CIF 3: Model-Based Engineering of Supervisory Controllers

D.A. van Beek, W.J. Fokkink, D. Hendriks, A. Hofkamp,
J. Markovski, J.M. van de Mortel-Fronczak,
and M.A. Reniers

Manufacturing Networks Group
Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract. The engineering of supervisory controllers for large and complex cyber-physical systems requires dedicated engineering support. The Compositional Interchange Format language and toolset have been developed for this purpose. We highlight a model-based engineering framework for the engineering of supervisory controllers and explain how the CIF language and accompanying tools can be used for typical activities in that framework such as modeling, supervisory control synthesis, simulation-based validation, verification, and visualization, real-time testing, and code generation. We mention a number of case studies for which this approach was used in the recent past. We discuss future developments on the level of language and tools as well as research results that may be integrated in the longer term.

1 Introduction

A supervisory controller coordinates the behavior of a (cyber-physical) system from discrete-event observations of its state. Based on such observations the supervisory controller decides on the activities that the uncontrolled system can safely perform or on the activities that (are more likely to) lead to acceptable system behavior. Engineering of supervisory controllers is a challenging task in practice, amongst others because of the high complexity of the uncontrolled system.

In model-based engineering, models are used in the design process, instead of directly implementing a solution. The Compositional Interchange Format (CIF) is an automata-based modeling language that supports the entire model-based engineering development process of supervisory controllers, including modeling, supervisory controller synthesis (deriving a controller from its requirements), simulation-based validation, verification, and visualization, real-time testing, and code generation. CIF 3 is a substantially enhanced new version of CIF, after CIF 1 [BRRS08] and CIF 2 [NBR12]. It has been improved based on feedback from industry, as well as new theoretical advances. The various versions of CIF have been developed in European projects HYCON, HYCON2, Multiform, and C4C. CIF is actively being developed by the Manufacturing Networks Group[1] of the

[1] Until recently the group was named Systems Engineering Group.

E. Ábrahám and K. Havelund (Eds.): TACAS 2014, LNCS 8413, pp. 575–580, 2014.

Mechanical Engineering department, at the Eindhoven University of Technology (TU/e) [BHSR13]. The CIF tooling (see `cif.se.wtb.tue.nl`) is available under the MIT open source license (see `opensource.org/licenses/MIT`).

In Section 2, we introduce a simplified version of the framework for model-based engineering of supervisory controllers. In Section 3, we outline the role CIF and its related tools play in this framework. The most prominent features of CIF and its tools are highlighted there. In Section 4, we briefly discuss some industrial cases where the framework and tooling have been applied. Finally, in Section 5, we present a number of enhancements that are being considered for future addition to the CIF language and its tool set.

2 Model-Based Engineering of Supervisory Controllers

Fig. 1 depicts an overview of model-based engineering of supervisory controllers. Our starting point is a model (*uncontrolled hybrid plant*) of the uncontrolled system. The goal is to obtain a *supervisory controller* either by supervisory controller synthesis or by design. A *hybrid observer* forms an interface between the plant model and its supervisory controller. The first purpose of the observer is to interface the variable-based continuous world of the plant to the event-based world of the discrete-event controller. Its second purpose is the generation of additional events, from the state observed at the hybrid plant. They can be interpreted as *virtual sensors* by the controller, abstracting away timed behavior. Examples are a timeout, or an event that signals that a certain combination of values of physical quantities has occurred.

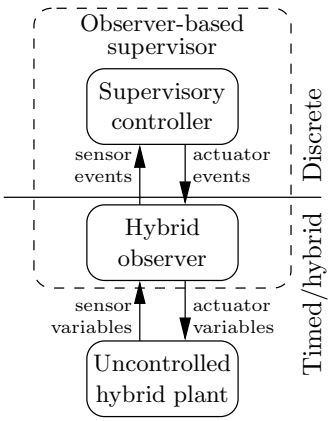

Fig. 1. Overview of model-based engineering of supervisory controllers

Fig. 2 depicts the workflow of the simplified framework for model-based engineering of supervisory controllers. First, the modeler manually designs an uncontrolled hybrid plant model, and a hybrid observer model. Next, from these two models, an abstraction of the uncontrolled system (*uncontrolled discrete-event plant*) is manually created.

From the uncontrolled discrete-event plant and discrete-event *control requirements*, a discrete-event supervisory controller is synthesized (generated). This controller is safe by construction. To also ensure that all relevant behavior is present, additional liveness verification can be performed on the supervisory controller and the uncontrolled discrete-event plant. For timed verification, the uncontrolled hybrid plant and hybrid observer can be used instead of the uncontrolled discrete-event plant. The automated synthesis and verification enable the designer to perform rapid iterative corrections and improvements of the

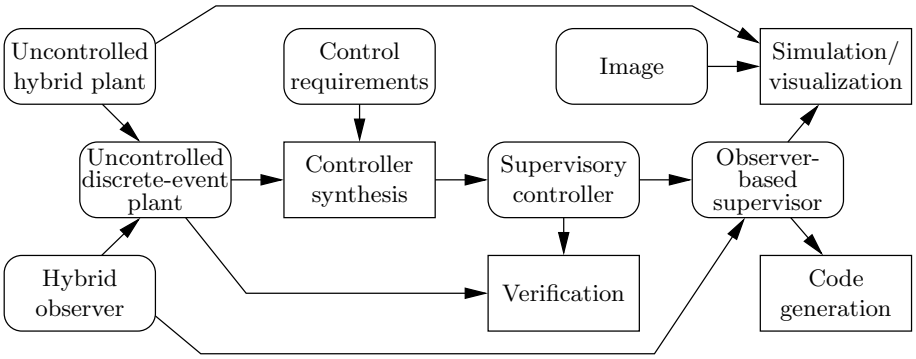

Fig. 2. Workflow for model-based engineering of supervisory controllers

plant model and the control requirements. The supervisory controller is combined with the hybrid observer, resulting in the actual controller (*observer-based supervisor*). This model is used for model-based validation, by means of real-time interactive simulation and visualization, based on a user-supplied *image* of the system. This brings a higher confidence that the models fulfill expected properties. The mentioned simulation-based visualization can also be used for validating the other models, such as the discrete-event and hybrid plant models.

As a final step, actual real-time control code is generated, for the implementation of the controller.

3 The Role of the CIF Language and Tools

The CIF language (see [BHSR13]) is based on networks of hybrid automata with invariants, and non-linear and/or discontinuous differential algebraic equations. The abilitiy of CIF to model large-scale systems is due to the orthogonal combination of parametrized process definition and instantiation (reuse, hierarchy), grouping of arbitrary components in sub-scopes, and an import mechanism. Components (automata) in a CIF model can interact in several different ways: multi-party synchronization via shared events, allowing communication via shared data; monitor automata to provide the functionality of nonblocking input events as defined in input-output automata [LSV01]; shared variables (local write, global read). Furthermore, CIF supports urgent events (events that must happen as soon as they are enabled by all synchronizing automata), a rich set of data types and expressions (e.g. lists, sets, dictionaries, tuples), functions, stochastic distributions, conditional updates, and multi-assignments.

CIF is well suited to model plants and supervisors in the application domain of cyber-physical systems (the blocks *uncontrolled hybrid plant*, *uncontrolled discrete-event plant*, *hybrid observer*, and *control requirements* in Fig. 2) as collections of automata using both discrete events and continuous-time behavior (see the cases mentioned in Section 4) in the same style as hybrid automata [ACH+95, Hen00, AGH+00, LSV01]. Developing a CIF model for the uncontrolled hybrid plant is an

iterative process of developing a model and using simulation and visualization to increase confidence in the model.

The main difference between CIF and the other currently available hybrid automata-based languages and toolsets, is that CIF covers the complete integrated tool chain for the development of supervisory controllers for complex cyber-physical systems. For the specification of CIF models for plants, requirements, observers, etc, an Eclipse-based (`eclipse.org`) textual editor is available, which features syntax highlighting, and continuous background syntax and type checking.

An important feature of the CIF toolset is the simulator. It can be employed to validate each of the CIF models mentioned before in isolation. Additionally it may be used to validate the controller when put in the context of the uncontrolled hybrid plant (as indicated in Fig. 2). Based on a CIF model, the simulator generates code for high-performance visual simulation. The simulator allows interactive exploration of the behavior of the controlled system, by using the interactive visualization-based simulation mentioned in the previous section. This requires user-supplied SVG vector images (`w3.org/TR/SVG11`). The simulator is highly configurable and versatile, allowing for automatic testing of various use cases, and the production of various forms of output.

To support event-based supervisory controller synthesis, CIF features plant and requirement automata, marker predicates [CL07], as well as controllable and uncontrollable events. Furthermore, the tools include efficient implementations of the traditional event-based supervisory controller synthesis algorithms as presented in [WR87, CL07].

For verification, a transformation of CIF models to Uppaal [NRS+11] is available. Compositionality has been a central concern when designing CIF, because a compositional semantics facilitates property-preserving model transformations and compositional verification techniques. Currently, over twenty transformations for various purposes are available ([BHSR13]), some via CIF 2.

Interoperability with other languages and tools is achieved by means of model transformations, external functions, and co-simulation via the Matlab/Simulink S-function interface [The05]. Programmable Logic Controller (PLC) code generation conforming to the IEC 61131-3 standard [JT10] is also available, allowing for implementation of CIF controllers in actual systems.

4 Applications

CIF has been used in an industrial context for a number of years now. We mention some of the more prominent applications.

- Development of a coordinator for maintenance procedures for a high-tech Océ printer [MJB+10]
- Improving evolvability of a patient communication control system using state-based supervisory control synthesis [TBR12]
- Application of supervisory control theory to theme park vehicles [FMSR12]
- Supervisory control of MRI subsystems [Geu12]
- Design of a supervisory controller for a Philips MRI-scanner [Dij13]

– Design and real-time implementation of a supervisory controller for baggage handling [Kam13]

Although these projects showed different parts of the suggested model-based engineering framework, together they demonstrate all mentioned activities.

Currently, a project on control and performance analysis of wafer flow in wafer handlers is carried out at ASML, and a project on design and implementation of a certified controller with multiple control levels for a baggage handling system is carried out for Vanderlande Industries. Also in these projects supervisory controller synthesis and verification are considered.

5 Future Developments

CIF is constantly being improved and extended. A planned extension of CIF is the addition of point-to-point communication by means of channels. Our experience with industrial cases has shown that these are well suited to model physical movements of objects. The channels will be fully integrated into the language. For instance, supervisors will be able to prohibit channel communications.

For verification, we intend to support a larger class of CIF models for the transformation to Uppaal. We will develop model transformations to other model checking tools as experimented with in [MR12a]. For performance analysis we are considering model transformations to MRMC and/or PRISM [MR12b, MER13].

The Manufacturing Networks Group also works on extensions of supervisory control theory, such as the domain of plant models to which synthesis may be applied, and the expressivity of the logic for requirements. See [HFR13] for a first publication of this line of research. As soon as such extensions reach an acceptable level of maturity, they are incorporated in the CIF tooling.

Acknowledgments. The research leading to these results has received funding from the EU FP7 Programme under grant agreements no. FP7-ICT-223844 (C4C), no. FP7-ICT-224249 (MULTIFORM), and the Network of Excellence HYCON (IST-2002-2.3.2.5). The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 257462 HYCON2 Network of excellence.

References

[ACH+95] Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science 138(1), 3–34 (1995)

[AGH+00] Alur, R., Grosu, R., Hur, Y., Kumar, V., Lee, I.: Modular specification of hybrid systems in CHARON. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 6–19. Springer, Heidelberg (2000)

[BHSR13] van Beek, D.A., Hendriks, D., Swartjes, L., Reniers, M.A.: Report on the extensions of the CIF and transformation algorithms. Technical Report HYCON Deliverable D6.2.4 (2013)

[BRRS08] van Beek, D.A., Reniers, M.A., Rooda, J.E., Schiffelers, R.R.H.: Concrete syntax and semantics of the Compositional Interchange Format for hybrid systems. In: IFAC World Congress 2008, pp. 7979–7986, IFAC (2008)
[CL07] Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems, 2nd edn. Springer (2007)
[Dij13] van Dijk, D.: Supervisory control of a Philips MRI-scanner. Master's thesis, Eindhoven University of Technology (2013)
[FMSR12] Forschelen, S.T.J., van de Mortel-Fronczak, J.M., Su, R., Rooda, J.E.: Application of supervisory control theory to theme park vehicles. Discrete Event Dynamic Systems 22(4), 511–540 (2012)
[Geu12] Geurts, J.W.P.: Supervisory control of MRI subsystems. Master's thesis, Eindhoven University of Technology (2012)
[Hen00] Henzinger, T.A.: The theory of hybrid automata. In: Verification of Digital and Hybrid Systems. NATO ASI Series F: Computer and Systems Science, vol. 170, pp. 265–292. Springer (2000)
[HFR13] van Hulst, A., Fokkink, W.J., Reniers, M.A.: Maximal synthesis for Hennessy-Milner Logic. In: ACSD 2013, pp. 1–10. IEEE (2013)
[JT10] John, K.H., Tiegelkamp, M.: IEC 61131-3: Programming Industrial Automation Systems, 2nd edn. Springer (2010)
[Kam13] Kamphuis, R.H.J.: Design and real-time implementation of a supervisory controller for baggage handling at Veghel Airport. Master's thesis, Eindhoven University of Technology (2013)
[LSV01] Lynch, N., Segala, R., Vaandrager, F.: Hybrid I/O automata revisited. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 403–417. Springer, Heidelberg (2001)
[MER13] Markovski, J., Estens Musa, E.S., Reniers, M.A.: Extending a synthesis-centric model-based systems engineering framework with stochastic model checking. ENTCS 296, 163–181 (2013)
[MJB+10] Markovski, J., Jacobs, K.G.M., van Beek, D.A., Somers, L.J.A.M., Rooda, J.E.: Coordination of resources using generalized state-based requirements. In: WODES 2010, pp. 300–305. IFAC (2010)
[MR12a] Markovski, J., Reniers, M.A.: An integrated state- and event-based framework for verifying liveness in supervised systems. In: ICARCV 2012, pp. 246–251. IEEE (2012)
[MR12b] Markovski, J., Reniers, M.A.: Verifying performance of supervised plants. In: ACSD 2012, pp. 52–61. IEEE (2012)
[NBR12] Nadales Agut, D.E., van Beek, D.A., Rooda, J.E.: Syntax and semantics of the compositional interchange format for hybrid systems. Journal of Logic and Algebraic Programming 82(1), 1–52 (2012)
[NRS+11] Nadales Agut, D.E., Reniers, M.A., Schiffelers, R.R.H., Jørgensen, K.Y., van Beek, D.A.: A semantic-preserving transformation from the Compositional Interchange Format to UPPAAL. In: IFAC World Congress 2011, pp. 12496–12502, IFAC (2011)
[TBR12] Theunissen, R.J.M., van Beek, D.A., Rooda, J.E.: Improving evolvability of a patient communication control system using state-based supervisory control synthesis. Advanced Engineering Informatics 26(3), 502–515 (2012)
[The05] The MathWorks, Inc. Writing S-functions, version 6 (2005), `http://www.mathworks.com`
[WR87] Wonham, W.M., Ramadge, P.J.: On the supremal controllable sublanguage of a given language. SIAM Journal on Control and Optimization 25(3), 637–659 (1987)