Exploiting the FAMOUSO Middleware in Multi-Robot Application Development with Matlab/Simulink

Michael Schulze and Sebastian Zug
Otto-von-Guericke University of Magdeburg
Faculty of Computer Science
Department of Distributed Systems (IVS)
{mschulze, zug}@ivs.cs.uni-magdeburg.de

Francisco Campos and Fernando Carreira
Instituto Sup. de Engenharia de Lisboa
Polytechnic Institute of Lisbon
{fcampos, fcarreira}@dem.isel.ipl.pt

ABSTRACT

We describe a framework for the development of distributed systems combining real and virtual components, sensors and actuators. We show the benefits of our approach for the development and validation of multi robot applications. Based on our middleware, which provides a flexible communication for distributed systems, virtual and real components are seamlessly exchangeable during different development steps. This modularity and compatibility allows appropriate adjustments for design, rapid prototyping and examination as soon as an opportunity to reduce the hardware effort for large scenarios.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed applications; D.2.11 [**Software Architectures**]: Middleware; D.2.12 [**Interoperability**]: Distributed objects

General Terms

Design, Experimentation, Languages

Keywords

Middleware, Publish/Subscribe, Multi-Robot, Development Framework, Embedded Devices

1. INTRODUCTION

The development of (multi) robot systems demands the integration of different technologies, namely, mechanics, electronics, hardware and software. The design cycle consists of a number of tasks usually sequenced as mechanical design, instrumentation and programming. As in other complex mechatronic systems, this process benefits from the early validation of the different levels of design. The control algorithms have to match to the mechanical design. For example the position of a distance sensor is an important parameter for the collision avoidance algorithm. A software framework to support robot design should therefore include capabilities for mechanical modeling and simulation, sensor and actuators specification, hardware interface and communication, control engineering and programming. Currently, due to the disparate nature of these requirements, the development of robotic systems is usually carried out in a fragmented fashion, making use of different software tools and programming languages that do not allow for automatic integration among them. This results in an extra burden for the designer.

Currently, a number of development environments are available, such as the Microsoft Robotic Studio [4] and the open-source Player Stage Project [1], which attempt to integrate the several aspects of a robotic system development process. These platforms support the simulation of multi-robot behaviour in virtual environment, based on realistic physical engines that reproduce the mechanical behaviour of the robots and offers the simulation of the most common sensors and actuators. Within these development environments, the same control schemes may be applied to real hardware, allowing for the control of real robots which can run in parallel with their Virtual Reality representations. Both tools are reliant on TCP/IP networks and in case of the Robotic Studio upon a .NET framework in the background. Accordingly a powerful hardware is required and the kind of networks is limited [6].

Even though these programming environments provide tools for robot simulation and control, they supports well known hardware platforms like the PIONEER robotics or the Lego Mindstorm system mainly. However, on a crucial position within the integration, a problem is the communication among software components of the robotic system that may exist in different hardware modules, run on different operating systems and have available different communication protocols. Due to this fact, the mentioned development frameworks and other works like [5] support only a limited number of well known hardware platforms.

This problem can be tackled by an appropriate middleware layer that guarantees a seamless distribution of information among the interested software components within a hardware network. Thus all components are able to communicate with each other independently of their development state. In this paper, we introduce the FAMOUSO (**F**amily of **A**daptive **M**iddleware for autonom**OU**s **S**entient **O**bjects) middleware for the combination of virtual, software components with real modules of robotic systems. Simulation is a major tool during the design cycle, since it allows for early validation of design options, iteration of solutions and correction. Along the design cycle, simulated modules should

Middleware '08 Companion, December 1-5, 2008 Leuven, Belgium.

be progressively substituted by real components, which have been properly validated. The proposed framework supports the simulation of software and hardware modules as well as the coexistence of real and simulated components. For an example, the design process starts with the realisation of the mechanical structure of the robot and with the development of control algorithms. This can be done in parallel by running the control algorithms on a PC in a Software-in-the-Loop system. Later the algorithms are downloaded onto the robot. In the same way the sensors can be simulated in a virtual environment first and be replaced by real sensors on the real robot finally. This combination of real and virtual components simplifies the development of high-grade distributed applications. Following, our approach allows the coexistence of simulated and real robots and reduces the hardware effort considerably. The FAMOUSO middleware provides means for information distribution between different platforms like PCs and dedicated micro-controllers. FAMOUSO permits smooth transitions from each development stage of this design cycle, due to its seamless integration of different platforms at runtime. To offer a development tool chain from model design, simulation and code generation we implemented a FAMOUSO interface for Matlab/Simulink.

In this paper, we describe an example scenario that demonstrates the potential of FAMOUSO for robot development. The example shows a running VR model of a robot and its environment on a PC, while another computer runs the control algorithms. This example intends to demonstrate a stage of development where the control algorithms are already running on the physical prototype, while the sensors are simulated in the development framework. This stage of development allows the testing of control algorithms against realistic environment situations and the specification of sensor characteristics and location.

2. FAMOUSO MIDDELWARE

FAMOUSO [11] is an event-based publish/subscribe middleware and the successor of the COSMIC (**CO**operating **SM**art dev**IC**es) middleware. On the level of exchanged events they are fully compatible. The main concepts are described in [8, 7]. Additionally, FAMOUSO provides means for adaption e.g. to a specific platform. The middleware offers an event-based communication model according to the publisher/subscriber concept. FAMOUSO is especially designed to allow cooperation between smart sensors and actuators on different hardware platforms ranging from 8-bit micro-controllers up to 32-bit PC/Workstations and interaction over a broad variety of communication media like Controller Area Network (CAN) [10], 802.15.4 [13], AWDS [2] and UDP-MultiCast.

In FAMOUSO an event is a programming abstraction and the carrier of the exchanged information. A FAMOUSO event consists of three different parts:

1. a subject, represented by a 64 Bit unique identifier (UID) that describes the content,
2. the content or payload itself for instance the value of a distance measurement and
3. additional attributes (e.g. sensor position, context, quality) which are optional.

The subject of a message has a meaning across multiple networks. This is exploited for filtering at the network boundaries. If a certain subject is not subscribed outside a specific subnet, it is not propagated by the respective gateway.

Generally, events may arise in two different ways. Firstly, an event is spontaneously generated by the hardware e.g. caused by detection on a sensor interface. This means the physical environment is the stimulus of an event. Secondly, an event could periodically initiated by a clock to sense a change of a variable or a state within the system.

FAMOUSO uses event channels as abstraction for event transfers. An event channel has the same subject as the corresponding event. The event is published by pushing it to the according event channel. In case of subscription, the application gets a notification by the event channel if an event occurs. The programming abstraction event channel is introduced to map the UID of an event to specific network addresses and therefore it hides the heterogeneity of the different network architectures by providing a global addressing scheme. Furthermore the event channel is used for network resource allocation - for instance a part of the bandwidth. Depending on temporal constraints or the importance of the event, the event is classified into three different quality levels which are hard real-time (HRT), soft real-time (SRT) and none real-time (NRT).

In FAMOUSO all events are handled by the event channel handler (ECH), which is part of the event layer (EL). The EL is the interface to the application level. Consequently, the publisher uses the EL to send events to event channels and on the other hand the EL provides the subscriber with notifications and supports it by reading events from event channels.

On the application side, the FAMOUSO API is accessible from several languages like C++, Java, Python, C# and also via engineering tools like Labview or Matlab/Simulink [12].

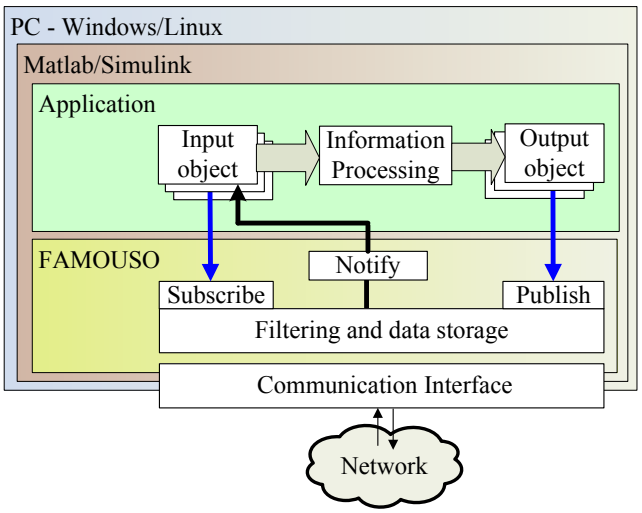

Figure 1: FAMOUSO Matlab/Simulink integration

3. MATLAB/SIMULINK INTEGRATION

Matlab is a computational tool which provides powerful toolboxes addressing the classic and advanced methods in control engineering and signal processing, such as fuzzy and

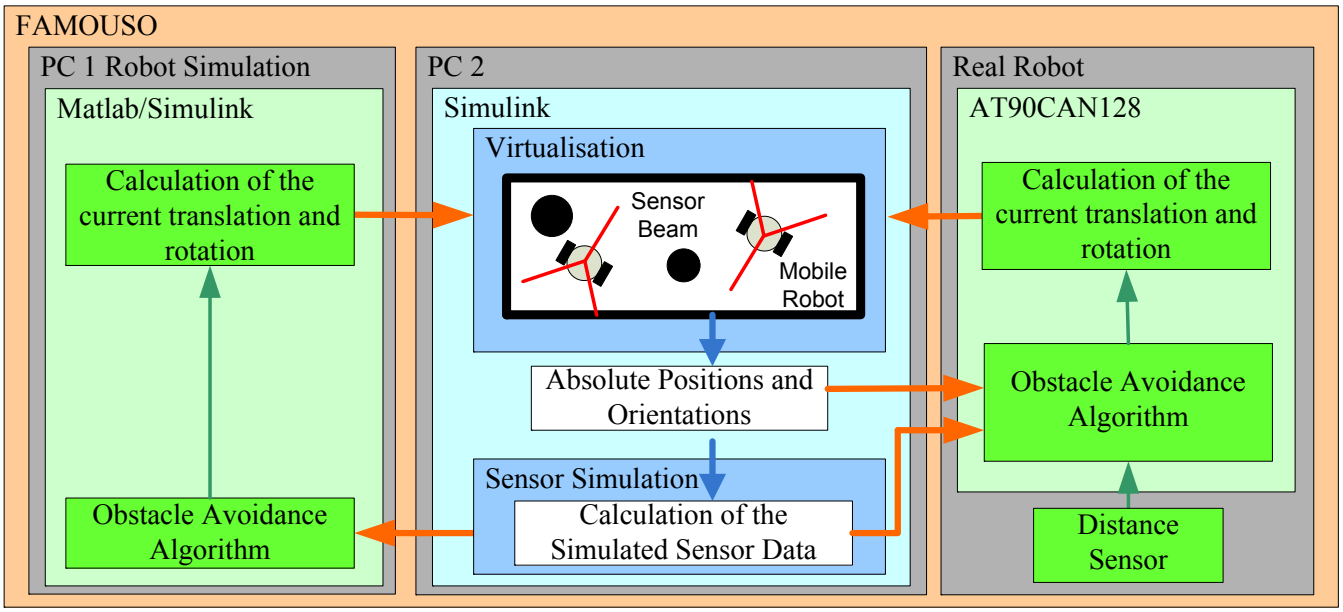

Figure 2: Structure of the demonstration scenario

neuronal network based controllers. Among the branches of Matlab, there are Simulink and the Virtual Reality toolbox, two applications which, when combined, allow for the development of dynamic systems simulations synchronized with a Virtual Environment. The integration of FAMOUSO into Matlab/Simulink offers the broad functionalities of this development suite combined with a transparent access to all relevant information through FAMOUSO, simplifying the development process seriously [9].

Instead of using the plain communication facilities of Matlab/Simlink like direct TCP/IP the application uses FAMOUSO for its communication. The Matlab/Simulink FAMOUSO interface offers the uniform FAMOUSO API as shown in Figure 1. In Matlab/Simulink each event channel is represented by an instance of the input or output class that contains the respective subject, administration variables and a configurable predefined number of the last received events.

The notification service that indicates new occurred events is implemented in two kinds – polling and callback. The callback functionality is typically used by Matlab applications. In contrast to Matlab, Simulink models run cyclically in predefined periods. In order not to break these polling is applied in each cycle, first. The polling function fetches all received events and refreshes the data storage of all input channel objects. Due to the storage feature, the user has the possibility to define in which way the events of the last periode have to be handled. Thus the return value of the access methods of each input channel object can be either ”newest”, ”mean” or ”sum”. In Section 4 we exploit that in our example scenario and that shows the importance of that feature.

4. EXAMPLE SCENARIO

As an example, we use a typical robot application and development challenge – the influence of the sensor capabilities for collision avoidance. Figure 2 shows our scenario comprising of several simulated and/or real mobile robots that navigate in a 2-dimensional environment with obstacles. According to a robot's configuration it is visualized in the virtual representation. The sensor information is calculated in a simulation module based on robots position. The collision avoidance algorithm runs in small applications on a PC for instance as a Matlab/Simulink script. On the other hand a real mobile robot with sensors and actuators does real measurements and movements instead of calculating them. To allow the perception of virtual elements by the real robot, it consumes virtual sensors information additionally. Hence, the mobile system ”views” both the virtual and the real environment, however, for the application the origin of information is transparent.

The information are exchanged via FAMOUSO. The values of the simulated distance sensors are published to an event channel, which is subscribed by the virtual and real robots. Each robot uses the ”newest” sensor values only. Hence, the real robot behaves on the values of the simulated sensor information and real measurements, accordingly. The virtualization and simulation modules obtain the movement from another FAMOUSO event channel in a ”sum” fashion.

The scenario provides an environment for testing movement algorithms based on sensor range and orientation. The closed loop of this interaction is illustrated in Figure 2 where a real robot and simulated systems interact. A lot of other combinations and configuration are imaginable with our flexible approach.

For the visualisation the virtual reality toolbox of Matlab/Simulink was used, enhanced by the VR Studio, a package developed by Campos et al. [3]. This toolbox allows for the creation of dynamic virtual environments in simple fashion. Elements of the simMechanics toolbox ensure the realistic physical simulation of the robot's behavior.

5. CONCLUSIONS

In this paper, we describe the integration of our middleware FAMOUSO within the Matlab/Simulink environment.

We show the resulting benefits of using FAMOUSO for the development of distributed systems. FAMOUSO's network transparent and flexible communication allows the combination of real and virtual components in each step of the development process. Exploiting the possibilities of FAMOUSO enables an easy and seamless integration of different components on different systems.

6. ACKNOWLEDGEMENT

This work is part of the project DECOMOR which has been supported by the DAAD and GRISCES in a German-Portuguese collaboration scheme [D/06/12913] and by the MiNEMA (Middleware for Network Eccentric and Mobile Applications) Scientific Programme of the ESF (European Science Foundation).

7. REFERENCES

[1] The player project. online, `http://playerstage.sourceforge.net`.

[2] AWDS project homepage. online, `http://awds.berlios.de`, 2008.

[3] F. M. Campos, F. Carreira, and J. M. F. Calado. VR Studio as an auxiliary tool for technical education. In *Proceedings of Engenharias 2007*, Covilhã, Portugal, November 21–23 2007.

[4] M. Corporation. Microsoft robotics studio. online, `http://msdn.microsoft.com/en-gb/library/bb881626.aspx`.

[5] G. Dudek and R. Sim. Robodaemon - a device independent, network-oriented, modular mobile robot controller.

[6] J. Jackson. Microsoft robotics studio: A technical introduction. *IEEE Robotics & Automation Magazine*, 14:82 – 87, Dezember 2007.

[7] J. Kaiser and C. Brudna. A Publisher/Subscriber Architecture Supporting Interoperability of the CAN-Bus and the Internet. In *2002 IEEE International Workshop on Factory Communication Systems*, Väesteras, Schweden, August 28–30 2002.

[8] J. Kaiser, C. Brudna, C. Mitidieri, and C. Pereira. COSMIC: A middleware for event-based interaction on CAN. In *9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2003)*, Lisbon, Portugal, September 2003.

[9] J. Kaiser, M. Schulze, S. Zug, C. Cardeira, and F. Carreira. Sentient objects for designing and controlling service robots. In *Proccedings of IFAC'08*, volume 17th International Federation of Automatic Control World Congress, Seoul, Korea, July 6-11 2008.

[10] Robert Bosch GmbH. *CAN Specification Version 2.0*. 1991.

[11] M. Schulze. FAMOUSO project website. online, `http://famouso.sourceforge.net`, 2008.

[12] M. Schulze and S. Zug. Using COSMIC – A real world case study combining virtual and real sensors. In *Proceedings of the 5th Minema Workshop on Middleware for Network Eccentric and Mobile Applications*, Magdeburg, Germany, September 11–12 2007.

[13] ZigBee Alliance. *ZigBee Specification - IEEE 802.15.4*. 2003.