Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-21 (2016) 439–446

Automatic Transformation of UML System Models for Model-based Error Propagation Analysis of Mechatronic Systems

Kai Ding *, **Thomas Mutzke** **, **Andrey Morozov** *, **Klaus Janschek** *

* *Technische Universität Dresden, Institute of Automation, 01062 Dresden, Germany (e-mail: {kai.ding, andrey.morozov, klaus.janschek}@tu-dresden.de)*

** *Siemens Healthcare GmbH, Röntgenstr. 19-21, 95478 Kemnath, Germany (e-mail: thomas.mutzke@{siemens.com, tu-dresden.de})*

Abstract:
Mechatronic systems consist of heterogeneous components: mechanical parts, hardware, and software. Appropriate models, which describe the mutual physical interaction on common abstract levels, are required. UML is a widely accepted candidate for design and model-based analysis of the mechatronic systems. For the error propagation analysis on system level we have introduced a stochastic dual-graph error propagation model. This model captures control and data flow aspects of the system and allows the computation of various reliability metrics using discrete time Markov chain models. In our recent case-studies, UML Activity Diagrams have been used as baseline models. However, the transformation process was not fully automatic. This process is not so straightforward, despite the obvious structural similarities of the activity diagrams and our error propagation models. This article presents a new fully automatic method for the transformation of annotated activity diagrams. The transformation algorithm is described in detail with formal set-based mathematical notations. The article addresses both theoretical and technical sides of the problem. The method is demonstrated as a part of a complete analytical workflow in the frame of a mechatronic case study.

Keywords: UML; Activity Diagrams; Control Flow; Data Flow; Error Propagation Analysis; Dual-graph Error Propagation Model; Medical Patient Table;

1. INTRODUCTION

Mechatronic systems consist of heterogeneous components (mechanical parts, hardware, software) with various mutual physical interactions (see Fig. 1). The goal of mechatronic system design is to ensure a proper and coordinated operation of these elements within a feedback structure under all possible operational conditions. One of the big challenges of mechatronic systems design is the use of appropriate models, which describe the mutual physical interaction on common abstract levels.

The unified modeling language (UML, see OMG (2015b)) and its extension for systems engineering, SysML (see OMG (2015a)), are widely accepted candidates for heterogeneous system modeling. In particular, UML Activity Diagrams (ADs) are convenient as a baseline model for data error propagation analysis (Morozov and Janschek (2014)). ADs display the sequence of actions showing the workflow from a starting point to the finish point and the data transfer between executable nodes. An example of an AD of a medical mechatronic system is shown in Fig. 7 and described in Section 4.

More than five years our research group for model-based system analysis is focused on stochastic error propagation

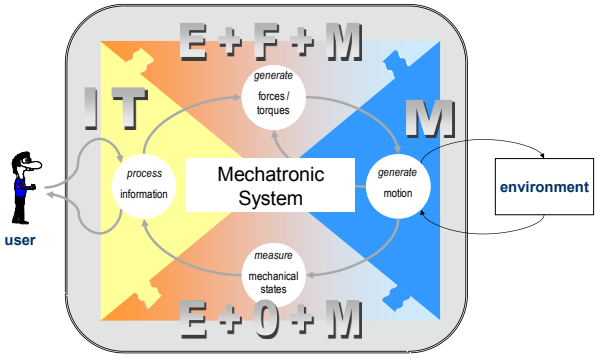

Fig. 1. Mechatronic system components (Janschek and Morozov (2015)).

analysis using an underlying dual-graph error propagation model (DEPM). The DEPM is an abstract mathematical model that captures structural and behavioral system properties, which have the most influence on data error propagation processes, as it is shown in Morozov and Janschek (2014). The DEPM represents a system as a set of executable elements. Two directed graph models are defined on this set: a control flow graph (CFG) and a data flow graph (DFG).

10.1016/j.ifacol.2016.10.643

Faults can be activated in the elements during their execution and result in the occurrence of data errors. Error propagation through the system comprises two aspects: error propagation between the elements and error propagation through the elements. Fault activation and error propagation probabilities are defined for each element in form of probabilistic conditions. Concurrent Markovian analysis of the CFG, the DFG, and these conditions forms a backbone of DEPM application, e.g. it allows to compute mean number of erroneous values in critical system outputs during a given execution time. An example of the DEPM is shown in Fig. 8 and the results of the analysis are described in Section 4.

ErrorPro is a software tool for stochastic data error propagation analysis, developed in our Lab, see Morozov et al. (2015). It is based on the DEPM concept and allows a user to build and compute a DEPM. However, manual creation of DEPMs requires a lot of time and is not acceptable for complex systems. We are using different types of baseline models that can be fully or partially transformed to a DEPM. Currently we are able to generate a DEPM from MATLAB Simulink® and StateFlow®, we implemented a prototype of the transformation from AltaRica (Batteux et al. (2013)) and work on generation of DEPMs from C/C++ code using LLVM technologies.

In recent case-studies (Morozov and Janschek (2011), Morozov and Janschek (2013), Morozov and Janschek (2014))) we widely used ADs as baseline models, because they explicitly depict information about control and data flows. However, the transformations were not fully automatic. Despite the obvious similarities of ADs and DEPMs the transformation is not so straightforward. In this article we present a complete algorithm for automatic transformation of annotated ADs to DEPMs. The transformation will be demonstrated as part of a complete analytical workflow shown in Fig. 2.

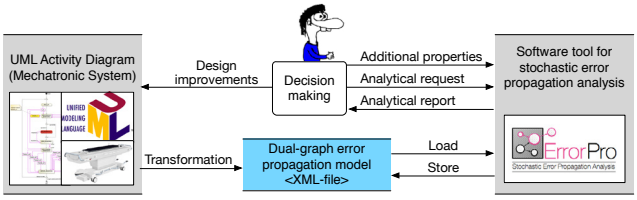

Fig. 2. The transformation algorithm, introduced in this article, as a part of an analytical, model-based workflow.

This article is organized as follows: Section 2 describes relevant methods and tools, related to the analysis of AD models. Section 3 introduces the transformation method itself. Concluding, Section 4 demonstrates how the algorithm can be used as a part of a complete analytical workflow of a mechatronic system.

2. STATE OF THE ART

UML (OMG (2015b)) and SysML (OMG (2015a)), which was introduced in 2006 as an extension of UML, are widely used for mechatronic systems design (see e. g.Mrozek (2002, 2003); Mhenni et al. (2012); Follmer et al. (2010)). The ADs are behavioral system models. They are based on directed graphs similar to Petri nets (see OMG (2015b)). However, ADs have only semi-formal semantics, which is not sufficient for analysis and verification. This is a standard motivation to transform ADs into formal models like Petri nets. Various transformation methods are presented in Störrle and Hausmann (2004); Störrle (2005); Staines (2008); Agarwal (2012); Andrade et al. (2009). Increasing complexity of the resulting Petri nets is a key drawback of these methods.

Another approach targeting formal analysis and verification of ADs is transforming ADs to NuSMV input language for analysis with NuSMV model checker (see Lam (2007)). NuSMV is a reimplementation and extension of SMV (symbolic model checker). NuSMV is a classic model checker that verifies finite state systems, which can be reliably used for the verification of industrial designs.

An approach which is targeting performance analysis by transformation of the AD into a discrete time Markov chain (DTMC) and analysis with the PRISM model checker (see Kwiatkowska et al. (2011)) is proposed in Jarraya et al. (2007). This method looks rather similar to our approach to error propagation analysis from the first view. We also use DTMC models and the PRISM model checker for computation of the DTMCs, but the DTMC models and an interface with the PRISM software are completely different. Data flow is not considered as well.

All the discussed transformation methods were developed in order to serve their particular goals. These methods either completely ignore the information, which is important for us, such as data flow or significantly limit design capabilities, for example restricting not to use *Pins*. *Pin* is an important AD node, which appears commonly in pairs and is used very frequently by system designers. No existing method for the transformation from ADs complying with our requirements, which could be even partially reused, has been found in literature.

3. TRANSFORMATION METHOD

3.1 Meta-model of UML ADs

A UML class diagram, shown in Fig. 3, describes the reduced meta-model of UML ADs. The diagram is based on several class diagrams from OMG (2015b). However, only components required for the transformation to a DEPM model are shown.

Formally, an activity diagram, is a directed graph that is composed of several types of nodes (*ActivityNode*) and directed edges (*ActivityEdge*). There are three types of *ActivityNodes*: *ExecutableNode*, *ObjectNode*, and *ControlNode*.

An *Action* is a fundamental *ExecutableNode* of an AD. Execution of an action represents a basic process or transformation that occurs within a modeled system. An *Action* may receive data, process it, and provide to other *Actions*.