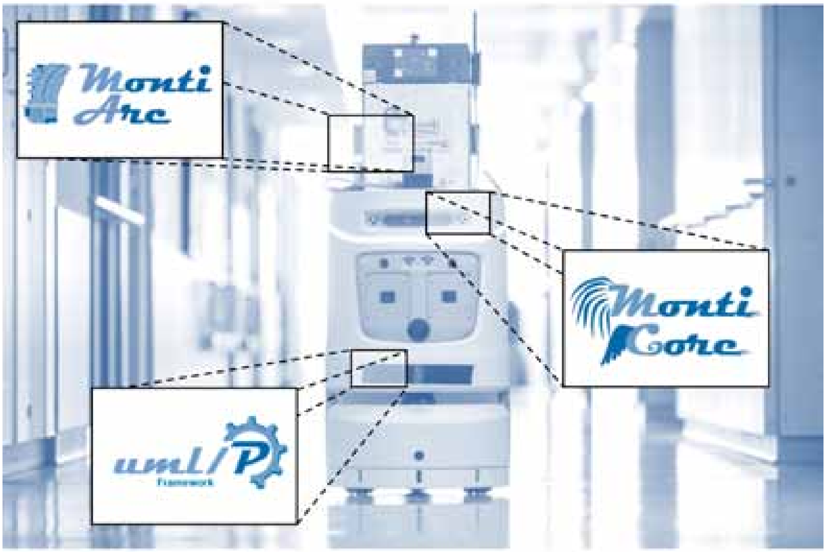
Kai Adam, Arvid Butting, Robert Heim,
Oliver Kautz, Jérôme Pfeiffer,
Bernhard Rumpe, Andreas Wortmann

Modeling Robotics Tasks for Better Separation of Concerns, Platform-Independence, and Reuse

Results of the iserveU Federal Research Project



Aachener Informatik-Berichte,
Software Engineering Band 28

Hrsg: Prof. Dr. rer. nat. Bernhard Rumpe

Aachener Informatik-Berichte, Software Engineering

herausgegeben von
Prof. Dr. rer. nat. Bernhard Rumpe
Software Engineering
RWTH Aachen University

Band 28

Kai Adam, Arvid Butting, Robert Heim, Oliver Kautz, Jérôme Pfeiffer, Bernhard Rumpe, Andreas Wortmann
RWTH Aachen University

Modeling Robotics Tasks for Better Separation of Concerns, Platform-Independence, and Reuse

Results of the iserveU Federal Research Project

Shaker Verlag
Aachen 2017

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Printed in Germany.

ISBN 978-3-8440-5319-7
ISSN 1869-9170

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • e-mail: info@shaker.de

This research has partly received funding from the German Federal Ministry for Education and Research under grant no. 01IM12008C. The responsibility for the content of this publication is with the authors.

Abstract

Deploying robotics applications requires expertise from multiple domains, including general software engineering and the application domain itself. Consequently, successful robotics applications are developed by teams of software experts, robotics experts, and application domain experts. The conceptual gap between application domain challenges and implementation domain solutions gives rise to accidental complexities from solving problem domain challenges with programming domain details. This complicates development and may lead to failure. Domain and robotics experts are rarely software engineering experts. Their involvement into the software engineering of reusable robotics applications requires that they become software experts or that implementation details can be abstracted.

Model-driven engineering reduces the conceptual gap by leveraging models to primary development artifacts. Models usually are more abstract than programming language artifacts and enable to use robotics vocabulary or application domain vocabulary. This supports domain experts in formulating solutions in established known vocabulary. Model transformations can embody the software engineering expertise required to translate such models into robust and reusable programming language solutions. Different target technologies can be addressed by different transformations, which decouples logical robotics tasks from heterogeneous platforms and ultimately produces deployable solutions. Proper separation of expert concerns is crucial to enable transformations and ultimately improve engineering of robotics applications. Component-based software engineering has proven useful for the integration of domain-specific solutions and system extensibility. Here, software components encapsulate functionality in reusable black boxes. That these components usually are programming language artifacts gives rise to accidental complexities again. Models of architecture description languages lift components to the model level. Component models can serve as building blocks of complex systems and can be translated into implementations for different target technologies as well. Combining model-driven engineering with component-based software engineering enables to provide robotics experts and domain experts with appropriate modeling languages, as well as software engineering experts with means to decouple their concerns while ensuring integration of their solutions.

We present a collection of domain-specific languages to describe service robotics applications. They enable formulating domains, tasks, actions, and properties of a robot and its environment free of GPL complexities. Their models are translated into component implementations of a MontiArcAutomaton [RRRW15] software architecture model and interpreted at system runtime. The architecture executes tasks via their transformation to Planning Domain Definition Language models, solves these, and executes the resulting plans with a robotics middleware. Leveraging separation of concerns and abstraction of modeling languages, this reduces the effort of describing robot tasks, facilitates extension

of the system with new components, and decouples logical task solving from the robot platform. This supports integration of domain experts and reuse of infrastructure constituents in different contexts and with different platforms.

Contents