USING AUTOMATIONML FOR SYSTEM DECOMPOSITION, DOCUMENTATION AND CONTROL REPRESENTATION

As an Industry 4.0 case study, the system design of a mobile robot is presented for use in the RoboCup@Work competition. RoboCup@Work is an international competition, mainly for university students, that aims to bring research and education into the field of industrial automation. RoboHub Eindhoven, a student team at Fontys University of Applied Sciences, has built a custom-made robot for competing in RoboCup@Work. To make the best possible robot design and obtain a good system overview, a model-based breakdown was created in AutomationML, a sophisticated tool for system decomposition, documentation and representation of the control algorithm sequences.

ESMAEIL NAJAFI

Introduction

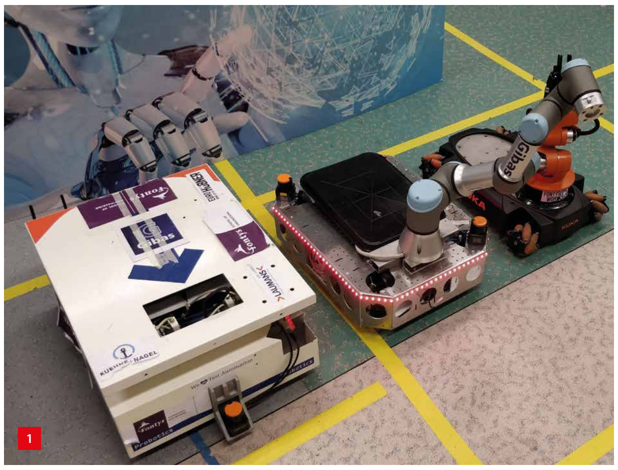
Industry has been making a switch from traditional mass production with people to a more modular production set-up that is automated. This is called the fourth industrial revolution, or Industry 4.0, and creates new challenges concerning the implementation of such a production set-up. Problems include object handling with differently-sized objects, computer vision to detect and scan objects, mobile platforms that can move objects around in the factory, and robot path planning for generating efficient routes while avoiding obstacles.

The international RoboCup competition was set up for tackling robotics challenges on many different levels and in various leagues, where the RoboCup@Work league [1] focuses on industry and aims to stimulate research into the aforementioned Industry 4.0 challenges. The underlying vision is to foster research and development that enables use of innovative mobile robots equipped with advanced manipulators for current and future industrial applications, in which robots cooperate with human workers on complex tasks ranging from manufacturing, automation and parts handling, to general logistics. The competition aims to simulate a factory environment, in which teams use a mobile robot that is equipped with a robotic arm and vision cameras to autonomously drive around and complete tasks.

This article presents an implementation of model-based design (MBD) for the RoboCup@Work competition as an Industry 4.0 case study. AutomationML [2] is the tool used for the documentation of system decomposition. By decomposing the challenge defined in the RoboCup@Work competition into a unified model, a breakdown of this challenge could be made. The decomposed model could be used to generate a systematic, high-level representation of the mobile robot for the competition tasks.

Model-based design

The RoboCup@Work competition covers three main tasks: navigation, manipulation and transportation. The robot (Figure 1) is subject to specific size restraints, because the competition is using a small-scale environment.


1
RoboCup@Work line-up of the RoboHub Eindhoven student team [3].

AUTHOR'S NOTE

Esmaeil Najafi obtained his Ph.D. in Systems and Control in the field of Robotics and Mechatronics at Delft University of Technology, Delft (NL). Then he joined Eindhoven University of Technology, Eindhoven (NL), as a post-doctoral researcher. Now he is a faculty member of the Mechatronics department at Fontys University of Applied Sciences, Eindhoven.

e.najafi@fontys.nl
www.fontys.nl

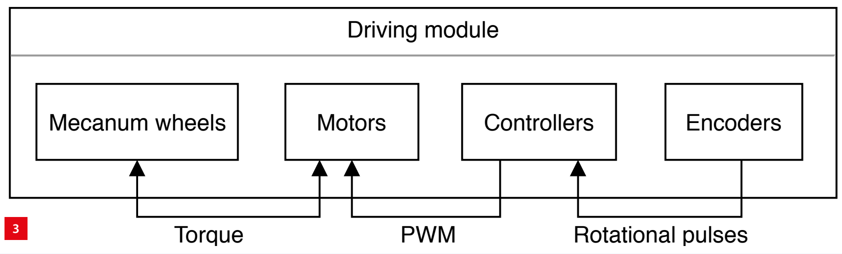
Model-based design is a methodology to find the solution to a problem by creating a proper model in four steps: modelling the system, designing a controller, simulating the controller, and deploying the controller. As such, the accuracy of the model, as the basic part of the MBD, is very important, because the controller design will be based on this model. Hence, finding an accurate model helps to design an efficient controller. In the implementation stage, there are some tools available to simplify the direct application of the control algorithm to the hardware, without having to convert to low-level hardware codes such as PLC, CNC or FPGA.

System structure

For designing the full system, the robot had to be placed in a virtual execution environment, like the Robotic Operating System (ROS) [4], with the static and dynamic variables that act on it externally. The system modelling language (SysML [5]) diagrams have been used to demonstrate the system structure, looking from the high-level to the low-level design.

Figure 2 illustrates the top level of the system. The SysML diagram offers all the functionality required for defining all parts of the system and provides a high-level decomposition of the robot model. When the mobile platform had reached the late design stage, the SysML diagram offered insights and could act as a basis for prototype building as well as hardware selection.
During the design phase, the requirements of the system were established, followed by the two main building blocks, i.e. the structure of the system and the behaviour in different stages of execution. The requirements block presents the system requirements and the general dependency between them in the design of the mobile platform. The two main building blocks were designed in parallel when the concept of the system had been defined, as the structure and behaviour are linked together, with respect to each system module.

The structure block presents the different modules that compose the system. In the current system, the focus was on the hardware comprising mechanical and electronic components. The structure block additionally presents



3

Driving module that contains the structural details of the hardware used and the interfaces that allow the module to connect to other modules.

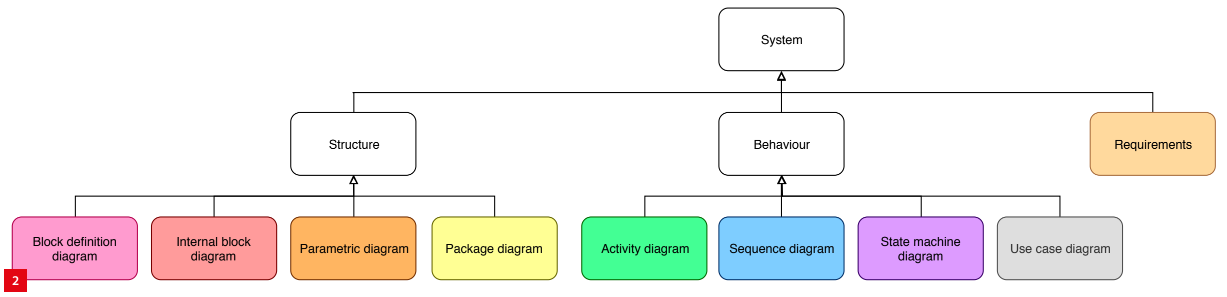
the interfaces of each system module and the way they are connected to the other modules through the known communication, data, interface or physical protocol.
Based on a more general view of the structure of the system, the block definition diagram presents the dependencies of each module with respect to the roots of the system. The behaviour block presents multiple views of the execution flow of the system. The views of the behaviour can be processed individually, but the full execution loop is created by combining all the behaviour diagrams.

Simulation in AutomationML

One of the structural elements within the robotic system is the Driving module (Figure 3), which was used for implementation in AutomationML. The module contains the structural details of the hardware used and the interfaces that allow the module to connect to other modules, which are the electronics and mechanical components that create the drive train of the mobile platform.

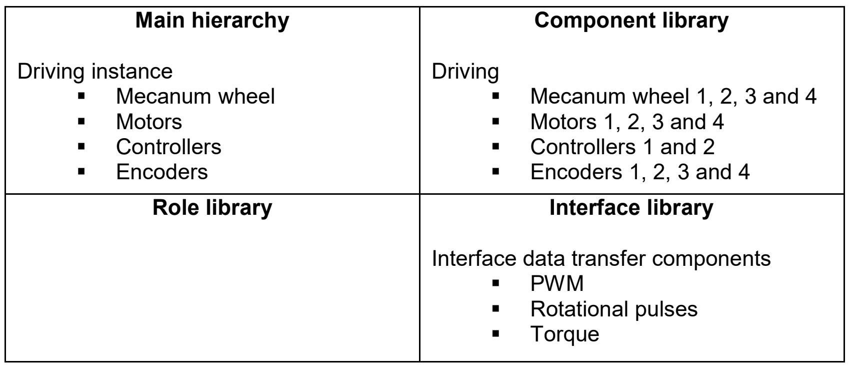
In the Driving module, the components are not fully defined, but their requirements and interfaces are presented in the internal structure of the module. For this module, it was known that it contains four mecanum wheels, actuated by four electric motors that in turn are controlled by a pulse-width modulation signal from motor controllers. On the motors, relative encoders are mounted offering a pulse-type signal used by the motor drivers for calculating the velocity and translation of the mobile platform.

The AutomationML file consists of four blocks, as shown in Figure 4. The block in the top left corner describes the main



2

System structure: a high-level decomposition of the robot model.



Main hierarchy	Component library
Driving instance ▪ Mecanum wheel ▪ Motors ▪ Controllers ▪ Encoders	Driving ▪ Mecanum wheel 1, 2, 3 and 4 ▪ Motors 1, 2, 3 and 4 ▪ Controllers 1 and 2 ▪ Encoders 1, 2, 3 and 4
Role library	**Interface library**
	Interface data transfer components ▪ PWM ▪ Rotational pulses ▪ Torque

4

The main layout of the AutomationML file for the Driving module.

hierarchy of the RoboCup@Work project, featuring the mechanical, software and electrical parts. The other three blocks are the libraries for the main hierarchy, namely:

- Components library, containing all components that were used in the RoboCup@Work project.
- Role library, containing every role of the modular mobile platform.
- Interface library, containing the communication signals and connections between the parts of the modular mobile platform.

In addition, Figure 5 partially represents the implementation of the Driving module in the AutomationML environment.

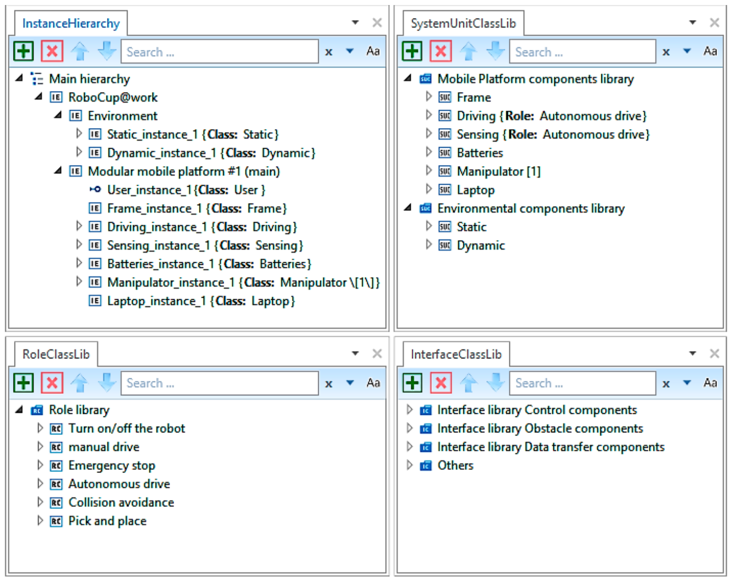
Discussion and conclusions

During the process of the robot design, system engineering is often one of the challenging parts. This is mainly because there are several experts involved from different backgrounds working on several modules, which have to be assembled into one complete system. Difficulties can arise, especially in case of complex systems developed by several teams. If one calculation is wrong, every submodule must change. Theoretically, this should not be a problem, but in practice it will take hours to inform every department about the change, not to mention the time and hence costs to adjust every module to fit in the new design.

By using AutomationML as a standard format for the project, all information is placed in one structure, making it easy for every department to check it at any time. An additional benefit of AutomationML is the fact that all modules are interlinked. If one aspect in a module changes, AutomationML automatically changes all dependent modules accordingly. This will save time and costs in Industry 4.0 projects.

In this work, AutomationML has been used to separate elements of behaviour and components of the mobile platform and RoboCup environment, to create a clear overview. By adding interfaces to connect every behaviour state with the relevant components and placing every component in an orderly and proper way inside the main hierarchy, every team member from every department can see their module, connected to the other relevant modules. This helps every member to get the relevant information they need to construct a fitting module and even solve future problems in other departments. For example, an electric malfunction inside the drive train can be prevented at an early stage by a member creating the sensing part of the platform.

As studied in the Industry 4.0 framework with an example of RoboCup@Work, using model-based methodology considerably decreases the number of project communication mistakes and unnecessary handling when one component is updated, adjusted or even either added or deleted. For example, in this case the handling time required for updating a component was reduced by almost 50%. So, project time and costs can be saved, providing more budget for research allowing a better design. If something is changed in the system, it must be guaranteed that correlating components keep working properly. It can be concluded that using AutomationML leads to an appropriate decomposition of robotic systems.



InstanceHierarchy
Main hierarchy
RoboCup@work
Environment
Static_instance_1 {Class: Static}
Dynamic_instance_1 {Class: Dynamic}
Modular mobile platform #1 (main)
User_instance_1 {Class: User}
Frame_instance_1 {Class: Frame}
Driving_instance_1 {Class: Driving}
Sensing_instance_1 {Class: Sensing}
Batteries_instance_1 {Class: Batteries}
Manipulator_instance_1 {Class: Manipulator \[1\]}
Laptop_instance_1 {Class: Laptop}

SystemUnitClassLib
Mobile Platform components library
Frame
Driving {Role: Autonomous drive}
Sensing {Role: Autonomous drive}
Batteries
Manipulator [1]
Laptop
Environmental components library
Static
Dynamic

RoleClassLib
Role library
Turn on/off the robot
manual drive
Emergency stop
Autonomous drive
Collision avoidance
Pick and place

InterfaceClassLib
Interface library Control components
Interface library Obstacle components
Interface library Data transfer components
Others

5

Implementation of the Driving module in the AutomationML environment.

REFERENCES

[1] S. Alers, D. Claes, J.-D. Fossel, D. Hennes, K. Tuyls, and G. Weiss. "How to Win RoboCup@ Work?-The Swarmlab@ Work Approach Revealed", *RoboCup*, pp. 147-158. 2013.
[2] R. Drath, A. Luder, J. Peschke, and L. Hundt. "AutomationML-the glue for seamless automation engineering", *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 616-623, 2008.
[3] *www.robohub-eindhoven.nl*
[4] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng., "ROS: an open-source Robot Operating System", *ICRA workshop on open source software*, 3 (2), p. 5. 2009.
[5] T. Weilkiens, *Systems engineering with SysML/UML: modeling, analysis, design*, Elsevier, 2011.