Embedded Systems Specific Requirements for Choreography Modelling Language Design

Nebojša Taušan, M3S - Software Engineering Research Group, University of Oulu, Oulu, Finland
Jouni Markkula, M3S - Software Engineering Research Group, University of Oulu, Oulu, Finland
Pasi Kuvaja, M3S - Software Engineering Research Group, University of Oulu, Oulu, Finland
Markku Oivo, M3S - Software Engineering Research Group, University of Oulu, Oulu, Finland

ABSTRACT

Software companies that develop embedded systems following the principles of service-oriented architecture can anticipate various benefits from choreography modelling. Current choreography modelling languages, however, have a limited applicability in embedded systems development since they are not expressive enough to capture all the choreography-relevant aspects that are typical in this domain. This problem is addressed in this study with the analysis of the needs in embedded systems domain for choreography modelling language. The analysis was guided by design science and relied on expert interviews, company-specific documents, relevant scientific literature and the experts’ evaluation of the redesigned choreography modelling language. The main results of the analysis presented in this paper are a) design requirements addressing the limitations of choreography modelling languages for embedded systems development and b) proposals for modelling language implementation technologies. The derived design requirements indicate on choreography-relevant embedded systems development aspects such as the constraint-based access and real-time execution. Modelling language implementation technology proposals include Eclipse modelling framework and Sirius. The feasibility of these results is evaluated by redesigning an existing choreography modelling language based on the derived design requirements, implementing a prototype editor for the redesigned language and by evaluating the redesigned language with experts.

KEYWORDS

BPMN, Choreography, Design Requirements, Design Science, Modelling Language

1. INTRODUCTION

The size and the complexity of today’s embedded systems (ES) have led to the growing adoption of model-driven engineering (MDE) (Liggesmeyer & Trapp, 2009) and service-oriented architecture (SOA) in their development (Cannata, Gerosa, & Taisch, 2008; Gilart-Iglesias, Macia-Perez, Marcos-Jorquera, & Mora-Gimeno, 2007). MDE, as an overall engineering approach, relies on practices such as modelling, model transformation and code generation to facilitate the software development practices (Schmidt, 2006). SOA relies on service interactions as a means of achieving system goals in a flexible manner (Scholz et al., 2009). A specification of service interactions can therefore be seen as an important development artefact which is commonly modelled form at least two viewpoints – choreography and orchestration (Dijkman & Dumas, 2004; Peltz, 2003).
Choreography and orchestration models capture different details relevant to the specifications of service interactions. The choreography captures the interactions between participants, which represent autonomous management authorities (e.g., companies, organization units or development teams) whose services are interacting to achieve a common goal (Barros, Dumas, & Oaks, 2005). The orchestration focuses on the service interactions needed to achieve the goal of a single participant (Dijkman & Dumas, 2004). These two viewpoints can therefore be seen as complementary. Several studies, however, indicate growing interest in choreography in the ES domain.

The basis for the growing interest in choreography in the ES domain is set by AESOP research project (IMC-AESOP, 2015). AESOP project results favored choreography viewpoint for setting up the main building blocks for industry automation systems since the information it conveys can contribute to system performance (Starke, Kunkel, & Hahn, 2013). Following these results, Kothmayr, Kemper, Scholz, and Heuer (2015) presented an approach for executing choreography scenarios with real-time guarantees that can result in faster execution of industrial systems. The growing interest in choreography, as a generic abstraction of collaboration among multiple services, is also present in the development of middleware products for future internet applications (Issarny et al., 2011). Additionally, the study by Kaur et al. (2013) highlights on the scalability potential of choreography in large-scale control systems domain.

The choreography viewpoint is commonly specified using Choreography Modelling Language (CML), and once specified, it can address various misalignments in the development process. Some of these misalignments include duplication of work, delays in the development and a loss of opportunity from the parallelization of work (Taušan et al., 2014). One drawback is that currently used CMLs have limited applicability in ES development since they are not expressive enough to capture various choreography-relevant development aspects that are typical in the ES domain.

This study goal was to address this drawback by producing the knowledge for the design of a comprehensive CML capable of supporting the choreography-relevant development aspects that are typical in ES development. Based on this goal, the following research question was derived:

**RQ:** How can the expressiveness of a CML be increased to enable its applicability in the embedded systems domain?

The answer to this RQ is formed of two parts. The first part consists of identifying the choreography-relevant development aspects in the ES domain, their articulation as design requirements, and the selection of technologies for language implementation. The second part consists of evaluating the feasibility of the derived design requirements and implementation technologies. The feasibility of the design requirements was evaluated by redesigning an existing CML based on the derived requirements. The feasibility of the selected technologies for language implementation was evaluated by implementing a prototype language editor using those technologies. Feasibility evaluation of the redesigned CML and editor with industry experts is presented in a separate study (Taušan, Lehto, Markkula, Kuvaja, & Oivo, 2016). This separate study identified testing and protocol development, as ES development areas where the CML can be applied, additional design requirements, and practical challenges regarding use of the language. The identified additional design requirements are also presented in this article. It should be noted that in the rest of the text, the terms design and redesign of the CML will be used interchangeably.

This article consists of seven sections. Related work is presented in following section. The data sources and analysis methods used in this study are presented in Research Method section. Results of the analysis are presented in sections Design Requirements and the Implementation Technologies for CML. The redesign of the CML based on the derived design requirements, the implementation using the identified technologies and the evaluation with industry experts is presented in the Feasibility Evaluation section. The Conclusion section concludes the study and presents the future work.
Towards an Integrated Personal Software Process and Team Software Process Supporting Tool
www.igi-global.com/chapter/towards-integrated-personal-software-process/61216?camid=4v1a

On Developing Hybrid Modeling Methods using Metamodeling Platforms: A Case of Physical Devices DSML Based on ADOxx
www.igi-global.com/article/on-developing-hybrid-modeling-methods-using-metamodeling-platforms/123607?camid=4v1a