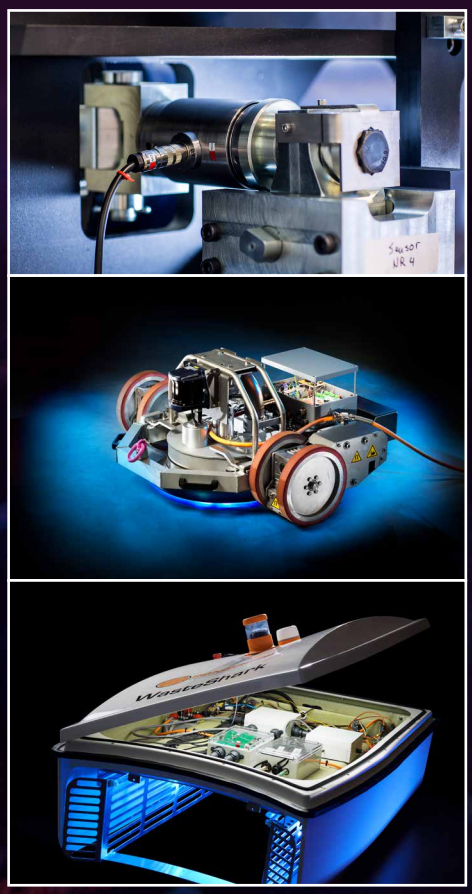
LET'S **OUTSMART** TOMORROW

Autonomous systems for service applications

Generic ROS-based architecture for dirty jobs



NOBLEO TECHNOLOGY

autonomous
intelligent
systems

Autonomous systems for service applications

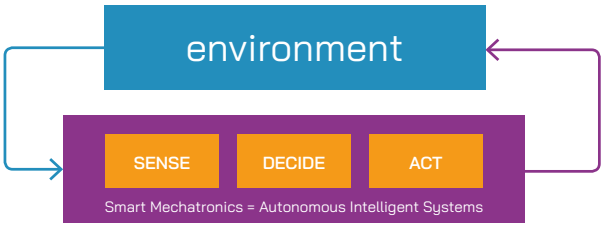
Generic ROS-based architecture for dirty jobs

Autonomous systems are systems that can sense, decide and act in a self governing manner, without a direct human command. Their practical benefit is that they can relieve humans from dirty, dull or dangerous tasks, but in order to do so, they must show dependable, robust and safe autonomous behaviour. Here, a generic architecture for autonomous service vehicles is proposed, based on an open-source standard, ROS. In addition, a number of applications developed by Nobleo are described.

Introduction

Ever since DARPA (the US Defense Advanced Research Projects Agency) decided to challenge engineers to come up with practical, implementable and robust forms of autonomous driving by organising the DARPA Grand Challenge in 2004 (en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2004)), autonomy has more or less become a household word. The event marked the beginning of the transition for autonomous systems from academia to practical applications and industry.

An autonomous system is, by definition, a system that is self-governing, i.e. a system that can receive inputs through sensors, make decisions and act without an active (human) controlling input. The attractiveness of such systems from a user perspective is similar to that of automation: relieving human operators of tedious tasks and putting them in a supervisory role, saving costs and reducing risks while maintaining quality.

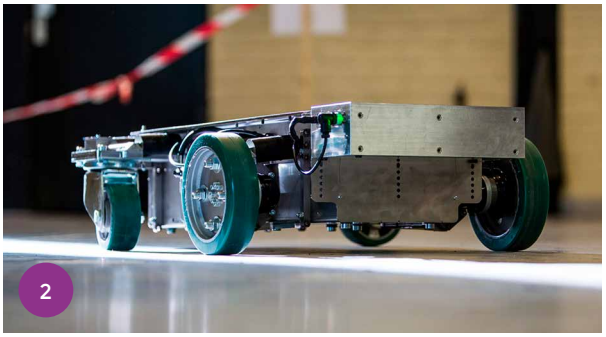


Smart mechatronics. 1

From an application perspective, it makes more sense to characterise autonomous systems by their practical merits, as mentioned above. From an academic perspective, the emphasis is usually more on the technology 'under the hood', which is usually dominated by the software, its architecture and decision-making and learning capabilities. From an engineering perspective, systems that can 'sense' and 'act' are often classical mechatronic systems. Adding signal processing, decision-making and learning capabilities yields 'smart mechatronics', taking the next evolutionary step from basic motion systems to autonomous, intelligent systems (Figure 1).

Software architecture for service vehicles

Service vehicles are (small) autonomous vehicles intended for repetitive, tedious or unsafe tasks, rather than for human transport or autonomous driving in a semi-controlled environment. Cleaning, lawn mowing, painting, logistics, bin-picking, etc., are typical examples of such tasks. Technology trends show that the time is ripe to automate such tasks and that the required technology is affordable.



Clara, a generic platform for autonomous service vehicles.

As a starting point, a generic vehicle hardware platform has been developed at Nobleo (the Clara platform, see Figure 2) for a number of purposes: as a testing ground for the software platform, for reliability testing of the system and components, and for generic service applications. The appropriate operating system/ software development environment was selected by comparing the available options and scoring them on aspects regarded as critical to quality (CTQ); see Table 1. The columns in Table 1 show a number of candidates and the table summarises the aspects, with ROS (Robot Operating System) scoring highest and hence being selected as the platform of choice. ROS is part of an open-source initiative that originally started off around 2000 in the academic world, and which has since become mainstream for many academic and industrial initiatives. This is underlined by the fact that 18 of the 23 contestants in the latest DARPA Robotics Challenge in 2015 used ROS and many of them also used Gazebo (the accompanying simulation and visualisation package).

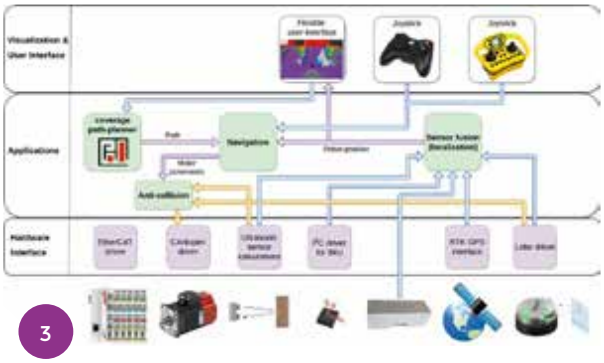
ROS 2.0 is a fundamentally revised concept of ROS with more focus on real-time behaviour and distributed systems (e.g. swarms). Conceptually, this is the preferred version. However, in 2016, when the above comparison was made, major parts of ROS 2.0 were still in the specification phase and little real functionality was available, hence the 0 score in the table. There is a concern among industrial users that an open source concept lacks the rigorous quality assurance and accountability that a commercial product can offer. This is exactly why ROS Industrial was conceived. It consists of a subset of ROS functions that have been rigorously tested and qualified. Since the set of qualified functions in ROS Industrial is smaller than the complete set, it scores lower in the comparison, lacking functionality. Using the open source version of ROS is in our view a manageable risk, as that there is a lot of interest in the ROS Industrial initiative (also actively supported by Nobleo), which will therefore sooner or later seamlessly replace the ROS functions.

Critical to Quality (CTQ)	In-house development	Matlab Simulink	ROS	ROS-Industrial	ROS 2.0	Orocos	Microsoft Robotics Developer Studio
Time-to-market, Development effort	–	+	+	0	0	0	–
Ease of use (simulation, diagnostics, debugging)	0	+	+	+	0	+	+
Standardised/traction solution	-	+	+	0	0	–	–
Software costs	+	–	+	+	0	0	0
Hardware costs	+	0	+	0	0	0	0
Quality assurance, robustness	0	+	0	+	0	0	+
IP ownership	+	–	0	0	0	0	0
Sum total	1	2	5	3	0	0	0

Table 1. Decision matrix for operating system selection.

SLAM

One of the first and foremost functionalities in a service vehicle is that it needs to 'know' where it is. Simultaneous localisation and mapping (SLAM) functionality needs to be embedded in the vehicle's architecture, of course independently of the relevant sensors used, such as GPS, odometers, range finders, lidar, radar, etc. A versatile architecture allows the physical sensor to be 'abstracted' into an information source with its own driver and allows effective 'fusion' of data from numerous sources into one common internal 'world representation'. Here too, ROS, as a communication platform provided with an extensive library of functions and building blocks, accelerates this development. Figure 3 shows the generic architecture for Nobleo's service vehicles.



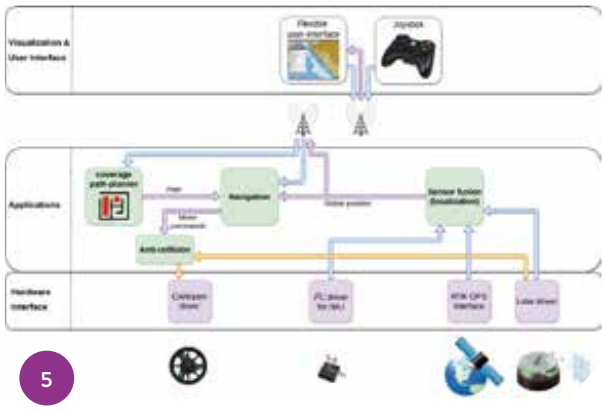
Generic architecture for an autonomous service vehicle.

Particularly the breakdown into functional modules, the definition of the interfaces and the layering make it unique. The upper layer shows hook-ups for different kinds of joysticks; either a cost-effective consumer type (game controller) or a more professional, heavy-duty type can be used. On the sensor/actuator side, numerous options are available. The Accerion sensor, an 'optical odometer' that can achieve high precision by automatic drift calibration, has been extensively tested in the vehicle. Other common modalities are Real Time Kinematic (RTK) GPS (high precision GPS) and Inertial Motion Units (IMUs).

Application: WasteShark

One of the first spin-offs from the generic architecture is a system incorporated in a boat designed to gather waste from harbours: the WasteShark from the company RanMarine. Figure 4 shows the WasteShark in action. Although it looks very different from Clara, the software is practically identical, allowing the re-use of extensively tested software components from Clara (Figure 5).

An aquatic drone: RanMarine's WasteShark equipped with Nobleo's ROS stack and SLAM architecture.

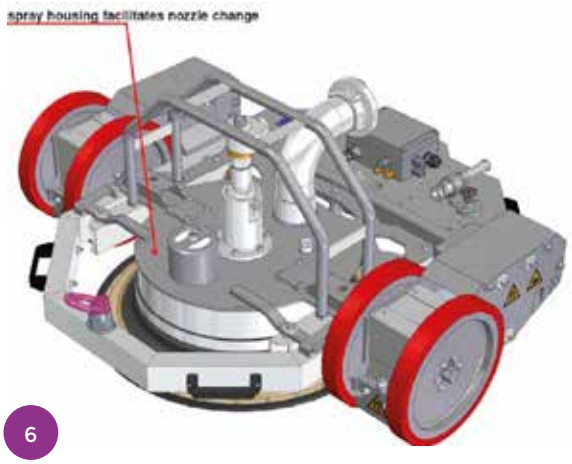


WasteShark architecture: the difference with the generic architecture is in the The architecture of the cleaning robot: 'same story', only different sensors and actuators. interfacing with sensors and actuators, while the core of the software is maintained. Note also the radio connection between the two separable top layers

Generic ROS-based architecture for dirty jobs

Application: Industrial cleaning

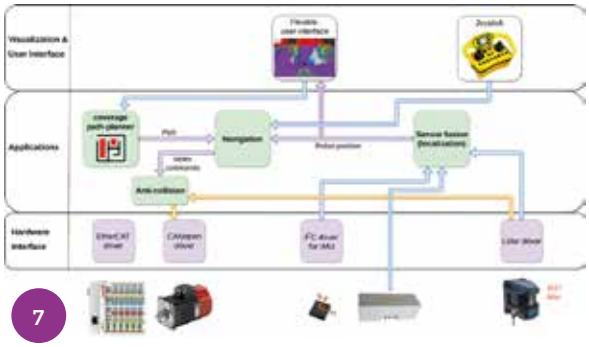
Cleaning in general and industrial cleaning in particular is a dirty, dull and dangerous job. In Nobleo's vision, this is one of the areas where there is the most 'urgent' demand for robotisation and autonomous service vehicles. There are several initiatives, from large OEMs to start-ups, that have designed dedicated cleaning and inspection robots, but motion control is often confined to manual control and hand-held consoles requiring a human in the loop. In the case of (petrochemical) tank cleaning and inspection, the call for 'no man entry' is leading to the banning of human entry into tanks, making autonomous 'meandering' of cleaning and inspection devices practically mandatory.



6

A cleaning crawler for steel-walled tanks; feeder and return hosing are omitted here, wheels are magnetic.

Figure 6 shows a typical prototype crawler now being tested for internal and external cleaning of ferro-magnetic tanks. It consists of a high-pressure cleaning system (feeder hoses omitted here) and magnetic wheels. In Figure 7, the differences with respect to the generic architecture are once again at the device interface level, not in the core of the system. This allowed for quick and cost-effective prototyping of the crawler.



7

The architecture of the cleaning robot: 'same story', only different sensors and actuators.

Autonomous systems roadmap

Service vehicles still have a long evolution ahead of them and at Nobleo we have only just crossed the border from remote control to 'situational awareness' and limited autonomy, i.e. SLAM, as depicted in the widely shared development roadmap of autonomous systems and service vehicles in Figure 8. There is still a long way to go in the development of robust industrial autonomous systems, i.e. smart mechatronics.

8

Development roadmap of autonomous functionality for service vehicles.

Autonomous systems for service applications

Generic ROS-based architecture for dirty jobs

NOBLEO TECHNOLOGY

autonomous
intelligent
systems

Heggeranklaan 1 • 5643 BP Eindhoven • the Netherlands • +31 (0) 40 – 82 00 180 • info@nobleo.nl • **www.nobleo-technology.nl**