2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Toward a flexible control design framework to automatically generate control code for mechatronic systems

Ö. Aydın Tekin, Robert Babuška, Tetsuo Tomiyama, and Bart De Schutter

Abstract— **Software development for the control of mechatronic systems in industry still suffers from the lack of integration of the domains involved, the lack of verification tools, and the fact that the physical structure of mechatronic systems, irregular situations, and service functions are often insufficiently taken into account. To address these problems, we present a framework and a set of tools with which a multidisciplinary product development team can (almost) automatically generate control software for mechatronic systems.**

I. INTRODUCTION

The design of mechatronic systems comprises the following domains: mechanics, electronics, (embedded) systems software, and control. Design approaches for mechatronic systems have traditionally been *sequential* (see Fig. 1). The main drawback of this approach is that possible deficiencies in the mechanical design can only be detected after physical prototyping and that it may be too expensive to restart the design from the beginning.

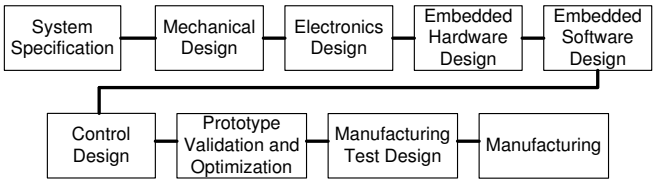

Fig. 1. Traditional sequential design approach for mechatronic systems (adopted from: National Instruments, http://zone.ni.com).

Besides the high costs, the traditional sequential design approach usually does not lead to optimal overall behavior, since the mechatronic system is characterized by an interaction of mechanical, electronics, and control behaviors. In many applications, the role of the controller is considered as a compensator for the imperfections of the mechanical construction in the last stage of the design [1]. If the mechanical structure is chosen well then a good performance can be achieved with a well designed controller after the mechanical design is completed. However, often a cheaper mechanical construction could be built with the same performance if the control design aspects would have been taken into account from the beginning [1]. Clearly there is a need for a concurrent engineering approach, where mechanical and control design parameters are simultaneously optimized [2], [3]. This can only be achieved by considering the design aspects of these domains in a concurrent manner.

The methodology we propose in this paper adopts a concurrent design approach where the mechanical, electronics, and control behaviors are simultaneously optimized, taking the embedded real-time system design concepts into account (see Fig. 2). This saves time and money, decreases the risk of errors in the design phase, and enables the use of new prototyping techniques. Concurrent design methods require a high-level framework to integrate all the separate design domains that have their own design tools and designers who are usually unaware of the tools and methods of the other domains. Developing such a high-level integration framework and integration of each design phase within this framework are the main challenges of this methodology.

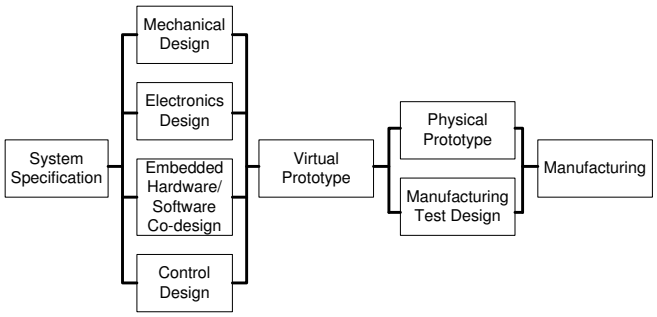

Fig. 2. Concurrent design approach (adopted from: National Instruments, http://zone.ni.com).

In the literature, an idea that is similar to our approach has been proposed in [1]. There, an early conceptual design phase is introduced, in which the requirements for the overall mechatronic system are established and clustered by the system architects for the different design domains. After this conceptual design phase, the designers try to maximize the performance of their own domains. However, even if the requirements for each domain hold perfectly, the requirements for the overall system may not hold globally due to the trade-offs between the design domains. This means that improving the performance in one design domain may cause the performance of another design domain to deteriorate.

In our concurrent design framework, we introduce a multiple-model management system to solve this problem.

All the authors are with the Faculty of Mechanical, Maritime, and Materials Engineering, Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands.

Ö. Aydın Tekin, Robert Babuška, and Bart De Schutter are with the Delft Center for Systems and Control.

Bart De Schutter is also with the Marine & Transport Technology department.

Tetsuo Tomiyama is with the Department of BioMechanical Engineering.

Ö. Aydın Tekin o.a.tekin@tudelft.nl
Robert Babuška r.babuska@tudelft.nl
Tetsuo Tomiyama t.tomiyama@tudelft.nl
Bart De Schutter b@deschutter.info

The design starts with functional description of the overall mechatronic system and multiple design models and the requirements for each model are generated automatically. Then the design for each domain goes on in parallel and the interactions between the design models are supported. While currently available tools can be used within our framework, we expect that new prototyping tools will be developed as well.

This paper is organized as follows. Section II reviews high-level approaches and tools for a concurrent mechatronic system design. Section III discusses the available methods for the integration of the design domains using the bond graphs and UML (Unified Modeling Language). Section IV presents our concurrent design framework the *multiple-model management system*. Section V reviews the tools and methods for automatic controller design and code generation. Section VI concludes the paper with an outline of future steps that should be taken to obtain an optimal mechatronic system design method.

II. HIGH-LEVEL APPROACHES: STATE OF THE ART

The need for high-level approaches for concurrent mechatronic system simulation and design has already been recognized [2], [4]. Although no integrated tools are available to support the interaction between the design domains, separate tools have been developed for pairs of domains. Examples are the integration of mechanical design and control design, or embedded software design and control design, as detailed below.

A. Toward a mechatronic compiler

Building a "mechatronic compiler" has been proposed in [2] to translate high-level specifications automatically into a mechatronic system where mechanical design and control design are optimized together. To illustrate the idea, design of a controller for a 3-axis machine configuration has been considered. The configurations are stored as classes in a library and controllers for this class of systems can be designed simultaneously in the same environment. A model of the machine is constructed by choosing the best mechanical configuration by means of a genetic algorithm, given high-level specifications like maximal static stiffness, high dynamic stiffness, maximal working space, etc. A robust H_∞ controller is applied in order to handle uncertainties arising among others from the fact that a family of machine configurations is considered. In this way, the mechanical design and the controller design are optimized simultaneously.

B. Embedded control systems

Relatively more effort has been put into the integration of embedded systems design and control design. It is well known that the performance of the control algorithm vitally depends on the respective real-time implementation and the computer hardware platform. The interplay between control design and embedded real-time system design has been addressed, e.g., in [5] and [6]. Tools that help take the timing effects (i.e. delay and jitter) into account in control design for the concurrent embedded control systems design have been reviewed and compared in [6]. They include mainly AIDA and XILO [7] of the Royal Institute of Technology, Sweden; Jitterbug and TrueTime of Lund University, Sweden; ORCCAD of INRIA, France; Ptolemy II of the University of Berkeley, California; TargetLink [8] by dSPACE, and Torsche [9] of the Czech Technical University, Prague.

However, as pointed out in [10], these tools are specialized on a single domain instead of supporting co-design to generate an embedded control algorithm taking into account the aspects of control design together with communication aspects. In addition, the theory and methods focus mainly on analysis rather than on design and synthesis.

In the following section we discuss the use of bond-graph modeling and UML to integrate the design domains of mechatronics. Afterwards, in Section IV our concurrent design framework called the multiple-model management system is presented.

III. INTEGRATION OF THE DESIGN DOMAINS OF MECHATRONICS: STATE OF THE ART

Optimizing the mechatronic system design requires an early stage design phase. A sufficiently sophisticated controller should work well for the initial conceptual design of a mechatronic system. Similarly, a reduced-order model of the physical structure can also be used initially to tune the controller. However, using a simplified model may cause the transfer functions or the state-space descriptions of the mechanical system to lose the relation with the physical parameters [1]. Therefore a new framework and/or new tools are required to keep the links to the dominant physical parameters (i.e. mass and stiffness) in the model and let the other design domains' tools access these parameters through these links, i.e., the control design parameters can be tuned as well as the mechanical system parameters in the same environment.

The literature indicates the importance of the bond-graph modeling for concurrent multidisciplinary systems design. Bond graphs are exploited to form a common design environment using the components of both mechanical and electronic design domains for physical modeling of the mechatronic systems [1], [11]–[13]. At the same time, UML, the object-oriented modeling language has attracted considerable attention for the integration in the software domain, utilizing the code reusability and modularity [4], [13]–[16].

The use of bond graphs and UML for an integrated mechatronic system design are discussed in the following sections.

A. Bond graphs

Bond graphs provide a way to represent typical components of mechatronics from electromechanical, electro-hydraulic, thermo-fluid, and electronic control system domains in a unified modeling environment [11], [17]. Energy is the common ground that connects the physical laws for different domains. Bond graphs are used to analyze the

flow of energy through the interconnections (ports) of the components from different domains as the product of a flow variable (i.e. velocity or current) and an effort variable (i.e. force or voltage).

While the flow of energy is useful for combining the components of mechanical and electronic design domains, one may argue that a general control system is more conveniently designed using the flow of information. To bridge the bond-graph models of electromechanical systems with the control domain, a tool called CAMP-G [11], 20-sim [1], and Dymola with Modelica language [18] can be used. These tools can generate state-space models corresponding to the physical (bond-graph) model of the plant and then export them into Matlab/Simulink or Scilab/Scicos [19]. Even the non-linear Modelica models can be used directly in Simulink. In addition, one does not necessarily need to know bond-graph modeling to benefit its advantages by using the iconic diagrams of 20-sim[1] [1].

The next section explores the use of UML to integrate the multiple views of mechatronics at the highest meta-modeling level.

B. The use of UML for mechatronic systems

UML is the most widely used object-oriented visual modeling language for software specification, analysis, and design [20]. Despite the debates on the use of UML for real-time software design [21], there is a clear trend to use UML for creating abstract meta-models that are platform-independent and that, later on, can be mapped into any platform-specific application by a model compiler [13]. The aim of all the efforts in this area is to build a common description language so that the designers from different disciplines can understand the global behavior of the whole system. Moreover, UML is an object-oriented modeling language that keeps the reusability and the modularity at the top level.

UML allows the model-specific design by means of stereotypes derived from the basic UML meta-model elements. A consistent set of stereotypes is called a profile. UML-RT (here RT stands for Real-Time) profile adds five new stereotypes to UML: capsules, ports, connectors, protocols, and protocol roles. These additional stereotypes help to make UML suitable for modeling complex real-time systems [14], [22].

A unified language to obtain a complete description of an automated system, suitable for its simulation, documentation, and validation has been presented in [13]. It integrates the specifications for the design of the automated system, including the dynamics of the physical plant, the discrete-event behavior of distributed control software, and the specification of interface ports between the plant and the controller. Exploiting bond graphs and their mathematical formalization as port-Hamiltonian systems, it has been shown that any physical system can be described within the UML-RT profile.

[1]http://www.20sim.com/

To illustrate the potential benefits of using UML for the integrated design of control systems further, two related projects are highlighted below.

C. Projects that use UML for the integration and control code generation

There are two projects related to the research on the development of the integrated design notation (IDN) which is proposed to integrate process engineering, software engineering, and control engineering: (1) the PiCSI (Process Control Systems Integration) Project [23], [24] and (2) the FLEXICON (Flexible Control Systems Development and Integration Environment for Distributed Control) Project [25].

Both projects aim to achieve the integration and co-simulation of continuous control and sequential control. Continuous control and sequential control are applied to control the continuous dynamics and the discrete dynamics of the hybrid system respectively. For the continuous control they use Simulink, and for the sequential control they use SFC (Sequential Function Chart) which is a graphical language for PLCs covered in the IEC 61131-3 standard by the International Electrotechnical Commission. UML is chosen as the core of the integrating development environment. The models designed by using different design tools like Simulink, Stateflow, and SFC are translated into UML diagrams for the integration. Translating UML diagrams into hardware independent Java code concludes the integration and code generation approach.

IV. PROPOSED FRAMEWORK: MULTIPLE-MODEL MANAGEMENT SYSTEM

The framework we propose will be described now in more detail, which has formerly been introduced in [26] and [27]. In our framework, the design starts with a functional and behavioral description of the plant and the requirements for the overall performance of the mechatronic system. This description is in a language that is close to human perception and serves as the input to the integrated design framework (see Fig. 3). The functional description is translated into qualitative behaviors for the mechanical, electronic, and control design domains. As control design relies on quantitative models and specifications, the qualitative models and the requirements, obtained from the functional description, must be converted into an appropriate form, such as a Simulink block diagram and specifications in terms of, e.g., overshoot and rise time. The next step is to decide which control method to use to satisfy these design requirements.

Note that our aim is not to develop a mechatronic system design tool that enables the design of the whole mechatronic system with all the components from multiple domains, but to design the controller automatically based on multiple models in a unified framework that integrates a set of tools from multiple domains. The former aim seems an unrealistic approach for two reasons: (1) the design of the whole system is often too big and complex to fit into one single environment, (2) the tools are usually designed for specific

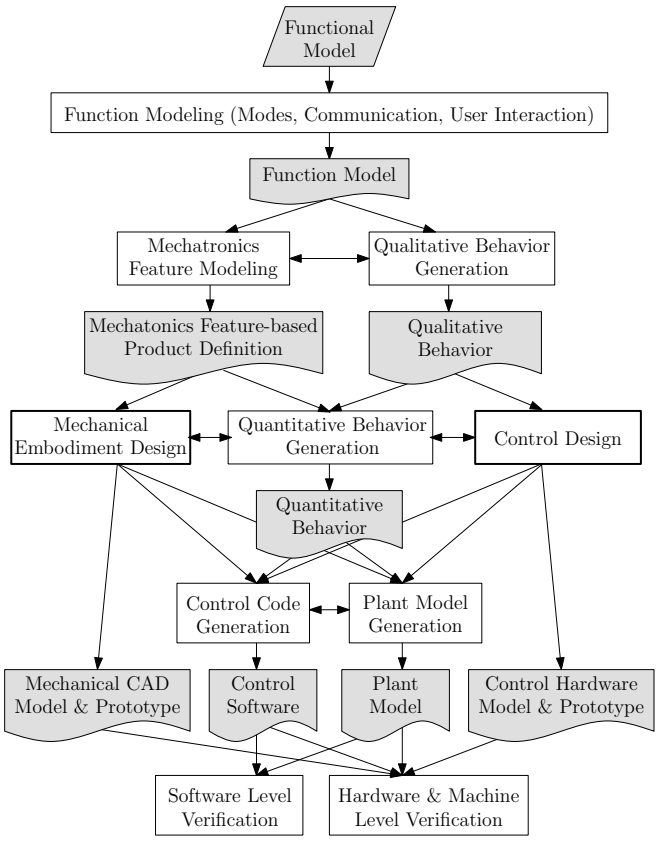

Fig. 3. Multiple-model management system.

domains and the designers have been educated to work with these tools.

Matlab serves a good basis for the design of controllers integrating many toolboxes and Simulink as a user friendly block-oriented design and simulation environment. Matlab also provides other tools viz. SimMechanics and SimElectronics to model the mechanical and electronic parts of the system; however, simulating the controller together with the mechanical and electronic models in closed loop is not necessarily convenient. Although Matlab/Simulink fits best for the control design, Dymola with the open source object-oriented language Modelica that can also use the bond graphs modeling framework may be an alternative choice for the design, modeling, and simulation of electromechanical systems [18].

Therefore, our approach keeps all the different design models as they are and enables the designers of different models to oversee the whole design such that the designers have more access to other domain models while working on their own specific design with the help of the qualitative behaviors generated within the framework. The controller is designed having all the qualitative behavior information of the whole system and the most up-to-date quantitative model of the (electro)mechanical plant. In parallel to the controller design (control code generation), the plant model is generated from the (electro)mechanical embodiment design domain which uses the recent quantitative model of the controller for the electromechanical system design.

The designed controller is verified using the dynamics of the latest physical model in the closed loop. Simultaneously, the physical model is verified using the dynamics of the controller in the closed loop. [27] can be referred to for further details about the plant model generation and the verification parts of the proposed approach. After the verification is accomplished by means of iterative simulations, the control software is generated by the available code generation mechanisms like the Matlab/Simulink/Real-Time Workshop. Finally the generated control code is validated on the real system (or its prototype). In summary, our framework enables the automatic generation of control code from functional descriptions of a mechatronic system supporting the co-design of control, mechanical, and electronic design domains [26], [27].

The next section elaborates further the control design and the generation of the control code automatically from the qualitative behaviors and the requirements obtained from the functional descriptions of the whole mechatronic system.

V. AUTOMATIC CONTROLLER DESIGN FROM REQUIREMENTS

This section starts with brief exposition about the computer aided control system design (CACSD) tools with code generation capabilities for real-time implementation and goes on with a survey of the attempts taken to develop a standard methodology for control system design from the requirements in a top down approach.

A. CACSD software tools for control code generation

Although there have been many CACSD tools developed for the design and simulation of some specific areas of control systems, only a few CACSD software packages like FBCad [28], 20-sim (by Technical University of Twente), Mathematica, Matlab, and Scilab are commonly used. Some of these tools are reviewed below starting with Matlab/Simulink which is the dominating CACSD software in the control engineering area [23], [29], [30].

1) Matlab/Simulink: The studies on Rapid Controller Prototyping (RCP) with Matlab/Simulink go back to 90s in the effort to fill the gap between the two essential phases of control engineering: (1) control systems analysis and simulation, (2) real-time implementation of control algorithms. Real-time extensions of MS-DOS have been considered to provide low-cost solutions using PCs with open-architecture standard hardware followed by a rapid controller prototyping toolbox for Windows 95/98/NT, which automatically generates real-time code from a Simulink model using Real Time Workshop (RTW) [29]. By the development of code generation mechanisms and Simulink modules to embed Matlab/Simulink models into microcontrollers [30], Matlab has become the dominating tool for the control system design and implementation.

2) RTAI – Real-Time Application Interface: Matlab/Simulink/RTW is integrated with the Linux Real-Time Application Interface (RTAI) by a tool called RTAI-Lab

which also supports the open source CACSD Scilab/Scicos[2]. RTAI-Lab is an alternative to the Simulink external mode, but with reliable hard real-time support on the Linux RTAI distribution. It is also possible to use distributed control with the Linux RTAI-Lab GUI, which facilitates a hard real-time network communication [31].

3) FBCad – Functional Block Computer-aided design: FBCad is a prototype CACSD tool for designing and analyzing reusable high-level control software components and for generating run-time code for distributed control systems [28]. FBCad can generate the C++ source code automatically for the simulation of robot control systems with 3-D graphical animations. FBCad uses function blocks (FBs) that are already defined in the IEC 61499 standard. FBs are discussed further in Section V-B.

4) R2D2C – Requirements 2 Design 2 Code: Initial steps have been taken for designing a system from requirements which is called "Requirements-Based Programming (RBP)" for software development [32]. There is still a large gap in going from requirements to design, but "RBP seeks to eliminate this gap by ensuring that the ultimate application can be fully traced back to the actual requirements of the system" [33].

After this brief review on the tools that can generate the control code from the control system models and on the first steps toward fully functional requirements based programming (RBP), in the following section, the idea of designing the controller systematically or automatically from the system requirements is discussed.

B. CDM – Control Development Methodology

The first question that one might ask when developing a control code generation software is: Is there a structured way of designing controllers?

The European Space Agency has already defined a basic control development methodology (CDM) for space applications. Additional features to the CDM have been investigated [34]. Moreover, some principles leading to structured and formal design methods for the development of distributed control systems for robotic applications have been addressed [28], [34].

The main purpose of such a structured control design methodology for robotic applications is to enhance the *reusability* of the previous projects by dividing the whole design problem, in a structured, hierarchical way, into smaller tasks and functions that are easily understood and that can be reused. So, the gap between the functional design and the final implementation is filled in five steps by the CDM [28].

The first step of the CDM defines the design problem clearly. The second step divides the design problem into simpler control tasks (and further subtasks) such that each task can be realized by a control algorithm. Formal representation of these control tasks have been defined by the industrial standard IEC-61499 as the fundamental functional block (FB). An FB represents a functional unit of the software associated to a control task. Connecting instances of FBs in a proper way, it is possible to create a complex control application. The execution control chart (ECC) of the FB model determines the execution order of the control algorithms as a supervisory control system. This allows the design of hybrid systems where continuous dynamics are combined with discrete event systems. The last three steps define the implementation, integration, and verification of the control software and hardware for the whole mechatronic system [28].

[2]www.scilab.org

VI. CONCLUSIONS, OPEN ISSUES AND CHALLENGES

A. Conclusions

Optimizing the mechatronic system design requires the concurrency of the constitutive design domains: mechanical design, electronic design, control design, and embedded systems design. Currently, each of these domains uses different tools and the designers of each domain are usually not familiar with the tools and methods used in the other domains. Easier interactions between the domains will thus lead to a more efficient and higher quality mechatronic system design.

We have discussed high-level approaches like bond graphs and UML for the integration of the design domains. Based on the flow of energy, bond-graph modeling integrates the mechanical and electronic design domains. Being an extendable object-oriented meta-modeling language, UML offers more flexibility for the integration of the software design domain in addition to its potential for integrating the bond-graph models.

A new integration framework called the multiple-model management system has been proposed such that multiple models for the domains of mechatronics are automatically generated from the functional description of the system. Developing such a high-level integration framework and integrating the design domains within this framework are the main challenges of this methodology. Finally, several computer aided control system design tools and the first attempts for a structured control system design methodology have been reviewed. A high-level design example that illustrates our integration framework will be presented in our future work.

B. Open issues and challenges

Toward the development of a framework for an optimal integrated mechatronic system design, there are still open issues and challenges the most important of which will now be outlined briefly.

The controller of the mechatronic system must handle the sequential control represented by discrete event systems as well as the control of continuous systems. To this end, modeling and control of hybrid systems and the respective real-time implementation must be investigated further to come up with a structured method for the design of mechatronic control systems.

New methodology and tools must be developed to generate qualitative behaviors and models for each design domain of mechatronics from the functional model. The requirements for designing the controller need to be generated from the functional model in a similar way. New tools can be developed or the current tools can be investigated to exchange the dominant parameters of the models which may influence the integrated design.

Finally, a "control compiler" should be designed such that controller is constructed automatically from several library components according to the plant model and the requirements. This will lead to the design of a common environment for the integration of the design domains of mechatronics and for the automatic generation of control code.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support of the Dutch Innovation Oriented Research Program 'Integrated Product Creation and Realization (IOP-IPCR-0602)' of the Dutch Ministry of Economic Affairs.

REFERENCES

[1] J. van Amerongen, "Mechatronic design," *Journal of Mechatronics, Elsevier*, vol. 13, no. 10, pp. 1045–1066, December 2003.

[2] H. Van Brussel, P. Sas, I. Nemeth, P. De Fonseca, and P. den Braembussche, "Towards a mechatronic compiler," *IEEE Journal of Mechatronics*, vol. 6, no. 1, pp. 90–105, 2001.

[3] J. M. Rieber and D. G. Taylor, "Integrated control system and mechanical design of a compliant two-axes mechanism," *Mechatronics*, vol. 14, no. 9, pp. 1069–1087, November 2004.

[4] K. Thramboulidis, "Model-integrated mechatronics - toward a new paradigm in the development of manufacturing systems," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 54–61, 2005.

[5] J. El-Khoury, "A model management and integration platform for mechatronics product development," Ph.D. dissertation, Department of Machine Design. Stockholm, Sweden, KTH. Trita-MMK, ISSN 1400-1179, 2006.

[6] M. Törngren, D. Henriksson, K.-E. Arzen, A. Cervin, and Z. Hanzalek, "Tool supporting the co-design of control systems and their real-time implementation: Current status and future directions," in *Proc. IEEE International Symposium on Computer-Aided Control Systems Design*, Munich, 2006, pp. 1173–1180.

[7] O. Redell, J. El-Khoury, and T. M., "The AIDA toolset for design and implementation analysis of distributed real-time control systems," *Microprocessors and Microsystems*, vol. 28, no. 4, pp. 163–182, 2004.

[8] I. Stuermer, M. Conrad, H. Doerr, and P. Pepper, "Systematic testing of model-based code generators," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 622–634, 2007.

[9] P. Šůcha, M. Kutil, M. Sojka, and Z. Hanzálek, "TORSCHE scheduling toolbox for matlab," in *IEEE Symposium on Computer-Aided Control System Design*, Munich, 2006, pp. 50–52.

[10] M. Törngren, D. Henriksson, O. Redell, C. Kirsch, J. El-Khoury, D. Simon, Y. Sorel, H. Zdenek, and K.-E. Arzen, "Co-design of control systems and their real-time implementation - a tool survey," Mechatronics Lab, Department of Machine Design, Royal Institute of Technology, KTH, 100 44 Stockholm, Sweden, Tech. Rep., 2006.

[11] J. J. Granda, "The role of bond graph modeling and simulation in mechatronics systems an integrated software tool: CAMP-G, MATLABSIMULINK," *Mechatronics*, vol. 12, no. 9-10, pp. 1271–1295, November-December 2002.

[12] J. van Amerongen and P. Breedveld, "Modeling of physical systems for the design and control of mechatronic systems," *Annual Reviews in Control*, vol. 27, no. 3, pp. 87–117, 2003.

[13] C. Secchi, M. Bonfe, and C. Fantuzzi, "On the use of UML for modeling mechatronic systems," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 105–113, 2007.

[14] B. Selic, "Using UML for modeling complex real-time systems," in *Proc. ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems*, London, UK, 1998, pp. 250–260.

[15] K. C. Thramboulidis, "Using UML in control and automation: a model driven approach," in *Proc. 2nd IEEE International Conference on Industrial Informatics*, Berlin, June 2004, pp. 587–593.

[16] M. Bruccoleri, C. D'Onofrio, U. La Commare, and F. Raimondi, "UML design and AWL programming for reconfigurable control software development of a robotic manipulator," in *Proc. 10th IEEE Conference on Emerging Technologies and Factory Automation*, 2005.

[17] S. Behbahani and C. W. de Silva, *Mechatronic Systems: Devices, Design, Control, Operation and Monitoring*. CRC Press, Inc., 2008, ch. Mechatronic Design and Optimization, pp. 17–1–17–26.

[18] G. Ferretti, G. Magnani, and P. Rocco, "Virtual prototyping of mechatronic systems," *Annual Reviews in Control*, vol. 28, no. 2, pp. 193–206, 2004.

[19] M. Najafi and R. Nikoukhah, "Modeling and simulation of differential equations in Scicos," in *Proc. 5th International Modelica Conference*, Vienna, Austria, September 2006.

[20] A. Watson, "Visual modelling: past, present and future," Online, Object Management Group, http://www.uml.org/, September 2008.

[21] B. Selic, A. Burns, A. Moore, T. Tempelmeier, and F. Terrier, "Heaven or hell? A "real-time" UML?" *Proc. Unified Modeling Language: UML 2000, Lecture notes in computer science, Springer-Verlag*, vol. 1939, pp. 93–100, 2000.

[22] P. Ferreira, A. Sampaio, and A. Mota, "Viewing CSP specifications with UML-RT diagrams," *Electronic Notes in Theoretical Computer Science*, vol. 195, pp. 57–74, 2008.

[23] D. N. Ramos-Hernandez, P. J. Fleming, and J. M. Bass, "A novel object-oriented environment for distributed process control systems," *Control Engineering Practice*, vol. 13, pp. 213–230, 2005.

[24] D. N. Ramos-Hernandez, I. Zubizarreta, P. J. Fleming, S. Bennett, and J. M. Bass, "Towards a control software design environment using a meta-modelling technique," in *IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002, pp. 255–260.

[25] H. A. Thompson, D. N. Ramos-Hernandez, J. Fu, L. Jiang, I. Choi, K. Cartledge, J. Fortune, and A. Brown, "A flexible environment for rapid prototyping and analysis of distributed real-time safety-critical systems," *Control Engineering Practice*, vol. 15, pp. 77–94, 2007.

[26] A. A. Alvarez Cabrera, M. S. Erden, M. J. Foeken, and T. Tomiyama, "High level model integration for design of mechatronic systems," in *IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, Beijing, China, October 2008, pp. 387–392.

[27] M. J. Foeken, M. Voskuijl, A. A. Alvarez Cabrera, and M. J. L. Van Tooren, "Model generation for the verification of automatically generated mechatronic control software," in *IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, Beijing, China, October 2008, pp. 275–280.

[28] L. Ferrarini and E. Carpanzano, "A structured methodology for the design and implementation of control and supervision systems for robotic applications," *IEEE Journal of Control Systems Technology*, vol. 10, no. 2, pp. 272–279, 2002.

[29] W. Grega, K. Kolek, and A. Turnau, "Rapid prototyping environment for real-time control education," in *Proc. Real-Time Systems Education III*, K. Kolek, Ed., 1998, pp. 85–92.

[30] S. Rebeschiess, "MIRCOS – microcontroller-based real-time control system toolbox for use with Matlab/Simulink," in *Proc. IEEE International Symposium on Computer Aided Control System Design*, Kohala Coast, HI, 1999, pp. 267–272.

[31] R. Bucher and S. Balemi, "Rapid controller prototyping with Matlab/Simulink and Linux," *Control Engineering Practice*, vol. 14, pp. 185–192, 2006.

[32] J. L. Rash, M. G. Hinchey, C. A. Rouff, D. Gracanin, and J. Erickson, "A requirements-based programming approach to developing a nasa autonomous ground control system," *Artificial Intelligence Review*, vol. 25, no. 4, pp. 285–297, 2006.

[33] M. G. Hinchey, J. L. Rash, and C. A. Rouff, "A formal approach to requirements-based programming," in *Proc. 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems ECBS '05*, Washington, DC, USA, 2005, pp. 339–345.

[34] G. Ferretti, G. Magnani, P. Putz, and P. Rocco, "The structured design of an industrial robot controller," *Control Engineering Practice*, vol. 4, no. 2, pp. 239–249, 1996.