackRApp_RAPP", "stop") → **HOPserviceRobot**
60. **HOPserviceRobot**: (message.xml) → **HOPserviceRAPP**
61. **HOPserviceRAPP**: (humanTrackRApp_RAPP", "stop") → **RAPPplatform**, and the application is stopped
62. **DynamicAgent**: (image.bmp, "oven.open", "analyse") → **HOPserviceRobot**
63. **HOPserviceRobot**: (message.xml) → **HOPserviceRAPP**
64. **HOPserviceRAPP**: (image.bmp, "oven.open", "analyse") → **hazardRApp_RAPP**
65. **hazardRApp_RAPP**: (imageProcRApp, image.bmp, "oven.open", "install") → **RAPPplatform**, installs the **imageProcRApp** executable and passes the data to it
66. **imageProcRApp** executable: (features, "oven.open") → **hazardRApp_RAPP**

Then two operations happen at the same time: A) an anonymous instance is created in the ontology, under the **oven** class, containing the oven's visual features and B) a registration is created in the database that connects the specific features to the specific user.

67. **HazardRApp_RAPP**: (features, "oven.open", "create Instance") → **KnowRob**, it creates new instance of type "oven" with given features
68. **HazardRApp**: (features, "oven.open") → **HOPserviceRAPP**
69. **HOPserviceRAPP**: (message.xml) → **HOPserviceRobot**
70. **HOPserviceRobot**: (features, "oven.open") →**DynamicAgent**

71. **DynamicAgent**: (“oven.open”, features, UserA, “create Instance”) →**RobotDatabase**.RappFunction() creates new instance of “oven” and registers it with given user

Now the robot has identified the oven and executes machine vision algorithms in order to classify its state (**open** or **closed**).

72. **DynamicAgent**: (“oven”, getmodel) →**RobotDatabase**
73. **RobotDatabase**:(features) → **DynamicAgent**
74. **DynamicAgent**:(“get move”, map, ownstate, oven, “find object”) → **RobotDatabase**.RappFunction()
75. **RobotDatabase**: (move) → **DynamicAgent**
76. IF **DynamicAgent**(move == empty) THEN GOTO **89**
77. **DynamicAgent**:(move, “execute”) → **CoreAgent**
78. **CoreAgent**:(done) → **DynamicAgent**
79. **DynamicAgent**:(newposition) → **RobotDatabase**
80. **DynamicAgent**:(“get image”) → **CoreAgent**
81. **CoreAgent**:(image.bmp) → **DynamicAgent**
82. **DynamicAgent**:(“oven”, features, map, ownstate, “find object”) → **RobotDatabase**.Function()
83. **RobotDatabase**: (instances[], features[]) → **DynamicAgent**
84. **DynamicAgent**: (instances[], features[], [“oven.open”, “oven.closed”], classify → **RobotDatabase**.Function()
85. **RobotDatabase**:(instances[], classes[]) → **DynamicAgent**
86. IF **DynamicAgent**: (NOT(type(instance) == “oven.open”)) THEN GOTO **74**
87. **DynamicAgent**: (“open oven”, “create audio message”) → **CoreAgent**.RappFunction()
88. **CoreAgent**: (audio.wav) → **UserA**
89. **DynamicAgent**: “continue searching for other Hazards” or “finish the application hazardRApp_Nao“

Supposedly, user B has downloaded the same RApp in his robot, which performs a daily routine check about hazards. Again, robot B cannot find any information about the B user’s oven in the database, so it queries for general knowledge about ovens in the ontology, where in fact a set of features are found, created from performing machine learning algorithms on all the available feature sets that correspond to all the oven’s instances (whose number is currently 1). Next, the robot roams the household, trying to visually identify the oven. Unfortunately, user B has a different oven than user A, so after an amount of time the robot decides that the oven cannot be found and invokes help from user B (similarly to user A before). So, again a different oven’s features are creating an anonymous instance in the ontology’s **oven** class and a named registration in the secure database.

In the meantime, every night (e.g.) batch processes are invoked from the RIC (RAPP Improvement Center) that collect the available anonymized instances of each class in the ontology and try to create general knowledge models, i.e. means of generally identifying the objects or actions by specific observations. At some point, a pretty good oven model is created, so that each robot can successfully identify a never-seen-before oven in a new household.

Although not scientifically exact and quite simplified, the above example showcases how the ontology will be utilized within the RAPP project. Briefly, it will comprise anonymized knowledge in the form of class instances from observed objects or actions. These instances will be used to derive generic models for each class, enabling the generalization of concepts, thus the knowledge centralization and distribution, as described in the RAPP Description of Work.

Conclusions

In this document, the RAPP approach towards the utilization of ontologies is described. The ontology employment will play a crucial role in the *Cloud Robotics* aspect of the RAPP project, enabling the knowledge collection, processing and distribution among heterogeneous robots that have certain common sensors.

Initially, the state of the art was described in the field of knowledge sources available online. As RAPP is oriented towards two different areas – robotics and inclusion or AAL – the main ontology schemes of these two fields were investigated. Initially, some general purpose ontologies were presented, which attempted to capture semantic information on an encyclopaedia scale. Then, the state of the art robotic ontologies were described (the most important of which is KnowRob), as well as the AAL/ICT ones. Finally some general information sources were presented, which are known to have been used in conjunction to robots by different scientific groups.

The third chapter contains the ontology design, i.e. which ontologies were selected for employment from each field and the reasons beyond this choice. As stated there, the selected ontologies were KnowRob, concerning robotics and OpenAAL concerning AAL applications.

Finally, the last chapter provides an insight on the actual ontology setup, means of access and utilization within the RAPP project. Additionally, an intuitive example was given, in order to conceptually present and support the aforementioned choices.

The main drawback of the presented implementation is that the KnowRob platform is not utilized in its full power, as it cannot be installed locally within a household. This can be avoided in future versions of RAPP, where additional tools will be supported for use, such as mobile phones, tablets or even PCs. If so, a future extension of the system may include local KnowRob instances that provide both specific spatiotemporal information to the robots that operate in the house, and generalized high level knowledge to a cloud instance of the RAPP Platform.

References

1. Waibel, Markus, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Haussermann et al. "A World Wide Web for Robots." IEEE Robotics & Automation Magazine (2011).
2. Arumugam, Rajesh, Vikas Reddy Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, Foong Foo Kong, Appadorai Senthil Kumar, Kang Dee Meng, and Goh Wai Kit. "DAvinCi: A cloud computing framework for service robots." In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 3084-3089. IEEE, 2010.
3. Tenorth, Moritz, and Michael Beetz. "KnowRob—knowledge processing for autonomous personal robots." In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 4261-4266. IEEE, 2009.
4. Lemaignan, Séverin, Raquel Ros, L. Mosenlechner, Rachid Alami, and Michael Beetz. "ORO, a knowledge management platform for cognitive architectures in robotics." In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 3548-3553. IEEE, 2010.
5. Henricksen, Karen, Jadwiga Indulska, and Andry Rakotonirainy. "Modeling context information in pervasive computing systems." In Pervasive Computing, pp. 167-180. Springer Berlin Heidelberg, 2002.
6. Wolf, Peter, Andreas Schmidt, and Michael Klein. "SOPRANO-An extensible, open AAL platform for elderly people based on semantical contracts." In 3rd Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI'08), 18th European Conference on Artificial Intelligence (ECAI 08), Patras, Greece. 2008.
7. Mileo, Alessandra, Davide Merico, Stefano Pinardi, and Roberto Bisiani. "A logical approach to home healthcare with intelligent sensor-network support." The Computer Journal (2009).
8. Saldaña-Jimenez, Diana, Marcela D. Rodríguez, Juan-Pablo García-Vázquez, and Adán-Noé Espinoza. "Elder: An ontology for enabling living independently of risks." In On the Move to Meaningful Internet Systems: OTM 2009 Workshops, pp. 622-627. Springer Berlin Heidelberg, 2009.
9. Wolf, Peter, Andreas Schmidt, Javier Parada Otte, Michael Klein, Sebastian Rollwage, Birgitta König-Ries, Torsten Dettborn, and Aygul Gabdulkhakova. "openAAL-the open source middleware for ambient-assisted living (AAL)." In AALIANCE conference, Malaga, Spain, pp. 1-5. 2010.
10. Lowe, David G. "Object recognition from local scale-invariant features." In Computer vision, 1999. The proceedings of the seventh IEEE international conference on, vol. 2, pp. 1150-1157. Ieee, 1999.
11. Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." In Computer Vision–ECCV 2006, pp. 404-417. Springer Berlin Heidelberg, 2006.