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

D3.4 RAPP Store

Deliverable status: First Version
File Name: RAPP_D3.4_V1.0_20150526.pdf
Due Date: 31 May, 2015
Submission Date: 31 May, 2015
Dissemination Level: Public
Task Leader: 5 - Ortelio
Author: Alexandros Gkiokas

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
INGEMA	Fundacion Instituto Gerontologico Matia - Ingema	Spain
AUTH	Aristotle University of Thessaloniki	Greece

Revision Control

VERSION	AUTHOR	DATE	STATUS
0.9	Alexandros Giokas (ORTELIO)	May 15, 2015	First Draft
1.0	Emmanouil Tsardoulias (CERTH/ITI)	May 30, 2015	Final corrections

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

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,

Deliverable named "D3.4 RAPP store" is a prototype. The code of the RAPP Modules is available in the project's git account: https://github.com/rapp-project/. This document is a brief description of the implemented RAPP store.

1 The RAPP Store: Current state

Currently, the RAPP store serves as a semi-functional prototype, which demonstrates the submission of source code, and how that code is compiled, processed, and packaged automatically, into a RAPP. This is an alpha (testing) version of the final product, which should be semi-autonomous, as a human user will be required to verify source code sanity and check for malicious source code. It follows the *Smartphone App* paradigm, through a web-portal, which we call "The RAPP store". At the moment, the store supports user registration, user login and submission of a new RAPP (C++, ROS, JavaScript) with functional C++ and JavaScript packaging.

2 Submission

The procedure which is used to submit a new RAPP is described in the figure below. In the current prototype, 3 different languages are used: C++11, ROS/C++ and JavaScript.

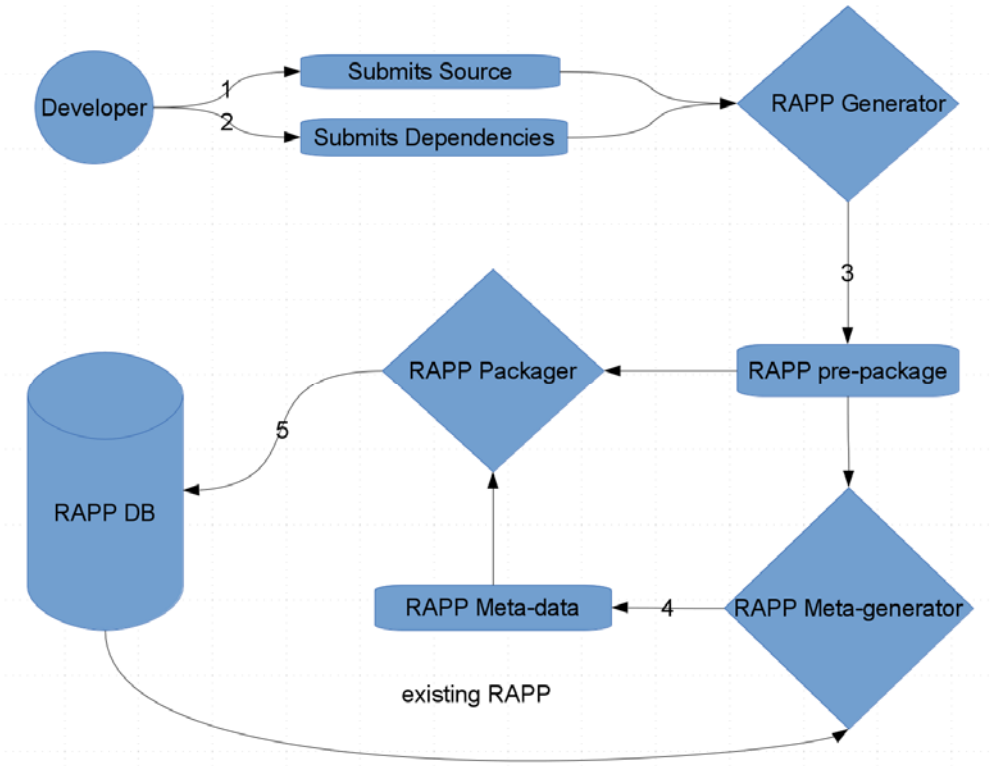

Figure 1:RAPP store architecture

However, the currently functional prototype only supports C++11 and JavaScript packaging, due to ROS using *catkin* as a wrapper around GCC's *make* compile toolset. The C++11 compilation and linkage process relies upon UNIX GNU GCC toolset, whereas the JavaScript execution requires Node.Js to be present on the robot.

Each user is able to create a new RAPP with a unique name, and version it appropriately. For each new RAPP, a new sandboxed directory is created on the RAPP store, where the user submits source code. We have provided an automatic file scanning script which detects source code and generates a CMake file. For C++ CMake is the project management and built tool.

JavaScript doesn't require building or linking, therefore we simply enable uploading of source files and we package them into a RAPP. Other types of files can also uploaded, such as SQLite databases, JSON, XML, audio files and even static libraries. We do not allow executables as that could be a severe security hole.

The web-interface, enables source code editing for every source file uploaded, with changes saved right away. This in turn, creates an online source-code generation and compilation tool. New files and directories can be created within the RAPP root directory, as well as deleted, copy/cut and paste, and renamed.

A requirement is that if C++ or ROS is used, the user must explicitly include the dependencies which his or her RAPP requires. In C++ this is done via the CMake *find_package or find_library* commands, whereas in ROS we use the *add_package* in combination with the underlying CMake mechanisms. For JavaScript we assume that the user will import the dependencies as source files. A future version may see a JavaScript validation tool used, in order to ensure that the provided code will execute without errors.

3 Meta-Data

A certain amount of meta-data is given by the developer for each RAPP:

- RAPP name
- RAPP version
- Programming language
- Public RSA key (optional)
- An Icon (*gif,jpeg,png*) (optional)
- Keywords and Tags (optional)
- Dependencies
- Original Entry Point (e.g., the name of the executable which contains the "*main*" function)
- SHA512 hash of the original executable as built in the RAPP store (optional)
- Homepage (optional)

Most of this data is saved in the RAPP database, and some is included with the packaged RAPP. The next version of RAPP store will use package signing for security and anti-tampering, as well as for version control and patching on the robot.

4 RAPP Package & Execution

Packaging is done using the GNU *Tar* archiving utility, in combination with the *Gzip* compression utility, which has become a standard in the UNIX world. The HOP framework used by the RAPP platform and provided by INRIA, already uses the *Tar/Gz* combination, therefore interaction between the RAPP packaged on the store and the HOP platform on the robot is seamless and transparent. The RAPP package contains a JSON description file with some of the meta-data as described in the previous section, as well as a file with the same name as the RAPP name, which is assumed to be the original entry point (OEP). In the case of JavaScript, the HOP platform simply executes the JavaScript bitcode, whereas in the scenario of a C++/ROS RAPP package, we provide a JavaScript wrapper, which uses the "*fork + exec"* procedure to spawn a new child process that executes the C++/ROS binary RAPP. Currently no parameters are passed to the executing RAPP, and no return types are obtained. In the near future we will examine pipes and passing data back and forth.

5 RAPP Store dependencies

The store itself depends upon other software. Most notably, the store is executed within a Virtual Machine for safety purposes. It uses Apache2/PHP and MySQL, as well as the GNU toolset (gcc, make), CMake, ROS indigo (catkin) and

Node.Js. The store extensively uses JavaScript for the user interface (web-portal) and all the libraries needed are included with the deliverable source code.

The database in MySQL is currently shared across with the RAPP platform, but at some point in the near future they will be separated.