Advances in Soft Computing 39

Editor-in-Chief: J. Kacprzyk

Advances in Soft Computing

Editor-in-Chief

Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Marek Kurzynski, Edward Puchala, Michal Wozniak, Andrzej Zolnierek (Eds.)
Computer Recognition Systems, 2005
ISBN 978-3-540-25054-8

Abraham Ajith, Yasuhiko Dote, Takeshi Furuhashi, Mario Köppen, Azuma Ohuchi, Yukio Ohsawa (Eds.)
Soft Computing as Transdisciplinary Science and Technology, 2005
ISBN 978-3-540-25055-5

Barbara Dunin-Keplicz, Andrzej Jankowski, Andrzej Skowron, Marcin Szczuka (Eds.)
Monitoring, Security, and Rescue Techniques in Multiagent Systems, 2005
ISBN 978-3-540-23245-2

Frank Hoffmann, Mario Köppen, Frank Klawonn, Rajkumar Roy (Eds.)
Soft Computing Methodologies and Applications, 2005
ISBN 978-3-540-25726-4

Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, Kryzysztof Trojanowski (Eds.)
Intelligent Information Processing and Web Mining, 2005
ISBN 978-3-540-25056-2

Abraham Ajith, Bernard de Bacts, Mario Köppen, Bertram Nickolay (Eds.)
Applied Soft Computing Technologies: The Challenge of Complexity, 2006
ISBN 978-3-540-31649-7

Mieczyslaw A. Klopotek, Slawomir T. Wierzchon, Kryzysztof Trojanowski (Eds.)
Intelligent Information Processing and Web Mining, 2006
ISBN 978-3-540-33520-7

Ashutosh Tiwari, Joshua Knowles, Erel Avineri, Keshav Dahal, Rajkumar Roy (Eds.)
Applications and Soft Computing, 2006
ISBN 978-3-540-29123-7

Bernd Reusch, (Ed.)
Computational Intelligence, Theory and Applications, 2006
ISBN 978-3-540-34780-4

Miguel López-Díaz, María ç. Gil, Przemysław Grzegorzewski, Olgierd Hryniewicz, Jonathan Lawry
Soft Methodology and Random Information Systems, 2006
ISBN 978-3-540-34776-7

Ashraf Saad, Erel Avineri, Keshav Dahal, Muhammad Sarfraz, Rajkumar Roy (Eds.)
Soft Computing in Industrial Applications, 2007
ISBN 978-3-540-70704-2

Ashraf Saad, Erel Avineri, Keshav Dahal,
Muhammad Sarfraz, Rajkumar Roy (Eds.)

Soft Computing in Industrial Applications

Recent and Emerging Methods and Techniques

Springer

Editors

Dr. Ashraf Saad
Department of Computer Science
Armstrong Atlantic State University
11935 Abercorn Street
Savannah, Georgia 31419-1997
USA
E-mail: ashraf@cs.armstrong.edu

Dr. Erel Avineri
Centre for Transport & Society
Faculty of the Built Environment
University of the West of England
Frenchay Campus
Coldharbour Lane
Bristol BS16 1QY
UK
E-mail: Erel.Avineri@uwe.ac.uk

Dr. Keshav Dahal
MOSAIC Research Group
University of Bradford
Department of Computing
Bradford BD7 1DP
UK
E-mail: K.P.Dahal@Bradford.ac.uk

Dr. Muhammad Sarfraz
Information & Computer Science Department
King Fahd University of Petroleum & Minerals
KFUPM #1510
Dhahran 31261
Saudi Arabia
E-mail: sarfraz@ccse.kfupm.edu.sa,
sarfraz@kfupm.edu.sa

Prof. Rajkumar Roy
Decision Engineering Centre
Manufacturing Department
Cranfield University
Bedford MK43 OAL
UK
E-mail: r.roy@cranfield.ac.uk

Library of Congress Control Number: 2007923718

ISSN print edition: 1615-3871
ISSN electronic edition: 1860-0794
ISBN-10 3-540-70704-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-70704-2 Springer Berlin Heidelberg New York

Typesetting: by the authors and SPS using a Springer LaTeX macro package

Printed on acid-free paper SPIN: 11585275 89/SPS 5 4 3 2 1 0

Preface

On behalf of all members of the International Technical Program Committee of the 11th Online World Conference on Soft Computing in Industrial Applications (WSC11), we would like to extend our sincere welcome to you. The conference continues a tradition started over a decade ago by the World Federation of Soft Computing (WFSC) to bring together researchers interested in advancing state of the art in the field. Continuous technological improvements since then continue to make this online forum a viable gathering format for a world class conference.

The program committee received a total of 63 submissions, of which 61 papers qualified for peer review by the International Program Committee. Each paper was then reviewed by at least three referees, culminating in the acceptance of 30 papers for publication. Authors of all accepted papers were then notified to prepare and submit their final manuscripts and conference presentations. This resulted in a total of 28 final submissions by 73 authors that comprise the six sessions of the conference program. Based on the reviewers' reports, the authors provided revised versions of the papers – all of them are featured in this book. Also featured is an invited paper based on a keynote presentation. The authors of several outstanding papers have been invited to submit significantly revised and extended versions of their papers to the Applied Soft Computing Journal.

We extend our sincere thanks to all authors and to all members of the International Program Committee for their clear and unwavering commitment to the success of WSC11. Reflecting the worldwide nature of WSC11, authors, members of the program committee and the conference organizers are from over 20 countries and five continents. We also extend our thanks to our keynote speaker, Dr. Pieter Mosterman of the MathWorks for his contributed talk.

November 29, 2006

Ashraf Saad
General Chair of WSC11
Savannah, Georgia, USA

Erel Avineri
Program Chair of WSC11
Bristol, UK

Message from the WSC11 General Chair and Program Chair

It is our pleasure to officially announce the start of the conference. The official WSC11 web site has been relocated since August to the following URL: http://www.cs.armstrong.edu/wsc11/. Please make the necessary changes to any web pages that you maintain with reference to the conference. That will increase the chances of search engines pointing to the correct WSC11 web site.

An opening note has been posted to the conference web site along with the final pdf version of all accepted papers. With regard to the presentation of papers and the keynote, we will be able to support (for the first time in WSC's history) real-time presentations via audio conferencing. This is made possible through a kind three-week trial offer (for the duration of the conference) of Elluminate (http://www.Elluminate.com), a Java-based (http://java.sun.com/products/javawebstart/) webinar environment. In return, we will provide feedback about the use of this web-based conferencing tool in support of our worldwide conference. In order to get an idea of the use of this tool, please visit the following URL: https://sas.elluminate.com/m.jnlp?sid=1125&password=M.161974A26FAAF95DB6C50F2C6CFF05 where an image version of the opening note is currently posted for testing purposes.

Therefore, we request from each correspondence author to email us back by Friday, September 22, with his/her availability to make a 25-30 minutes presentation during the upcoming two weeks (Sep 25-Oct 6). Please provide us with 2-3 possible times, and indicate your local time zone as it relate to GMT (e.g., EST in the US is GMT-5, while Brazil should be GMT-4). A presenter will need a Java-enabled computer, with a reasonable high quality connection to the Internet, and which is also equipped with a speaker and a microphone (or a headset). We will schedule all presentations and upload into Elluminate the presentation slides that have been submitted in August. A final schedule of presentations will be posted and emailed to all by Monday, September 25. All interested participants will then be able to connect to a presentation at the scheduled time, up to a maximum of 30 seats per session. We will expect session chairs to attend as many of the presentations of their sessions as possible.

It is indeed an exciting development for us to be able to support a synchronous mode of interaction for WSC11 given our global community. We also hope to witness a strong level of participation in the sessions by researchers from all four corners of the globe.

September 18, 2006

Ashraf Saad
General Chair of WSC11
Savannah, Georgia, USA

Erel Avineri
Program Chair of WSC11
Bristol, UK

WSC11 Organization and International Program Committee

General Chair

Ashraf Saad, Armstrong Atlantic State University**, USA
** Formerly with the Georgia Institute of Technology

Program Chair

Erel Avineri, University of the West of England, Bristol, UK

Advisory Board

Hisao Ishibuchi, Osaka Prefecture University, Japan
Rajkumar Roy, Cranfield University, UK
Ajith Abraham, Chung-Ang University, Korea
Mario Köppen, Fraunhofer IPK, Berlin, Germany

International Co-chairs

Lakhmi Jain, University of South Australia, Australia
Serge Popov, Kharkiv University of Radio Electronics, Ukraine
Muhammad Sarfraz, King Fahd University of Petroleum and Minerals, Saudi Arabia
Ashitosh Tiwari, Cranfield University, UK

Publicity Chair

Keshav Dahal, University of Bradford, UK

International Technical Program Committee

Janos Abonyi, University of Veszprem Folyamatmérnöki Tanszék, Hungary
Bart Baesens, Catholic University of Leuven, Belgium
Valeriu Beiu, United Arab Emirates University, UAE
Sugato Bagchi, IBM Research, USA
Soumya Banerjee, BITS Mesra, India
Christian Blum, Universitat Politecnica de Catalunya, Spain
Ulrich Bodenhofer, Software Competence Center, Austria
Andrea Bonarini, Politecnico de Milano, Italy

Oscar Castillo, Instituto Tecnológico de Tijuana, Mexico
Siam Charoenseang, King Mongkut's University of Technology, Thailand
Leandro Coelho, Pontifical Catholic University of Parana, Brazil
Carlos A. Coelho, CINVESTAV, Mexico
Oscar Cordon, University of Granada, Spain
Gaspar Cunha, University of Minho, Potugal
Suash Deb, National Institute of Science & Technology, India
Guy De Tré, Ghent University, Belgium
Mauro Dell'Orco, University of Bari, Italy
Giuseppe Di Fatta, University of Konstanz, Germany
Katrin Franke, Fraunhofer IPK, Germany
Aureli Soria Frisch, Universitat Pompeu Fabra, Spain
Xiao-Zhi Gao, Helsinki University of Technology, Finland
Takeshi Furuhashi, Nagoya University, Japan
Crina Grosan, Babes-Bolyai University, Romania
Roderich Gross, Universite Libre de Bruxelles, Belgium
Hani Hagras, University of Essex, UK
Ioannis Hatzilygeroudis, University of Patras, Greece
Ayanna Howard, Georgia Institute of Technology, USA
Yaochu Jin, Honda Research Institute Europe, Germany
Uri Kartoun, Ben Gurion University of the Negev, Israel
Okyay Kaynak, Bogazici University, Turkey
Frank Klawonn, University of Applied Sciences, Germany
Joshua Knowles, University of Manchester, UK
Andreas König, Technische Universitat Kaiserslautern, Germany
Renato Krohling, University of Dortmund, Germany
Reza Langari, Texas A&M, USA
Luis Magdalena, Universidad Politecnica de Madrid, Spain
Max Manfrin, Universite Libre de Bruxelles, Belgium
Christophe Marsala, Universite P. et M. Currie, France
Patricia Melin, Instituto Tecnológico de Tijuana, Mexico
Sanaz Mostaghim, ETH-Zurich, Switzerland
Mehmet K Muezzinoglu, University of Louisville, USA
Lakshmi Narasimhan, The University of Newcastle, Australia
Detlef D Nauck, British Telecom, UK
Nadia Nedjah, State University of Rio de Janeiro, Brazil
Andreas Nuernberger, Universität Magdeburg, Germany
Jae C. Oh, Syracuse University, USA
Sankar K. Pal, Indian Statistical Institute, India
Vasile Palade, Oxford University, UK
Gerardo Rossel, Universidad Abierta Interamericana, Argentina
Yos Sunitiyoso, University of the West of England, Bristol, UK
Vicenc Torra, AI Research Institute, CSIC, Spain
Edward Tunstel, Jet Propulsion Lab/NASA, USA
Marley Vellasco, Pontifical Catholic University of Rio de Janeiro, Brazil
Christian Woehler, DaimlerChrysler AG, Germany
Berend Jan van der Zwaag, University of Twente, The Netherlands

Contents

Hybrid Dynamic Systems in an Industry Design Application

Pieter J. Mosterman and Elisabeth M. O'Brien

The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760, USA
{pieter_j_mosterman,elisabeth.obrien}@mathworks.com

Abstract. The term *hybrid dynamic system* is a term for a mathematical system that combines behavior of a continuous nature with discontinuous changes. Such systems are often formed by the underlying computational representation of models used in the design of control and signal processing applications, for example in the automotive and aerospace industries. This paper outlines the benefits of Model-Based Design and illustrates how many different formalisms may be essential in model elaboration, such as time-based block diagrams, state transition diagrams, entity-flow networks, and multi-body diagrams. The basic elements of the underlying hybrid dynamic system computational representation are presented and it is shown how these elements combine to form different classes of behaviors that need to be handled for simulation.

Keywords: Model-Based Design; Hybrid Dynamic Systems; Hybrid Systems; Multi-Formalism Modeling; Embedded Control Systems; Networked Embedded Systems.

1 Introduction

Model-Based Design improves the design workflow of engineered sytems by employing computational models. In the embedded control systems realm, these models often are designed using Simulink® [20]. An embedded control system typically consists of a controller and a plant, where the plant is a physical system that is controlled to operate according to desired behavior.

The elements of Model-Based Design, illustrated in Fig. 1, can be summarized as:

- Executable specifications from models allow immediate feedback on the behavior of a specification, as opposed to documented behavior that often is misinterpreted.
- Design with simulation supports a faster exploration of the design space as opposed to constructing physical prototypes.
- Automatic code generation reduces the tedious and error-prone process of translating a design into a specification for the software engineers and manually writing the corresponding computer code.
- Test and verification can be performed in a much earlier stage in the design as a computational model is available with access to all internal variables, including those that may be difficult to obtain on a physical prototype.

A. Saad et al. (Eds.): Soft Computing in Industrial Applications, ASC 39, pp. 1–16, 2007.
springerlink.com

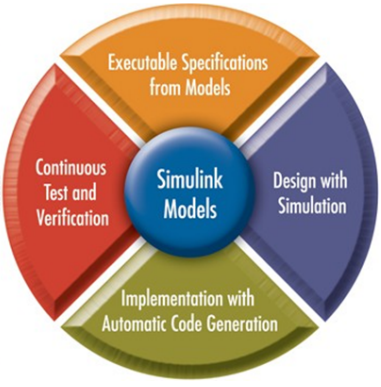


Fig. 1. Model-Based Design elements leverage Simulink® models

The adoption of Model-Based Design has enterprise-wide implications [1]. For example, the extensive use of models throughout the design process has created the desire to facilitate model reuse. This reuse, in turn, requires design tools that support the exchange of models between engineering teams. For example, to obtain a high-fidelity plant model, a SolidWorks [21] computer-aided design (CAD) model of the geometry can be exported into a SimMechanics [18, 25] multibody model of the dynamics. Thus, modeling effort is reused as models are shared across teams.

A controller model may initially be a discrete state based model that is then extended to include implementation effects such the validation of input data. This approach implies that an execution engine for a supporting tool set such as Simulink and Stateflow® [22] has to efficiently handle both a data driven approach as well as an event driven approach. Because of the widely differing execution semantics that different models may employ, execution engines are required to be versatile and powerful such that efficient algorithms, tailored to the needs of a specific model, can be invoked.

An important distinction in execution semantics can be made between those that require continuity of variables, possibly in higher derivatives, and those that allow discrete changes. Combining those two execution semantics results in *hybrid dynamic systems*, or *hybrid systems* for short (e.g., [2, 10, 23]).

The modeling formalisms that capture the discrete part of a hybrid system often are state transition diagrams [9], for example, a high-level language such as statecharts [7] may be employed. Statecharts are state transition diagrams that include language features such as hierarchy, parallelism, and event broadcasting.

The modeling formalisms that capture the continuous part of a hybrid system often are designed for plant modeling, i.e., the modeling of physics [5, 8], and they typically rely on differential equations, possibly combined with algebraic constraints.

The combination of state transition diagrams and differential and algebraic equations may be desired if, for example, there are widelydiffering time scales

at which physical phenomena occur. In such a situation, it may be beneficial to abstract fast continuous behavior into a discrete change. The slower continuous behavior is then modeled by differential equations, while the discrete behavior may be modeled by a state transition diagram [12].

For example, a nonelastic collision between two bodies can be modeled in detail by accounting for dissipation effects that occur from when the bodies initiate contact to when they achieve the same velocity. Alternatively, detailed behavior from the dissipative effects can be disregarded, and the velocities can be instantaneously set to be equal.

This paper provides the elements that constitute a hybrid dynamic system. Complications and idiosyncrasies in the behavior of such hybrid dynamic systems and an ontology of mode transition behavior are presented. It is illustrated how instantaneous changes in variables, in combination with the inequalities that define mode switching, can lead to rich and complex mode transition behavior [13].

Section 2 provides a more detailed introduction to Model-Based Design. Section 3 illustrates the use of Model-Based Design for a power window control system, which concretely shows a number of different modeling formalisms that are employed throughout the design. Section 4 introduces the underlying computational representation across different modeling formalisms as a hybrid dynamic system and discusses the characteristics of such a system. Section 5 presents the conclusions of this work.

2 Model-Based Design

The benefits of Model-Based Design are manifold and mostly stem from the use of computational technologies. In addition, rather than isolated usages of computational models, it is important that a tool infrastructure is available to move a model through the design stages while elaborating it along the way.

2.1 Why Model-Based Design?

Model-Based Design uses an executable specification, which facilitates communication across engineering groups and enables rapid design iterations which greatly decreases development time. This approach contrasts with a more traditional approach in which the specification typically consists of a paper document. The document needs to be shared among many engineers or groups of engineers, and is often miscommunicated or distributed copies are not kept up to date.

The model that results from an executable specification is not only the repository for all of the information about the concept and design but also the design implementation. Once the specification has been made executable, Model-Based Design enables the exploitation of simulation so that the design space can be searched for an optimal design efficiently. Moreover, this search may now be automated.

Following simulation, implementation is achieved through automatic code generation. Transforming a paper specification of a design into software such

as C-code is an error-prone process. Automatic code generation can reduce both design and hand-coding errors while substantially alleviating the tediousness of the coding task.

Model-Based Design further enables unambiguous communication between everyone involved in the overall design, within one company and across companies, such as between suppliers and the original equipment manufacturer (OEM). When everyone works off the same model, or at least an elaborated form of a core model, they can speak the same language and communicate more effectively.

Another key benefit of Model-Based Design is early test and verification. If a model is available early on in the design process, and it is executable, it is possible to design the tests to ensure that the final product complies with the original requirements based on the model. Therefore, design testing can be performed early on in the design process, as opposed to having to wait until the physical product has become available.

As a result, Model-Based Design eliminates the need for physical prototypes in the early design phases. Their use can be deferred much longer than in a traditional design approach, which decreases the reworking of a prototype because it has already been tested in much greater detail in a computational setting.

2.2 Practicing Model-Based Design

Model-Based Design relies heavily on model elaboration, as shown in Fig. 2. On the left of the diagram is the core control algorithm, which is often designed using synthesis techniques based on simplified plant models, such as low-order linear versions of more complex plant models.

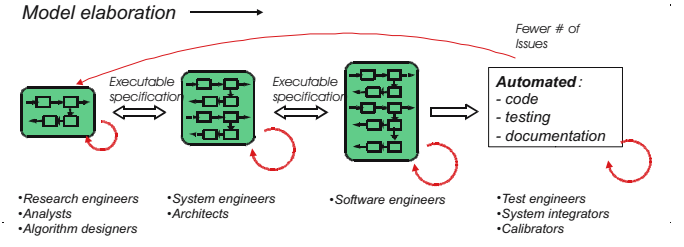


Fig. 2. Model elaboration

Once the core control algorithm has been derived, it is handed to the system engineers who embed it into an overall system. At this point, data validation, input/output (I/O) functionality, redundancy management, and testing functionality will be included.

The next step is implementation, in which the control algorithm needs to be coded in C, Ada, or any other desired target language, to embed the control algorithm into a physical environment as software that executes on a hardware target. This step is typically done by software engineers. Operating system

issues may arise here; for example, computations that have been designed for the algorithm as well as for the system must fit into the computational resources available. The algorithm may need to fit onto a number of microprocessors; there may be high priority tasks, low priority tasks, and different sample rates, which are all coded into tasks or multiple tasks; and it is necessary to verify and validate that the system still operates according to specification.

Finally, the system must be integrated with other systems that have been built. This requirement leads to the notion of "systems of systems." Using an automobile power window as an example, it may be necessary to validate that the window operates properly in concert with the electrical system by not drawing electrical power when the engine is started. This is achieved by combining and integrating the system of systems, as well as calibrating it to make sure that it operates properly.

Model elaboration, then, is the process of moving the model through a number of phases where increasing detail is included. This facilitates communication between the engineering teams responsible for the separate phases. As mentioned previously, data validation and analysis need to be performed, I/O and interfaces need to be established, and redundancy management all need to be included in the design. With Model-Based Design–and its use of executable models–testing happens every time a model is simulated, and thus is an integrated aspect of the design process. This integration enables continuous testing and validation that the model satisfies the requirements and is working according to specifications.

3 A Power Window

To provide a concrete example of the use of Model-Based Design, the design of a power window (see Fig. 3) is outlined. The power window is an example of Model-Based Design for embedded control system development from concept through to implementation. It illustrates the use of different modeling formalisms that have different models of computation, the combination of which results in a hybrid dynamic system.

3.1 System Requirements

Electronics are used in automobiles to control various functions such as the opening and closing of windows and sun-roof, adjusting the mirrors/headlights, and locking and unlocking the doors. These systems are subject to stringent operating constraints, as failure may result in dangerous and possibly life-threatening situations. Therefore, careful design and analysis is mandatory before deployment.

Some quantitative requirements for the control of a power window may be as follows:

- The window must be fully opened and closed within 4 s.
- If the down or up command is issued for at least 200 ms and at most 1 s the window has to be fully opened or closed, respectively (auto-up/auto-down).

Fig. 3. An automobile power window

- After a command is issued, the window must start moving within 200 ms.
- The force exerted in the presence of an object should be less than 100 N.
- When an object is present, the window should be lowered by approximately 10 cm.

3.2 Discrete Event Control

The core control algorithm is of a discrete event nature and best modeled by using a statechart. The statechart contains the basic states of the power window system: up, auto-up, down, auto-down, rest, and emergency. It models the state transitions between these states and accounts for the precedence of driver commands over the passenger commands. It also includes emergency behavior that is to be activated when an object is detected to be present between the window and the door frame while moving up. In the emergency state, the window is moved down by 10 cm.

While in the state in which the driver command is neutral, the passenger is in control and can command the window up or down. Figure 4 shows part of the Stateflow chart that switches between neutral, up, and down states, as commanded by the passenger, *passengerNeutral*, *passengerUp*, and *passengerDown*,

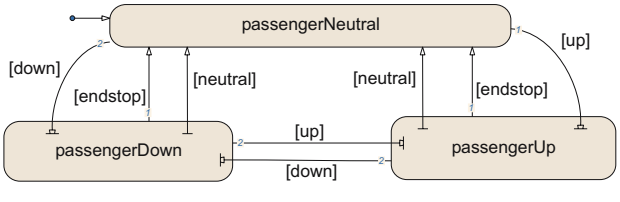


Fig. 4. A state transition diagram

respectively. The transitions between states are based on conditions *down*, *up*, *neutral*, and *endstop*. The statechart is executed periodically at a 10 ms rate and the conditions are evaluated at this rate. If one of them is true, the corresponding transition is taken, where the order of evaluation is explicitly shown by the numbers on the state transition arrows.

The *passengerDown* and *passengerUp* states contain subcharts that implement the auto-up and auto-down state transition logic.

Simulink enables testing the design with a variety of test vectors as inputs to the state machine. A model coverage report permits verification that the design is completely excited with the test vectors that have been employed, thereby showing that the design is void of hidden functionality. The generated report documents which transitions have been excited and which have not.

3.3 The Emergency Rollback

Further fulfillment of the requirements results in increased design complexity. Once the discrete event control has been designed and verified, it can be coupled to the continuous time plant model shown in Fig. 5 to ensure the window is retracted 10 cm upon detecting an object. The plant model contains two integrators. One computes velocity from the acceleration that results from the actuation force. The other computes the window position from its velocity. Viscous friction is modeled by the gain block that feeds back a friction force to be subtracted from the actuation force.

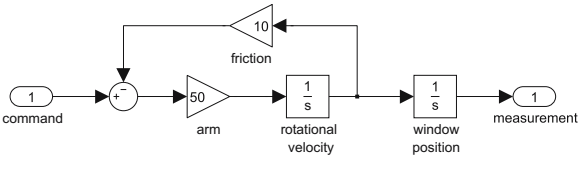


Fig. 5. A second-order plant model

Implementing additional functionality by embedding the statechart in a continuous time simulator converts the design from an untimed formalism to a timed formalism. By simulating the system, commanding the window up by switching the switch embedded in the driver switch block, the position signal can be analyzed to verify that the 10 cm requirement is satisfied.

3.4 Verifying the 100 N Force Limit

After an initial analysis of the discrete event control and continuous dynamics, a detailed plant model can be used to evaluate performance in a more realistic implementation. Models at such a level of detail are best designed in the power domain, i.e., as energy flow. This approach is facilitated by several domain specific blocksets.

Using a tool for modeling physical systems such as SimMechanics allows inertias, joints, and bodies to be used as basic elements of the modeling formalism. For example, Fig. 6 shows a SimMechanics model of the scissor-type lift mechanism that is used to move the window up and down and that is shown in Fig. 3. On the one end, a DC motor drives one of the two levers that constitute the scissor-like mechanism. Driving torque provided by the DC motor causes the worm part of a worm gear to rotate which, in turn, causes the lever to rotate.

The SimMechanics model shows the torque coming from the DC motor as an inport block on the left. This torque is used to actuate a rotational joint, which represents the worm part with inertia modeled by the *worm* rigid body block. The worm connects to the main gear through a gear ratio as modeled by the *worm gear* block. Both the worm and the main lever rotate with one degree of freedom relative to the door. The *main gear* lever body attaches to the bottom of the window by a *rotate & slide* joint, to ensure the attachment can move to the left or right as the window moves up and down. The angle of the main lever is measured, in this case for visualization purposes.

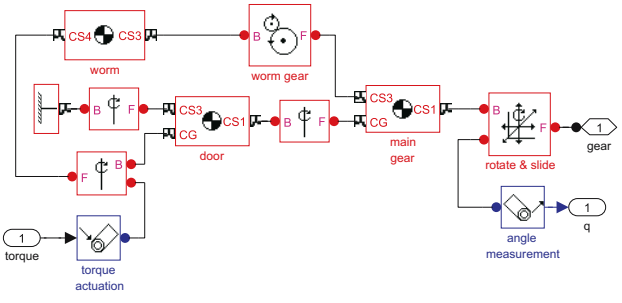


Fig. 6. A multibody diagram of the lift mechanism

Note that there is no direction associated with the connection of joints and bodies. Instead, a joint carries two variables: force and velocity. The modeler does not have to determine if the *main gear* block computes the force or the velocity. This is automatically derived by the compiler. Similarly, the DC motor model is designed using SimPowerSystems [19], and contains undirected connections in the electrical domain.

At this point in the design it becomes clear that the armature current drawn by the DC motor is the only available measurement. The control system as derived earlier now has to be modified to accommodate an input different from the window position. Instead, when the armature current is more than 1.7 A, an object is detected.

In Fig. 7 the force exerted by the window during a simulation of the window moving up is shown. At approximately 2.7 s, an obstacle is detected and the window is retracted by 10 cm. As shown in Fig. 7, the force, indeed, remains below 100 N, as per the requirement. An assertion check can be inserted so that if the window exerts a force above 100 N the simulation will stop. This step is done by way of

adding a *check static bound* block. Note that the force does fall below -100 N when the direction of motion is reversed, which does not violate the requirement and is safe because it is irrelevant how forcefully the window is being pulled down.

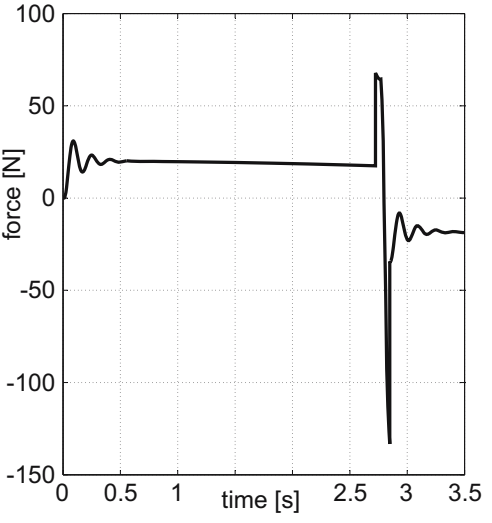


Fig. 7. Simulation of the force exerted by the window

3.5 Further Model Elaboration

Further model elaboration may include architectural elements such as the use of a controller area network (CAN) [4] bus to communicate the user command as input using some switch hardware to the hardware that controls the window movement. Communication is achieved by packaging the commands entered through the window control switches into a network frame that is sent to the window controller, which unpacks the frame to retrieve the command value. The CAN bus is modeled as an entity-flow network using SimEvents™ [17]. Part of this model is depicted in Fig. 8. It shows a write port that sends a prepared frame to a transmit buffer. A flow controller connects to a gating block to release frames for actual transmission. Once released, a frame is copied so as to make it available on the communication channel and to queue it so the channel state can be determined.

Network traffic is often best modeled as irregularly spaced in time. To efficiently simulate such behavior, a discrete event simulator typically employs an event calendar that captures the times when an event occurs [3]. Simulation then progresses in time by simply updating the current time with the time at which the earliest event on the calendar occurs. In some applications, this update may take place in the order of a hundred thousand times over the course of one simulation run. Numerical integration schemes are not required, which enhances the efficiency of the simulation significantly and allows handling a large number of discrete events.

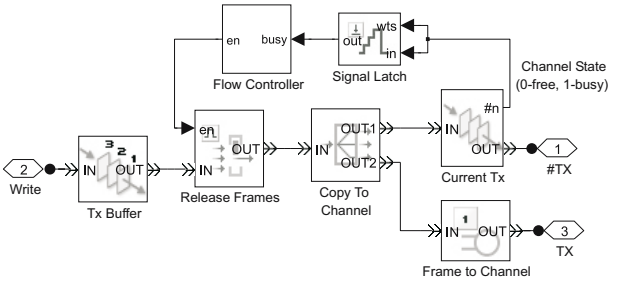


Fig. 8. An entity-flow network

Using SimEvents for the design of event driven systems allows convenient modeling of how packets of information on the network move and how other network traffic may affect the performance of the system. Because the CAN bus is shared and driver commands are put on the bus, the speed at which commands are retrieved by the control system, which moves the window, is affected by other network traffic.

Additional communication effects can be added until a sufficient level of detail is achieved. Controller code can then be automatically generated for any specific target platform, and coverage analysis tools can be used to ensure that the model is generating the desired output.

4 Hybrid Dynamic Systems

As illustrated by the power window design example in Section 3, many different modeling formalisms are typically employed in the design of an engineered system. In the case of the power window design, these formalisms include state transition diagrams (Fig. 4), time-based block diagrams (Fig. 5), multibody diagrams (Fig. 6), and entity-flow networks (Fig. 8).

4.1 Elements of a Hybrid Dynamic System

The semantics of the formalisms used are rather different from each other, varying from mechanical primitives to discrete states and transition elements, and are based on widely differing models of computation. For example, whereas time-based block diagrams may be used to model ordinary differential equation behavior, state transition diagrams may be used to capture finite state machine behavior. Similarly, multibody diagrams may be based on differential and algebraic equations, while entity-flow networks may rely on discrete-event models of computation.

An important aspect of these different models of computation is whether state behavior is allowed to exhibit discontinuous changes or whether state behavior must be continuous, possibly with further constraints on higher order derivatives.

A mathematical system that contains both classes of behavior is often referred to as a *hybrid dynamic system*, or a *hybrid system* for short. In this paper, state variable behavior with continuity constraints corresponds to differential equation behavior as captured by, for example, time-based block diagrams (Fig. 5) and multibody diagrams (Fig. 6); state behavior that may be discontinuous corresponds to discrete event behavior as captured by, for example, state transition diagrams (Fig. 4) and entity-flow networks (Fig. 8).

To illustrate a hybrid dynamic system, consider a model of the dynamics of the power window in Fig. 3, presented in Fig. 9. Here, the window is modeled as a rigid body that moves in the vertical direction. When the window moves between the bottom and the top of the door frame, the window movement is determined by the net force acting on it, which derives from the actuator force combined with the frictional force and gravity.

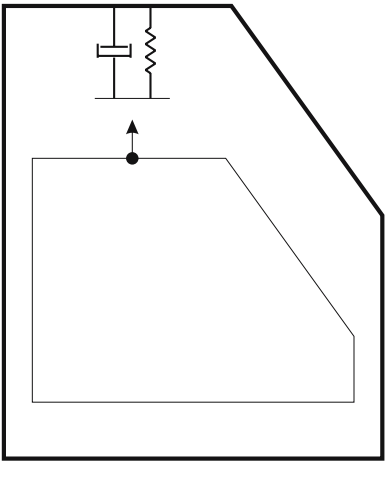


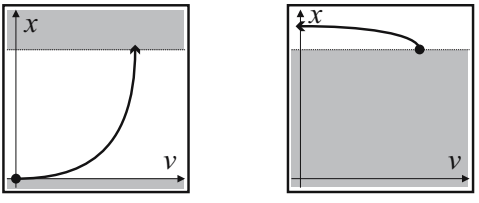
Fig. 9. A power window system model

As illustrated in Fig. 9, the top of the door frame can be modeled as a stiff spring-damper system. This system acts as an additional force when the window reaches the top of the door frame. The force is composed of a viscous (damping) force and a displacement (spring) force. The spring-damper force builds up very quickly to balance the combination of the actuator force, the frictional force, and gravity. When a balance of forces is achieved, the window stops moving.[1]

The window behavior can now be schematically captured by the state spaces in Fig. 10. The two state spaces correspond to the two *modes* of operation of the window. In Fig. 10(a), the behavior of the window when it is between the

[1] Note that typically the actuator force will be turned off when the window reaches the top of the door frame. Such feedback aspects are not considered here to avoid unnecessary complexity in the illustrative behavior.

bottom and the top of the door frame is shown, which is called the *free* mode. In this mode, the actuator force causes a positive velocity, v, according to which the window starts to move up and increase its position, x. In Fig. 10(b), the behavior of the window when it is moving against the top of the door frame is shown, which is called the *stuck* mode. In this mode, the door frame, modeled by the spring-damper system, exerts a rapidly increasing force to bring the window movement to a halt.



(a) Free (b) Endstop

Fig. 10. Modes of behavior for a power window

The state space behavior in Fig. 10 shows a number of important elements that are present in hybrid dynamic systems [14, 15]:

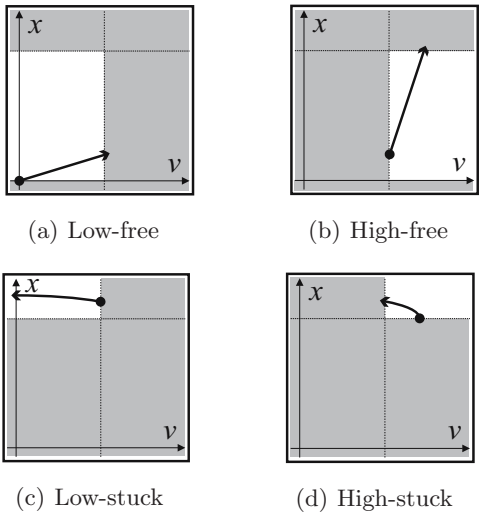
- Differential equations determine the window behavior in continuous time. For example, for the power window, $F_{net} = m_{window}\dot{v}$, i.e., the net force, F_{net}, acting on the window with mass m_{window} corresponds to the window acceleration $\dot{v}$, where the dot operator is used to express differentiation with respect to time.
- Inequality constraints determine where the differential equations are operational. This is called the *operational area* (or *patch* [6]). For the power window, the differential equations for the *free* mode are operational when $x > x_{bottom}$ and $x < x_{top}$, with x being the window position and x_{bottom} and x_{top} the values corresponding to the bottom and top of the door frame.
- A mode transition function determines which mode is active. For the power window, the mode transition function captures the change from *free* to *stuck* when $x \geq x_{top}$. The mode transition function is often provided as a state transition diagram.

4.2 Further Model Abstraction

For many analysis and synthesis tasks, abstractions are applied to the model to obtain a simplified representation. The abstractions applied determine the level of detail to capture versus the level of computational complexity suited for the algorithms employed.

For example, the state space behavior in Fig. 10 may be simplified, as shown in Fig. 11. Here, the differential equation behavior that couples velocity and position in the *free* mode is partitioned into two piecewise linear modes of operation,

shown in Fig. 11(a) and Fig. 11(b). Though this may reduce the complexity of the mathematics involved in computing the up movement, it requires the derivation of inequalities to properly define the operational areas.



(a) Low-free (b) High-free

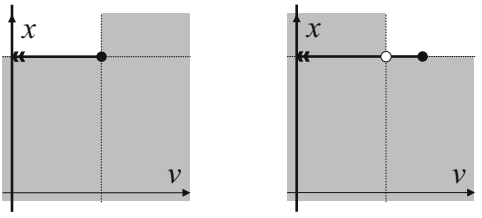
(c) Low-stuck (d) High-stuck

Fig. 11. Simplified continuous-time model of the power window behavior

The model can be further simplified by removing the stiff behavior caused by the spring-damper system in order to quickly reduce the window velocity to 0, shown in Fig. 11(c) and Fig. 11(d) for the low and high velocity partitioning. Instead, the window velocity may be immediately set to 0, resulting in a nonelastic collision model. This instantaneous change in velocity is shown in Fig. 12 as a line with a double arrow head. An important observation is that the instantaneous change covers two modes: it is initiated in the *high-stuck* mode and terminates in the *low-stuck* mode. This exemplifies that the instantaneous change may exit the operational area. In general, the point in the state space where an instantaneous change leaves the operational area indicated by the open circle in Fig. 12(b), is difficult to determine.

The simplification in Fig. 12 illustrates another important element of hybrid dynamic systems, the *admissible space*. Note that, in general, a system of differential and algebraic equations may contain variables that are operated on by a time differentation, but that are not state variables. These variables are sometimes referred to as *generalized state variables* [24]. Referring to Fig. 12 of the power window example, even though the window position and velocity are two generalized state variables, the window velocity is required to be 0, leaving only one degree of freedom, or state, for the dynamic behavior, i.e., the window position.

The space that represents the degree of freedom is called the *admissible space*. In Fig. 12 it is the line at which the velocity is 0, indicated by the thick line. In Fig. 11, the admissible space is the entire state space, indicated by the thick border.



(a) Low-stuck (b) High-stuck

Fig. 12. Abstraction classes for endstop models

4.3 Mode Transition Sequences

In Fig. 12(b) the admissible space lies outside of the operational area, and thus this mode has to be departed immediately when it is reached. In general, an important characteristic of hybrid dynamic systems is that one mode change may immediately be followed by another mode change without any continuously evolving behavior in between. This is illustrated by the scenario in Fig. 13. Once the window reaches the top of the door frame, it changes from the *high-free* mode to the *high-stuck* mode. Before another mode of continuously evolving behavior is arrived at, a consecutive mode change moves the hybrid dynamic system into the *low-stuck* mode.

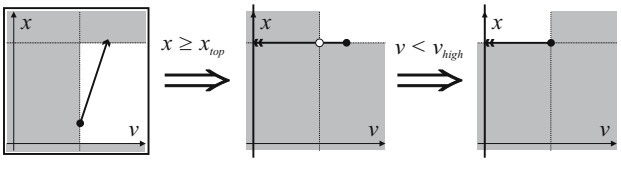


Fig. 13. A sequence of mode transitions at one point in time

In previous work [13, 16] an ontology of state space transition behavior has been developed. In this ontology, an intermediate mode that is only active at a given point in time is either called:

- A *pinnacle*, which causes a change in the state. This situation happens when the admissible space is outside of the operational area, and the mode is entered with a state outside of the admissible space.
- A *mythical mode*, which has no effect on the state. This situation happens when the mode is entered with a state outside of the operational area and within the admissible space.

To support computational simulation, these different classes of behavior have to be properly handled. Details on approaches and algorithms are discussed elsewhere [11, 14].

5 Conclusions

Model-Based Design is increasingly adopted in industry to aid in the design of engineered systems. The use of computational models offers a variety of advantages over the use of paper documents and physical prototypes. An important aspect of computational models is that they typically can be executed so the behavior of a design can be studied by means of simulation.

This paper has given an overview of Model-Based Design and introduced some of the benefits that can be derived from it. A concrete example has been given by illustrating elements of the design of a power window control system. This example motivated the need to support widely differing formalisms such as state transition diagrams, time-based block diagrams, entity-flow networks, and multibody diagrams.

The execution semantics of each of these formalisms are very different and require different technology for simulation. A general classification can be made in terms of behavior that is continuous in time and behavior that may be discontinuous. Combining formalisms with elements in both classes leads to *hybrid dynamic systems*.

The basic elements of a hybrid system and an overview of hybrid dynamic system behavior in geometrical terms was given.

Acknowledgments

MATLAB, Simulink, Stateflow, Handle Graphics, Real-Time Workshop, and xPC TargetBox are registered trademarks and SimBiology, SimEvents, and SimHydraulics are trademarks of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.

References

[1] Paul Barnard. Graphical techniques for aircraft dynamic model development. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*. Providence, Rhode Island, August 2004. CD-ROM.

[2] Maria Domenica Di Benedetto and Alberto L. Sangiovanni- Vincentelli, editors. Hybrid Systems: Computation and Control, volume 2034 of Lecture Notes in Computer Science. Springer- Verlag, March 2001.

[3] Randy Brown. Calendar queues: A fast 0(1) priority queue im- plementation for the simulation event set problem. Communi- cations of the ACM, 31(10):12201227, 1988.

[4] CAN specification. Technical Report, 1991. Robert Bosch GmbH.

[5] F.E. Cellier, H. Elmqvist, and M. Otter. Modelling from physical principles. In W.S. Levine, editor, The Control Handbook, pages 99107. CRC Press, Boca Raton, FL, 1996.

[6] John Guckenheimer and Stewart Johnson. Planar hybrid sys- tems. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Shankar Sastry, editors, Hybrid Systems II, volume 999, pages 202225. Springer-Verlag, 1995. Lecture Notes in Computer Science.
[7] David Harel. Statecharts: A visual formalism for complex sys- tems. Science of Computer Programming, 8:231274, 1987.
[8] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. Systems Dynamics: A Unified Approach. John Wiley and Sons, New York, 2 edition, 1990.
[9] Zvi Kohavi. Switching and Finite Automata Theory. McGraw- Hill, Inc., New York, 1978.
[10] Nancy Lynch and Bruce Krogh, editors. Hybrid Systems: Com- putation and Control, volume 1790 of Lecture Notes in Com- puter Science. Springer-Verlag, March 2000.
[11] Pieter J. Mosterman. An overview of hybrid simulation phe- nomena and their support by simulation packages. In Frits W. Vaandrager and Jan H. van Schuppen, editors, Hybrid Systems: Computation and Control, volume 1569, pages 164177. Lecture Notes in Computer Science; Springer-Verlag, March 1999.
[12] Pieter J. Mosterman. HyBrSim a modeling and simulation environment for hybrid bond graphs. Journal of Systems and Control Engineering, 216:3546, 2002. special issue paper.
[13] Pieter J. Mosterman. Mode transition behavior in hybrid dy- namic systems. In Proceedings of the 2003 Winter Simulation Conference, pages 623631, New Orleans, LA, December 2003. invited paper.
[14] Pieter J. Mosterman. Hybrid dynamic systems: Modeling and execution. In Paul A. Fishwick, editor, Handbook of Dynamic System Modeling, chapter 15, pages 15-115-23. CRC Press LLC, Boca Raton, FL, 2007.
[15] Pieter J. Mosterman and Gautam Biswas. A hybrid modeling and simulation methodology for dynamic physical systems. SIM- ULATION: Transactions of The Society for Modeling and Sim- ulation International, 178(1):517, January 2002.
[16] Pieter J. Mosterman, Feng Zhao, and Gautam Biswas. An on- tology for transitions in physical dynamic systems. In AAAI98, pages 219224, July 1998.
[17] SimEvents. SimEvents Users Guide. The MathWorks, Natick, MA, March 2006.
[18] SimMechanics. SimMechanics Users Guide. The MathWorks, Natick, MA, March 2006.
[19] SimPowerSystems. SimPowerSystems Users Guide. The Math- Works, Natick, MA, March 2006.
[20] Simulink. Using Simulink. The MathWorks, Inc., Natick, MA, March 2006.
[21] SolidWorks. Introducing SolidWorks. SolidWorks Corporation, Concord, MA, 2002.
[22] Stateflow. Stateflow Users Guide. The MathWorks, Natick, MA, March 2006.
[23] Frits W. Vaandrager and Jan H. van Schuppen, editors. Hy- brid Systems: Computation and Control, volume 1569 of Lecture Notes in Computer Science. Springer-Verlag, March 1999.
[24] George C. Verghese, Bernard C. Levy, and Thomas Kailath. A generalized state-space for singular systems. IEEE Transactions on Automatic Control, 26(4):811831, August 1981.
[25] Giles D. Wood and Dallas C. Kennedy. Simulating mechani- cal systems in simulink with simmechanics. Technical Report 91124v00, The MathWorks, Inc., Natick, MA, 2003.

Part I

Soft Computing in Computer Graphics, Imaging and Vision

Object Recognition Using Particle Swarm Optimization on Fourier Descriptors

Muhammad Sarfraz and Ali Taleb Ali Al-Awami

[1] Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
sarfraz@kfupm.edu.sa
[2] Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran, 31261 Saudi Arabia
aliawami@kfupm.edu.sa

Abstract. This work presents study and experimentation for object recognition when isolated objects are under discussion. The circumstances of similarity transformations, presence of noise, and occlusion have been included as the part of the study. For simplicity, instead of objects, outlines of the objects have been used for the whole process of the recognition. Fourier Descriptors have been used as features of the objects. From the analysis and results using Fourier Descriptors, the following questions arise: What is the optimum number of descriptors to be used? Are these descriptors of equal importance? To answer these questions, the problem of selecting the best descriptors has been formulated as an optimization problem. Particle Swarm Optimization technique has been mapped and used successfully to have an object recognition system using minimal number of Fourier Descriptors. The proposed method assigns, for each of these descriptors, a weighting factor that reflects the relative importance of that descriptor.

Keywords: curve fitting, NURBS, approximation, simulated evolution, algorithm.

1 Introduction

Fourier descriptors [1, 2, 14], like Moment descriptors [9], have been frequently used as features for image processing, remote sensing, shape recognition and classification. Fourier Descriptors can provide characteristics of an object that uniquely represent its shape. Several techniques have been developed that derive invariant features from Fourier Descriptors for object recognition and representation [1-5, 14]. These techniques are distinguished by their definition, such as the type of data exploited and the method for deriving invariant values from the image Fourier Descriptors.

Granlund [1] introduced Fourier descriptors using complex representation in 1972. This method ensures that a closed curve will correspond to any set of descriptors. The Fourier descriptors have useful properties [3, 4]. They are invariant under similarity transformations like translation, scaling and rotation. The objects having these kind of transformations can be easily recognized using some recognition algorithms with Fourier descriptors as invariant features. For example, the Fourier descriptors, of the boundary [11-13], for recognizing closed contours is proposed in [5]. However,

A. Saad et al. (Eds.): Soft Computing in Industrial Applications, ASC 39, pp. 19–29, 2007.
springerlink.com