Research and Development of a New Robot Soccer Platform

J. Boonen, M. Houbraken, O. Lehmacher, J. Robers, G. Theunissen

January 21, 2016

Abstract

The RoboCup Soccer League is an international competition in which teams from all over the world compete in order to develop autonomous humanoid robot soccer players. The Dutch Nao Team participates in the Standard Platform League: a domain in the RoboCup for which teams find the best software for a standard robot, in this case the NAO robot. This summer, the Dutch Nao Team has decided to build a new platform for this software development from scratch, based on ROS. This research investigated whether this basis was the right choice, and concluded that ROS is indeed a good option to base a platform on. The research then focused on battling some of the numerous challenges that occurred when combining ROS with the NAO robots for this specific research domain.

1 Introduction

The RoboCup Soccer League is an international initiative that aims to promote the fields of Robotics and Artificial Intelligence and to improve already existing technologies. Its official goal consists of achieving a soccer team of fully autonomous humanoid robot players that is able to win a game against the winner of the most recent World Cup by 2050, while complying with the official FIFA-rules [6]. Currently the RoboCup is divided into several competition domains, one of which is the Standard Platform League in which all teams use identical robots and focus on proper software development rather than robot mechanics. The robots used in this League are NAOs, humanoid robots developed by Aldebaran Robotics [6].

This research project focuses its research around the following problem statement:

Find and research a new software platform that can be used with the NAO robots in the Standard Platform League

This problem statement can be subdivided in two research questions:

1. What is the most appropriate platform for this application?
2. What is needed additionally for this platform to work with the NAO Robots?

The structure of this paper looks as follows. First, some background knowledge will be provided on the NAO robots and the history of the Dutch Nao Team. Then, the two sections thereafter will aim to each answer one of the research questions. Finally, a conclusion will be drawn and the possible future work on this topic will be presented.

2 Background

This section will give a short introduction on the Standard Platform League and the NAO robot. Furthermore, a context will be given around the Dutch Nao Team and their previous experiences in the RoboCup Standard Platform League.

2.1 The NAO Robot

The RoboCup Standard Platform League uses a standard robot that all teams must conform to. This makes sure that in this League improvements are made on the software development only, and no attention has to be paid to improving the hardware of the robots. The robot has to be able to operate fully autonomously, no remote control by humans or other computers is allowed during the game. The first robot the SPL used as its standard robot, was the Sony AIBO [4] in 1999. At that time, the League was still called the Sony Four-Legged League and only a small number of teams were invited to join. After Sony announced it would discontinue the production of the AIBO in 2006, the League searched for a new robot and considered the Aldebaran Robotics humanoid NAO. Then in 2009 the League decided that the NAO would be the new standard robot and the name of the League was changed to the Standard Platform League.

2.2 The Dutch Nao Team

The Dutch Nao Team has been one of the competing teams in the RoboCup since 1998, and competes in the Standard Platform League since 2004 [8]. At that point, Python was their main programming language until they switched to a specialised code base in 2013. This code base (then called NAOTH) was developed by the University of Amsterdam and specifically designed for the Standard Platform League [9]. Due to lack of documentation, it was decided to use the B-Human framework instead during the soccer competitions of 2014 and 2015. The B-Human framework is developed by another competing team in the RoboCup from the University of Bremen [10]. However, this summer the Dutch Nao Team decided to develop their own framework, starting from scratch while developing it on the basis of ROS (Robot Operating System). In order to make the transition between the two platforms easier, it is intended to encapsulate some of the most recent version of the B-Human motion model and use it in the new framework. This is possible as it was done before in the research conducted by the University of Chile [1]. Since Maastricht University is currently part of the DNT, this research may contribute to their development of a new soccer platform.

3 Choice of Platform

When developing a framework to operate the NAO robots from scratch, many different platforms are available. Some of these platforms were compared during this research project, such as Choregraphe, NAOqi, B-Human and ROS. These different options will be presented in the following paragraphs and their advantages and disadvantages will be explained.

3.1 Choregraphe

First, Choregraphe is an easy to use platform that was developed by Aldebaran itself as an accompanying tool for the NAO robot. It uses an understandable graphical interface that allows for pre-programmed modules to be dragged and dropped into a larger structure that is compatible with the robot. This interface is easy to understand for a user with limited programming knowledge. These modules can be created from scratch as well, but in order to do that the user needs a comprehensive understanding of either Python or C++. It is made to work with the NAO so an advantage is that it is easy to start up and make the NAO do simple tasks. However, it is a very simple platform that does not include a large amount of readily made modules. The disadvantages of this platform are that the number of modules that can be put together is limited, and there are no specific modules available for playing soccer.

3.2 NAOqi

Aldebaran developed another framework for the NAO robot, called NAOqi. It is the operating system that runs on the NAOs and acts like a development kit in which the user can program tasks in either Python or C++. Since it is provided by Aldebaran, it is compatible with the NAO robot and therefore easy to set up, but each task needs to be programmed from scratch.

3.3 B-Human

B-Human is a soccer specific platform that was developed by the team representing the University of Bremen [10]. It has been used by several other teams in the Standard Platform League, amongst which is also the Dutch Nao Team. This platform is ready to use, well documented and has active support. However, using it means being dependent on another team in the same competition. On top of that, it is difficult to adapt or optimize specific features in the platform without consulting the B-Human team.

3.4 ROS

Finally, the Robot Operating System (ROS) is an open source platform that can be used for a large amount of applications. It contains a large set of ready-made libraries, such as messaging, localization and mapping modules. A ROS-version that is specifically developed for the NAO robot is also available. The fact that the platform is open source implies that a large community is currently working on and improving it, so it also has active support. In order to use ROS, the user needs to have knowledge of C++ and run a version of Linux (preferably Ubuntu). A drawback is that it has a very steep learning curve, which makes it difficult to start running ROS on a robot without prior knowledge. However, since ROS currently is the standard operating system for most robotics projects many users working with the soccer platform will most likely already have this background knowledge. If not, it will only be highly useful for their future career to learn more about this platform. Furthermore, when fully developed and bug-fixed, it will be easy to implement and improve upon different modules that are needed for the Standard Platform League.

3.5 Comparison

In table 1, all these options are compared. Taking all these factors into account, it can be concluded that both Choregraphe and NAOqi would not be very suitable for a soccer league platform. Expanding these frameworks to the extent at which they would be useful for playing soccer would most likely have to be done from scratch and would be unnecessarily difficult. When comparing the B-Human framework to the ROS framework,

	Advantages	**Disadvantages**
Choregraphe	- Easy to understand - Simple drag & drop UI - Easy start up - Compatible with NAO - Contains built-in simulator	- Number of modules is limited - No specific modules for soccer - Need knowledge of Python or C++ to develop own modules
NAOqi	- Easy start up - Compatible with NAO	- Each task needs to be programmed from scratch - Need knowledge of Python or C++ to develop own tasks
B-Human	- Ready to use - Well documented - Active support	- Dependent on other teams in the League - Difficult to understand - Difficult to optimize features
ROS	- Open source - Large range of applications - Readily available useful libraries - Large community - Currently standard OS for robotics applications	- Need knowledge of C++ - Need to run Ubuntu or other version of Linux - Steep learning curve - Difficult start up - No compatibility with the NAO

Table 1: Shows the different options for the development of a new soccer platform and their advantages and disadvantages.

the advantages of ROS in the long term outweigh the usability advantages of B-Human. Therefore, it can be concluded that a framework based on ROS would be the best choice for the development of a soccer league platform. The Dutch Nao Team has come to the same conclusion independently of this research team, and therefore the decision is made to use ROS as a basis for the new platform.

4 Development Progress

After deciding on the ROS platform, the development process could be started. In this section the development progress is discussed. First, a short explanation is given on the architecture and workings of the ROS platform. Then, it is discussed what is necessary to get the platform running. After that, it is explained what is required to play robot soccer using this platform. To get the platform running some previous work can be used. This previous work is discussed in the second subsection. In the third subsection an overview is given of what is needed in order to play soccer with this platform. In the last two subsections the current achievements are presented, and some difficulties that were encountered in the process are summed up.

4.1 The ROS Platform

The ROS platform uses a structure consisting of a master, which is in control, and a number of nodes. These nodes can be anything, from small programs to a full robot. The communication between nodes happens over so-called topics. Nodes can publish messages on one or multiple topics or they can subscribe to one or multiple topics. If a node is subscribed to a certain topic, it recieves all the messages that are published on this topic. The master generally runs on a PC and the nodes can either run on the same PC or on an external system or both. A detailed overview of the ROS platform can be found on the website [13].

4.2 Running ROS on the NAO Robot

A special version of ROS is available for the NAO robot [11]. In principle, this ROS version is ready to use and can be installed on a NAO robot using the available tutorials. Additionally, some teams that have competed in the RoboCup in the past already used ROS as a basis for their soccer platform, for example a team from Sweden [2] and Chile [1]. There are many tutorials available online that explain the ROS platform and all of its features. The use of these tutorials proved to be somewhat limited however, as will be explained in the last section. Because of the limited use of the available tutorials, the installation of ROS on the NAO proves to be more difficult than expected. In addition, a development environment has to be set up to allow for easy development and compilation of custom C++ or Python code. This code is ordered into packages which then can be run as ROS nodes on the NAO robot. The installation of ROS on the NAO robot is described in Appendix B. A tutorial to set up a development environment can be found in Appendix A.

4.3 Playing Soccer

The RoboCup rules dictate that limited communication is allowed between robots. Therefore running only one master on one NAO robot and having the other robots connect to that master is not allowed. Similarly, running a master on an external PC is not allowed. Thus an individual master has to be set up and run on each NAO robot. These should only communicate when necessary. The form of this communication can be chosen freely (e.g. WiFi or speech). There has been some previous work on this issue and there are some solutions available. One solution [3] makes use of a special ROS package [12] to allow a network of ROS masters to communicate with each other. This package is however not available for the ROS distribution used for the NAO. Therefore, to use this solution, the package needs to be ported somehow to work on the NAO robot. Another solution [7] makes use of a package for WiFi communication. This package or a similar package allowing WiFi communication should be available for the NAO robots. Therefore this second solution seems to be the best choice for solving the issue of needing multiple masters, however more research is necessary to confirm this.

To achieve actual soccer behavior in the NAO robots, a lot of implementational work has to be done. The work can be spread over a longer time period by using an intermediate solution. In this intermediate solution, parts of the B-Human platform [10], mentioned in section 3, can be encapsulated and integrated into the ROS framework. This can be seen as a transition process where at each iteration, a part of the B-Human platform that was encapsulated can be replaced by a new ROS node. The encapsulation of parts of the B-Human platform is an idea that is also proposed by the Dutch Nao Team in their qualification document for the RoboCup 2016 [9]. Furthermore, since winning a soccer match in the RoboCup is fairly ambitious using a new platform like ROS, it is best to focus more on the technical challenges of the RoboCup first. These technical challenges focus on single tasks instead of a whole soccer game. For example, one challenge could be on how to take corner kicks.

4.4 Achievements

A ROS distribution is currently running on one of the NAO robots. Specifically, the version that is installed is an adapted version of ROS Indigo tailored for the use on a NAO robot [11]. A ROS master can be started on the NAO robot and the provided nodes and topics can be accessed. Also a development environment has been set up to allow development and compilation of C++ or Python code. This development environment makes use of a virtual machine with NAOqi installed. The installation of the ROS platform on the virtual machine and on the physical NAO robot is practically the same. The development and compilation of C++ code has been tested and it is possible to run custom C++ code on the NAO robot in the form of a ROS node. So far only basic behavior has been achieved since a lot of time has been lost on dependency issues and other difficulties which will be explained in the next section.

4.5 Difficulties

During the development process several difficulties were encountered which caused delays. First of all, as mentioned before, there is an abundance of tutorials available on ROS for the NAO platform and for the ROS platform in general. The problem with these tutorials is that they are often incomplete and mostly outdated. The tutorials are mostly written by community members and can therefore be very specific to that user's system. Another problem is that the ROS distribution for NAO is quite complicated. Most tutorials describe how to run ROS on the NAO and controlling it via an external PC (for example, through RVIZ [5]). The goal in this project however is to run ROS completely and independently on the NAO robot itself. Furthermore, the ROS distribution for NAO does not seem to be under active development. Most tutorials and forum posts are a few years old and also some packages are based on older ROS distributions. This causes issues with certain dependencies, which leads to a lot of work trying to figure out which dependencies are missing and how to install or bypass them. Additionally, the Linux distribution running on the NAO robot (Gentoo) is very restrictive and uses its own package manager (Portage). This complicates the problem of missing dependencies even further, since some particular dependencies are not available in Portage and thus have to be downloaded on a different Linux distribution (e.g. Ubuntu) and then copied to the NAO robot. One can understand that this is not an ideal way of working. Finally, there is an issue with the file system on the NAO robot. By default, the Gentoo operating system tries to install everything on the root folder of the NAO robot. Since this root folder has only limited disk space, it quickly runs out of space. When there is no more space left on the root folder, it is not possible to install the necessary ROS dependencies. It is possible to insert an SD-card into the NAO robot, but for some reason this can not be used to install system dependencies.

To summarize, the following difficulties were encountered during the development process:

- Outdated or incomplete tutorials
- Tutorials often specific to the author's system configuration
- Running a ROS master on the NAO is complicated

- ROS distribution for NAO does not seem to be under active development
- Operating system on the NAO robot is restrictive
- Limited disk space on the NAO robot

5 Conclusion

This research project aims to solve the following problem statement: *Find and research a new software platform that can be used with the NAO robots in the Standard Platform League.*

In theory, to base the soccer platform on a ROS framework would be a well motivated choice. Choregraphe and NAOqi would not be very suitable, since the process of making these frameworks useful for the soccer application would be unnecessarily difficult. B-Human has advantages, as mentioned in section 3, but the dependency on another team in the League is undesirable. ROS has many useful properties, such as a large backing up community, as well as readily available libraries.

However, in practice, ROS proves to be difficult to integrate with the NAO robot for this specific application as explained in section 4. Many achievements were accomplished, such as the installment of a ROS distribution on the NAO robot, the setup of the development environment and the ability to run custom C++ code via ROS on the NAO. However, tutorials are mostly outdated and many dependency problems occur, which caused a lot of difficulties and hindered the development progress. Therefore, ROS needs to be researched more extensively. It needs to be well understood and properly integrated with the NAO robot in order to be useful for this application. Thus, a platform based on the ROS framework might not be the best short term solution since a lot more research is needed. However, this can be bypassed by encapsulating useful parts of B-Human and integrating them into the ROS framework.

6 Future Work

More experiments with the combination of ROS and the NAO robot need to be done in order to get more in depth knowledge of ROS. The tutorials that are available now need to be reviewed and updated since most are outdated.

Furthermore, useful modules can temporarily be taken and encapsulated from the B-Human platform and used in the new ROS framework. This can act as an intermediate solution until the framework for the soccer league is completely redesigned, a process that will most likely take a few years to be accomplished.

Issues like the multi-master problem explained in section 4.3 need to be solved, and the knowledge should be shared with the Dutch Nao Team.

References

[1] Forero, Leonardo Leottau, Yáñez, José Miguel, and Solar, Javier Ruiz-del (2014). Integration of the ROS framework in soccer robotics: the NAO case. *RoboCup 2013: Robot World Cup XVII*, pp. 664–671. Springer.

[2] Fredrik Heintz, Mattias Tiger Gustav Häger Jon Dybecl Karl Homquist Tore Haglund Michael Felsberg, Fredrik LöfgrenLinköping Humanoids - Team Description.

[3] Hernádez Juan, Sergi and Herrero Cotarelo, Fernando (2015). Technical report: Multi-master ROS systems.

[4] Hu, Huosheng and Gu, Dongbing (2001). Reactive behaviours and agent architecture for sony legged robots to play football. *Industrial Robot: An International Journal*, Vol. 28, No. 1, pp. 45–54.

[5] Kam, Hyeong Ryeol, Lee, Sung-Ho, Park, Taejung, and Kim, Chang-Hun (2015). Rviz: a toolkit for real domain data visualization. *Telecommunication Systems*, pp. 1–9.

[6] Kitano, Hiroaki, Asada, Minoru, Kuniyoshi, Yasuo, Noda, Itsuki, and Osawa, Eiichi (1997). Robocup: The robot world cup initiative. *Proceedings of the first international conference on Autonomous agents*, pp. 340–347, ACM.

[7] McEachern, Kevan (2012). Using Multiple Masters to Preserve Robot Independence and Reduce Network Latency in a ROS System. `https://www.ri.cmu.edu/education/McEachernRISSposter.pdf`. Accessed: 15-01-2016.

[8] Oomes, Stijn, Jonker, Pieter, Poel, Mannes, Visser, Arnoud, and Wiering, Marco (2004). The dutch aibo team 2004. *Proceedings CD of the 8th RoboCup International Symposium*.

[9] Kok, Mustafa Karaalioglu Caitlin Lagrand Michiel van der Meer Jonathan Gerbscheid Thomas Groot Patrick de, Sébastien Negrijn and Visser, Arnoud (2015). Dutch Nao Team, Team Qualification Document for RoboCup 2016 Leipzig, Germany.

[10] Röfer, Thomas, Laue, Tim, Burchardt, Armin, Damrose, Erik, Müller, Judith, and Rieskamp, Andrik (2008). B-human team description for robocup 2008. *in RoboCup 2008: Robot Soccer World Cup XII Preproceedings*, Citeseer.

[11] Aldeberan nao - package summary. `http://wiki.ros.org/nao`. Accessed: 15-01-2016.

[12] Package summary of the multimaster_fkie package. http://wiki.ros.org/multimaster_fkie. Accessed: 15-01-2016.

[13] Ros. http://www.ros.org/. Accessed: 15-01-2016.

Appendix

The appendix contains additional information about how to set up ROS on either a virtual machine or a real robot. The following steps are tested for NAOqi version 2.1.4 and ROS Indigo. They may not work for different versions of NAOqi or ROS due to changes, that were made for either of these programs. Additionally, Appendix C explains the cooperation within the project group during this research project.

A Setting up a Development VM

The following section contains a guide through the necessary steps, that need to be done in order to set up a virtual machine, which can be used to program and compile packages for the NAO.

A.1 Download and Import the VM

- Download the Image of the NAOqi VM from Aldebaran Website (account required)
- Download and Install Virtual Box
- Import the downloaded file into the Virtual Box (File >Import Appliance >downloaded file)
- Connect to the VM

  ssh nao@localhost -p 2222

  type yes

  password: nao

A.2 Install Packages and Dependencies for ROS

- su -c 'sed -i "s |#allow user nao to shutdown the robot|nao ALL=(all) all\n&|" /etc/sudoers'

  root password: root
- Download the bootstrap file

  curl -k -s https://chili-research.epfl.ch/ros4nao/bootstrap.sh | sh
- Install required packages

  sudo emerge log4cxx netifaces pyyaml poco apr apr-util

  Note: may need to add –autounmask-write

  Note: may need to call sudo etc-update
- Install ROS

  robotpkgin install nao-robot

  You need to call source ~./bash_profile
- Install some basic packages

  robotpkgin install ros-robot ros-image-common

  See robotpkgin for more packages

A.3 Setting up the Workspace

- Create folder for the workspace

  mkdir -p /catkin_ws/src
- Initialize the catkin workspace

  cd /catkin_ws/src

  catkin_init_workspace
- Build the workspace

  cd /catkin_ws

  catkin_make
- Source the /devel folder

  source ~/catkin_ws/devel/setup.bash

If the error Unable to find either executable 'empy' or Python module 'em' occur than install the python empy module with: sudo pip install empy

A.4 Creating and Building Packages

- Create the package folder

  cd ~/catkin_ws/src

  catkin_create_package <NAME> <DEPENDENCIES>
- Create a source file

  cd ~/catkin_ws/src/<PACKAGE_NAME>/src

  touch [file].cpp
- Configure the CMakeList.txt

  open the CMakeList.txt file under src/<NAME>/

  uncomment the following entries

  add_executable(<node_name> <path_to_src_file>)

  add_dependencies(<node_name> <src_file.cpp>)

  target_link_libraries(<node_name> ${catkin_libraries})
- Build the package

  cd ~/catkin_ws

  catkin_make
- Sync the file to the robot

  catkin_make install -DCMAKE_INSTALL_PREFIX=/opt/openrobots

  sudo rsyn -raz /opt/openrobots –no-perms -O nao@<IP addr>:/opt/

A.5 Launch the Code on the NAO Robot

- Start ros

  roslaunch nao_bringup nao.launch
- Start your node (new terminal)

  rosrun [package_name] [node_name]

If the error unable to contact my own server occurs, export the ip of the NAO robot: export ROS_IP=<ros_ip>. This might be needed to be done for each new terminal.

B Installation of ROS on a Real NAO

This section deals with installing ROS on a real NAO. The following steps were tested with NAOqi 2.1.4, ROS Indigo on a NAO V4. They might not work when using other versions of ROS or NAOqi.

B.1 Flash the NAO

Flash the NAO in order to ensure, that there is enough space left on the NAO robot. If the robot was recently flashed or there is enough space left on the NAO, this step can be omitted. To flash the NAO, please follow the steps described below:

- Prepare the USB stick by formatting it
- Download the flasher tool from the Aldebaran website
  There are two different versions
  Please download the tool according to your OS
- Download the Naoqi image from the Aldebaran website
- Start the flasher tool and use it to copy the image to the USB stick
- If prompted choose factory reset
  This will remove everything from the nao
- Shutdown the NAO
- Put in the USB stick
- Press the chest button for at least five seconds
  The eyes will turn blue
- Progress of reset will be shown by the circle of the ears

B.2 Install ROS

In order to install ROS on the NAO follow the same steps, that are required to install ROS on the development VM (see section A.2).

C Cooperation

The project group met weekly on Wednesday in order to work together on the project task. In the first two periods there were weekly project meetings and in the third period there were two project meetings each week. These project meetings were held to inform the project coordinator of the status of the project and to discuss any issues.

C.1 Division of Tasks

The first two periods of the project consisted mainly of researching the different platforms. This was a shared task for the whole group. Furthermore, each group member had some specific tasks as well, as can be seen in table 2.

	Tasks
Jeroen	ROS development, research
Mara	Report, research
Oliver	ROS development, research
Jonas	Contactperson, research, ROS development
Guy	Research

Table 2: Task division