Advanced Engineering Informatics 28 (2014) 232–240

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

A co-modelling method for solving incompatibilities during co-design of mechatronic devices

Yunyun Ni *, Jan F. Broenink

Robotics and Mechatronics Group, CTIT Institute, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 30 September 2013
Received in revised form 7 May 2014
Accepted 31 May 2014
Available online 11 July 2014

Keywords:
Cyber-physical systems
Co-design
Co-modelling
Event detection
Controller layered structure
Fault modelling

ABSTRACT

The design process of mechatronic devices, which involves experts from different disciplines working together, has limited time and resource constraints. These experts normally have their own domain-specific designing methods and tools, which can lead to incompatibilities when one needs to work together using these those methods and tools. Having a proper framework which integrates different design tools is of interest, as such a framework can prevent incompatibilities between parts during the design process. In this paper, we propose our co-modelling methodology and co-simulation tools integration framework, which helps to maintain the domain specific properties of the model components during the co-design process of various mechatronic devices. To avoid expensive rework later in the design phase and even possible system failure, fault modelling and a layered structure with fault-tolerance mechanisms for the controller software are introduced. In the end, a practical mechatronic device is discussed to illustrate the methods and tools which are presented in this paper in details.

1. Introduction

Designing mechatronic devices is a challenging task for various reasons: (1) There are limited time and resources (money, human) for new innovation and assessment of mechatronic devices in current commercial markets; (2) New devices with novel functionality and an acceptable reliability need to reach the market before other competing products; (3) Mechatronic device development is a multi-disciplinary process, involving those who have domain-specific knowledge in their own field, such as electrical engineers, software engineers and mechanical engineers. In order to accomplish a design process in shorter design cycles, lower cost and better quality, engineers often use their own domain-specific design tools to perform simulations to avoid expensive physical prototypes in early design stages. Mechatronic devices are currently also known as *Cyber-Physical Systems* (CPS), as they consist of digital devices (computer, micro-controller, etc.) interacting with analogue (continuous-time) machines. Intrinsically, CPS have their incompatibilities, as the arithmetically and logically (binary) computed controller software is executed in discrete time while the dynamic plant is modelled in continuous time (i.e.: differential equations). Experts from different domains have different terms for the same concept or have the same term for different concepts. Both these cases are problematic, which can lead to serious problems later [1].

To handle possible fatal design flaws of mechatronic devices, modelling possible faulty behaviour of these devices and designing software that deals with this faulty behaviour at an early design stage is helpful to construct the actual devices "First-Time-Right".

Methodologically, there are two major directions to perform modelling and simulation for CPS [2]: (1) use a *homogeneous* system model, i.e. using a single modelling language to express the whole CPS, and consequently use a single simulator and (2) use a *heterogeneous* system model, i.e. using different domain-specific modelling languages to model components from different domains, each simulated with their own simulator, and thus need a means to couple the involved simulators.

Using the *homogeneous* modelling approach, a model transformation from one domain to another is needed in order to model CPS in one single language. This regularly involves more abstractions and simplifications than originally planned,which in general compromises model fidelity. Furthermore, engineers from different specific domains often have the conceptual incompatibilities as mentioned above, causing misunderstandings and abstracting away relevant aspects, resulting in incompetent model parts. However, the single modelling formalism approach does work in case one of the domains is most relevant for the design: When one domain behaviour of the system is dominant, a system model

* Corresponding author. Tel.: +31 53 489 2626; fax: +31 53 4892223.
E-mail addresses: y.ni@utwente.nl (Y. Ni), j.f.broenink@utwente.nl (J.F. Broenink).

http://dx.doi.org/10.1016/j.aei.2014.05.004
1474-0346/

can be made in which the other domains are either ignored or simplified and formulated in the formalism of the dominant domain. For example, when the continuous-time (CT) part behaviour of the system is dominant, a purely CT representation can be made, in which the discrete-event (DE) part is abstracted away, or modelled very concise.

The *heterogeneous* system modelling approach preserves the hybrid properties of the systems by modelling the components in their own most suitable formalism. In this way, the CT components of CPS are modelled in one language which is best suitable for physical-systems dynamics modelling, while the DE components are modelled in an other appropriate modelling language. In this case, no sacrifice in any modelling domain needs to be made. This approach, however, has the risk that since each of the modelling formalisms and thus simulators has its own notion of time, they simply do not work together naturally. A proper synchronisation scheme to couple these different simulators is therefore needed. Simulation of such a combination takes in general more simulation time than when a homogeneous approach was used.

The proposed approach in this paper is to perform a co-modelling methodology (heterogeneous system modelling methodology) supported by a co-simulation tool framework which can address the incompatibilities described in the previous paragraphs.

Other research work has been done related to co-modelling methods and implementing the methods using tool frameworks: *Modelica* [3] is an object-oriented, equation based multi-domain language for simulating controlled physical systems, and provides a number of open and closed source libraries of physical components. However, in general, Modelica simulators cannot perform co-simulations that combine DE and CT computation domains together. The Functional Mockup Interface (FMI) [4] is a tool-independent standard for exchanging data between dynamic models, which is executed by implementing Functional Mock-up Units (FMU) that contain concrete mathematical models describing possible events in the related models. However, as it is indicated in [5], due to the fact that FMU is at a lower abstraction level comparing to Modelica and more target-oriented, it is less flexible. *Ptolemy* II [6] supports heterogeneous simulation from a methodology point of view, where per diagram a Model of Computation must be specified. It is implemented as a single tool. However, in [7,8], it was shown that dynamic plant modelling in Ptolemy II is less intuitive than 20-sim[1] (details about this tool will be mentioned later in Section 2.2.3).gCSP [9] is to graphically model concurrent process-oriented software based on the CSP formalism [10]. Co-simulation of networked control systems has been tried out [11], but the tool never reached maturity. *Cosimate*[2] is a backplane co-simulation tool offering interfaces to tools like Simulink, Modelsim, Modelica. Only time synchronisation is supported as exchanging data between simulators every time step. *Cosimate* has been tried out on the control of a mechatronic test set up [12]. The two discipline-specific models involved have to be connected in a rather cumbersome way.

In this paper, we discuss how our co-modelling method can help to solve the incompatibilities coming along with the designing process of mechatronic devices. In Section 2, we present some essential modelling and simulation concepts, explaining our modelling top-level structure, the details about the co-simulation tool integration framework. In order to make the design process of mechatronic devices "First-Time-Right", details about fault modelling and its corresponding controller software fault handling will also be introduced. Section 3, the introduced methods and the tools are demonstrated by using an existing mechatronic device as an example. Finally, conclusions are drawn and directions for future work are indicated in Section 4.

2. Approach

In Section 2.1, the proposed co-modelling concepts and methods are introduced, following by details about the co-simulation tool integration framework in Section 2.2. Fault modelling and fault handling in controller software are presented in Section 2.3.

2.1. Modelling methodology

To avoid unnecessary misunderstanding about commonly used terms in this paper for readers from different background (domains), a short explanation about concepts related with co-modelling is included.

Co-modelling is a heterogeneous modelling approach in which different, domain-specific modelling methodologies are used. It is supported by a co-simulation tool framework which integrates different domain-specific tools. The simulators under the co-simulation framework are connected through a co-simulation engine. The details about the synchronisation schemes among different tools are explained in Section 2.2.

Collaborative modelling is one step of the whole *co-design* process, which means more than one person is working together. Engineers from different domains can perform collaborative modelling, but this process does not necessarily need to be a co-modelling process unless the tools can synchronise with each other. Details about collaborative modelling on a pilot study can be found in [1].

Our proposed *co-modelling* approach is one of the options to perform collaborative modelling. It is considered as less error-prone than those depending purely on human communication, since there, the human factors involved easily introduce unnecessary faults.

2.1.1. System top-level model

In our methodology, a mechatronic device is divided into several top-level components as shown in Fig. 1: *Controller*, *IO* and *Plant*. The `Controller` (DE domain) block represents the control algorithm and/or logic, which ultimately get implemented in a control computer. The `Plant` (CT domain) block models plant dynamics, which involve the relevant physics domains, like electrical, mechanic, pneumatic, hydraulic, and thermal. The data conversion between the controller and the plant (as is needed to connect the two different modelling domains) is modelled in the `IO` block, as it is also there in the real system. I/O components such as A/D, D/A converters, Samplers, Zero-Order Holds are modelled inside this `IO` block as well. The discrete-event parts of the `IO` block (i.e. which will eventually be implemented in the control computer) are modelled in DE domain, while its continuous-time parts are modelled in CT domain. This is indicated in Fig. 1 by the `IO` block having two different background shades.

2.2. Tool integration framework

2.2.1. General concepts

In this paper, we use the concept of co-model and co-simulation to express and execute CPS models [13]. A co-model is a model

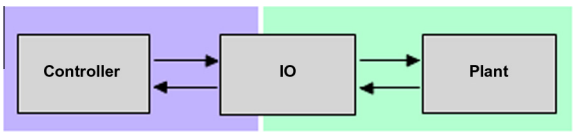

Fig. 1. Top-level structure of the system model.

[1] www.20sim.com.
[2] www.chiastek.com/products/cosimate.html.