Eclipse ESCET$^{\text{TM}}$: The Eclipse Supervisory Control Engineering Toolkit

W.J. Fokkink[1,2](✉), M.A. Goorden[3,4], D. Hendriks[5,6], D.A. van Beek[1], A.T. Hofkamp[1], F.F.H. Reijnen[7], L.F.P. Etman[1], L. Moormann[1], J.M. van de Mortel-Fronczak[1], M.A. Reniers[1], J.E. Rooda[1], L.J. van der Sanden[5], R.R.H. Schiffelers[1,8], S.B. Thuijsman[1], J.J. Verbakel[1], J.A. Vogel[4]

[1] Eindhoven University of Technology, Eindhoven, The Netherlands
[2] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
w.j.fokkink@vu.nl
[3] Aalborg University, Aalborg, Denmark
[4] Rijkswaterstaat, Utrecht, The Netherlands
[5] TNO-ESI, Eindhoven, The Netherlands
[6] Radboud University, Nijmegen, The Netherlands
[7] Vanderlande Industries, Veghel, The Netherlands
[8] ASML, Veldhoven, The Netherlands

Abstract. The Eclipse Supervisory Control Engineering Toolkit (ESCET$^{\text{TM}}$) is an open-source project to provide a model-based approach and toolkit for developing supervisory controllers , targeting their entire engineering process. It supports synthesis-based engineering of supervisory controllers for discrete-event systems, combining model-based engineering with computer-aided design to automatically generate correct-by-construction controllers. At its heart is supervisory controller synthesis, a formal technique for the automatic derivation of supervisory controllers from the unrestricted system behavior and system requirements. Vital for the future development of these techniques and tools is the ESCET project's open environment, allowing industry and academia to collaborate on creating an industrial-strength toolkit. We report on some crucial developments of the toolkit in the context of research projects with Rijkswaterstaat and ASML that have considerably improved its capability to deal with the complexity of real-life systems as well as its usability.

1 Introduction

A supervisory controller, supervisor for short, coordinates the behavior of a cyber-physical system according to discrete-event observations of its system behavior. Based on such observations, the supervisor decides which events the system can safely perform and which events must be disabled, because they would lead to violations of requirements or to a blocking state. Engineering of supervisors is a challenging task, due to the high complexity of real-life discrete-event systems.

Supervisory control theory [21] underpins a model-based technique for automatically deriving a model of a supervisor from models of the uncontrolled system behavior and the system's requirements, such as functional or safety-related requirements that intend to rule out all undesired behavior. This is achieved by

© The Author(s) 2023

disabling *controllable* (output) events, such as starting a motor. Supervisors exert no control over *uncontrollable* (input) events, such as sensor reports.

The Eclipse Supervisory Control Engineering Toolkit (ESCET$^{\text{TM}}$, pronounced *èsèt*) project,[1][2] provides a model-based approach and toolkit for the development of supervisors. It targets the entire engineering process for the development of supervisors, including modeling, synthesis, simulation-based validation and visualization, formal verification, real-time testing, and code generation. This entire process is supported by CIF [1],[3] featuring an automata-based modeling language for convenient specification of large-scale systems, and tools that support synthesis-based engineering (SBE). SBE is an engineering approach to design and implement supervisors that combines model-based engineering with computer-aided design to produce correct-by-construction controllers, by automating the engineering process as much as possible. While not detailed further in this paper, the ESCET project also comprises Chi [28], a hybrid language and toolset for modeling and simulation, developed by the same research group that developed CIF, and the ToolDef scripting language for the definition and execution of model-based toolchains, useful for combining different ESCET tools.[4]

The ESCET project, an Eclipse Foundation open-source project since 2020, builds upon decades of research and tool development at Eindhoven University of Technology. Vital for the evolvement from an academic into an industrially applicable toolkit are the years-long ongoing research collaborations with industry, including Rijkswaterstaat [7], ASML [27], and Vanderlande [29]. Rijkswaterstaat, part of the Dutch Ministry of Infrastructure and Water Management, is responsible for infrastructure in the Netherlands, including roads, bridges, tunnels, and waterway locks. ASML is an innovation leader in the semiconductor industry, providing chipmakers with all they need to mass produce patterns on silicon through lithography. Vanderlande is a market leader in logistic process automation for the warehousing, airport and parcel sectors. The quality of supervisory control software for such systems impacts their availability and reliability. Synthesis-based engineering allows for automation, modularization, and standardization, increasing quality and evolvability and decreasing life-cycle costs.

With the move to the Eclipse Foundation, and supported by the Eclipse Foundation's principles of transparency, openness, meritocracy and vendor-neutrality, the ESCET project aims to be an open environment and a growing community. It allows interested parties, such as academic and applied research institutes, industrial partners and tool vendors, to collaborate on and profit from further tool development for the model-based construction of supervisors. Furthermore, the project's open nature allows any vendor to develop commercial tool support.

We report on some crucial developments of the toolkit that have considerably improved its capability to deal with the complexity of real-life systems as well as its usability, as shown by the case studies reported in Section 5.

[1] See https://eclipse.org/escet.

[2] 'Eclipse', 'Eclipse ESCET' and 'ESCET' are trademarks of Eclipse Foundation, Inc.

[3] See https://eclipse.org/escet/cif.

[4] See https://eclipse.org/escet/chi and https://eclipse.org/escet/tooldef.

2 Supervisory Controller Synthesis

Figure 1 depicts the general system structure for supervisory control. A cyber-physical system consists of mechanical components to be controlled. Actuators drive their operation, while sensors indicate their status. Resource control provides low-level control, often offering more abstract actuator and sensor signals for higher levels of control to use. Supervisors ensure actuator signals at lower layers (the *plant*) that would violate requirements are disabled. Large systems may be divided into (layers of) subsystems, and supervisors can be present at each level, coordinating lower-level subsystems (only a single layer is depicted). A (sub)system is often controlled by a human operator through a graphical user interface, or part of a larger system to which it is connected by an interface.

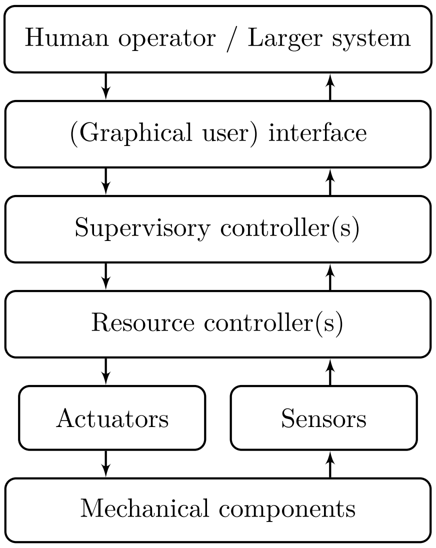
Human operator / Larger system
(Graphical user) interface
Supervisory controller(s)
Resource controller(s)
Actuators
Sensors
Mechanical components

Fig. 1. Structure of supervisory control.

Supervisory controller synthesis [21,33] automatically generates a correct-by-construction supervisor model for a discrete-event system, given precise descriptions of the behavior of the plant components as well as the (safety) requirements for the overall plant behavior. These can be specified conveniently as extended finite automata (EFAs), i.e., automata with variables, guards and updates, possibly carrying invariants that restrict the state space [13].

Synthesis considers the synchronous product of the plant automata together with the requirement automata. That is, these automata synchronize on shared events, meaning these events must be executed simultaneously. If an event is missing in the local state of any plant automaton, or is restricted by a plant invariant, it is absent from the overall system state, and it is considered physically impossible. If, on the other hand, an event is missing only in the states of requirement automata, or is restricted by a requirement invariant, it is physically possible but must be disabled by the synthesized supervisor to ensure *safety*.

Controllable events (such as output signals to actuators) can be prevented by a supervisor, but uncontrollable events (such as input signals from sensors) cannot. To ensure *controllability*, if an uncontrollable event must be prevented, the supervisor makes the system state where it occurs unreachable by disabling all controllable events leading to it. Moreover, if an uncontrollable event leads to such a state, the origin state of this event must be made unreachable too.

If safety of, for instance, a drawbridge is ensured by forcing it to remain raised forever, it is useless for road traffic. Therefore states of the plant and requirement EFAs can be marked, for instance states where the bridge deck is lowered, the barriers are open, and the signals are green. A marked state in the synchronous product means all individual plant components are in a marked local state, in this case allowing traffic to proceed over the bridge. The supervisor must guarantee that the plant can always reach a marked state, by disabling (events leading to) states that violate this property. Such a supervisor is said to be *nonblocking*.

Supervisory controller synthesis ensures *safety*, *controllability* and *nonblockingness* of a system with respect to its requirements, accounting for all possible behavior, also disabling events that lead to problems such as blocking behavior or requirement violations much later in the system's execution. It does so by restricting as little behavior as possible, thus ensuring *maximal permissiveness*.

Next to ESCET toolkit, other supervisory controller synthesis tools include DESTool [16], DESUMA [25], Supremica [12], and TCT [6]. For a comparison between these tools see [24]. The ESCET toolkit can be used to specify various different models during the entire development process, including simulation models, as it has a rich set of concepts. This prevents having to use multiple languages. It has a strong focus on industrial application, with, e.g., modeling convenience, efficient algorithms, and checking for common mistakes.

3 Synthesis-based Engineering Process

Figure 2 shows ESCET's synthesis-based engineering process. It starts with a model-based specification, consisting of plant and requirement models, modeled as EFAs and/or invariants. To these models, supervisory controller synthesis is applied, resulting in a model of the supervisor. The ESCET toolkit supports synthesis both with its own synthesis tools, and by a transformation to Supremica.

Synthesis ensures that all specified requirements are satisfied by the synthesized supervisor. Verification, such as model checking, supported through transformations to UPPAAL [2] and mCRL2 [3], can be used to check other requirements not yet supported by synthesis, including liveness guarantees or timing requirements. Validation, supported by ESCET's automated or interactive simulation and visualization, helps to determine whether the specified requirements, and thus the supervisor, achieve the desired system behavior.

An implementation of the controller can be obtained automatically from a model of the supervisor, by generating code for its control software. The ESCET toolkit supports code generation for multiple languages and platforms, including Java, C, Simulink, and PLC code (IEC standard 61131-3) for multiple vendors.

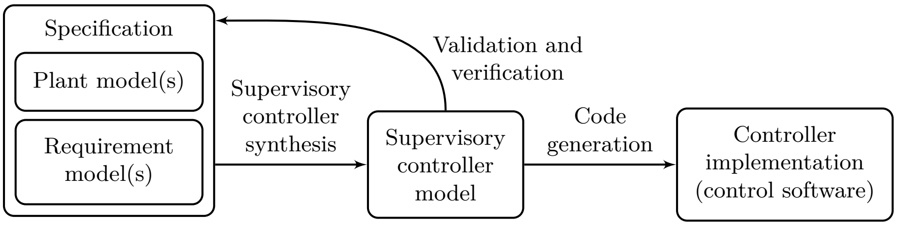
Specification
Plant model(s)
Requirement model(s)
Supervisory controller synthesis
Validation and verification
Supervisory controller model
Code generation
Controller implementation (control software)

Fig. 2. Simplified representation of ESCET's synthesis-based engineering process.

4 Technical Improvements

We describe recent improvements and novel techniques that have been vital in making supervisory controller synthesis applicable to industrial-size cyber-physical systems. Some have already been integrated into the ESCET toolkit, while others are being integrated or are planned to be integrated.

Symbolic synthesis The ESCET toolkit is based on the symbolic supervisory controller synthesis algorithm from Ouedraogo et al. [19]. It iteratively strengthens guard predicates on transitions so that forbidden states become unreachable in the controlled plant. This represents a major step forward for the industrial applicability of supervisory controller synthesis, by allowing for synthesis of plants and requirements intuitively modeled as EFAs.

The use of EFAs also opens up the possibility to extract and represent the synthesized supervisor more compactly and intuitively [15]. The ESCET toolkit represents the supervisor model as the collection of the provided plant and requirement models together with the addition of a single EFA containing a strengthened guard for each controllable event.

BDD Data Structure The Binary Decision Diagram (BDD) data structure allows to efficiently and symbolically represent and manipulate predicates representing (parts of) state spaces [14]. Its use in ESCET's symbolic supervisory controller synthesis algorithm leads to major reductions of state space representations and computation times, which is essential for scalability.

Vital to the memory and running time characteristics of Reduced Ordered BDD representations and manipulations, as used by the ESCET toolkit, is the ordering of the Boolean variables [30]. Heuristic variable ordering algorithms that exploit the inherent structure of the system modeled as EFAs are able to significantly reduce the synthesis effort [11], especially for larger inputs, making synthesis applicable to more complex systems.

Multilevel Synthesis Contrary to monolithic synthesis, where only a single supervisor is synthesized, with multilevel synthesis [10] the plant components and requirements are grouped together into a hierarchical structure, and a separate supervisor is synthesized for each group. This allows to distribute the control problem over multiple cooperating supervisors, which together are significantly smaller than one monolithic supervisor. By encoding relations between plant components and requirements in a design structure matrix [5], and algorithmically reordering its rows and columns to place tightly coupled plant components side by side [32], a suitable multilevel structure can be obtained. Compared to monolithic synthesis, this can for certain systems substantially reduce synthesis effort [8], enabling synthesis for much larger variants of such systems.

Avoiding Nonblockingness Checks Although the local supervisors in multilevel synthesis are nonblocking, the overall supervisor may not be. A global nonblockingness check can be used to guarantee that all local supervisors can reach a marked state at the same moment in time, but is often expensive, nullifying much of the gains obtained through applying multilevel synthesis. However, in a dependency graph that encodes which plant components by means of requirements depend on state of other plant components to perform certain events, plant components do not give rise to blocking behavior if they are not part of an infinite path [9]. For certain systems, using such graphs, the global nonblockingness checks may be skipped entirely, or may be reduced to consider less subsystems.

Symmetry Reduction Real-life systems tend to contain a significant number of similar components, that for instance only differ by the instantiation of some of the parameters or their physical locations within the overall system. Such symmetries can be exploited to reduce the number of plant and requirement automata needed in the synthesis process, further reducing the synthesis effort [18].

5 Case Studies and Applications

Rijkswaterstaat Initially the collaboration with Rijkswaterstaat focused on generating control software with supervisory controller synthesis for bridges, waterway locks, and storm surge barriers. Notable case studies are the Algera complex, comprising a bascule bridge, a waterway lock and two storm surge barriers in the river Hollandse IJssel [22], and the Oisterwijksebaanbrug, a rotating bridge in Tilburg [23]. For the latter, a fault-tolerant controller was synthesized, from which PLC code was generated, which passed the regular site acceptance test.

Recent case studies target road tunnels, notably the Eerste Heinenoord tunnel [18] and the Swalmen tunnel [17], and roadside systems [31]. For the Swalmen tunnel, a digital twin, a 3D digital copy of a physical system, was conveniently constructed from the plant and requirement models. Combined with visualization, this allows simulation of the system in a setting close to real life.

ASML A prominent result of the collaboration with ASML is the use of the ESCET toolkit in a toolkit from another Eclipse Foundation open-source project, the Eclipse Logistic Specification and Analysis Toolkit (LSAT$^{\text{TM}}$) [26]. The LSAT toolkit is used at ASML to create fully calibrated models of subsystems of a wafer scanner, responsible for transporting wafers in and out of the scanner and performing preprocessing steps before the wafer is being exposed on the wafer stage subsystem. The LSAT toolkit exploits ESCET's supervisory controller synthesis to compute valid orderings of logistics activities, while maintaining the maximum freedom to subsequently perform scheduling on the synthesis result to compute a supervisor that optimizes productivity [20].

6 Conclusions

The ESCET project and toolkit support synthesis-based engineering to efficiently generate high-quality correct-by-construction supervisors. The toolkit is being applied to complex industrial systems in different domains. The project's open environment enables effective collaboration between industry, researchers and tool vendors. Owing to positive experiences with the ESCET toolkit, Rijkswaterstaat is seriously considering whether its document-based development process for control software could be adapted to one based on SBE with the ESCET toolkit.

7 Data-Availability Statement

The artifact that supports this paper is available at Zenodo under identifier doi:10.5281/zenodo.7296616 [4]. It contains Eclipse ESCET v0.7 for Linux. However, the authors prefer that the Eclipse ESCET toolkit is downloaded directly from the Eclipse Foundation, where the latest version of the toolkit is available for multiple platforms.[5]

References

1. van Beek, D.A., Fokkink, W.J., Hendriks, D., Hofkamp, A.T., Markovski, J., van de Mortel-Fronczak, J.M., Reniers, M.A.: CIF 3: Model-based engineering of supervisory controllers. In: Proc. 20th Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 8413, pp. 575–580. Springer (2014). https://doi.org/10.1007/978-3-642-54862-8_48
2. Behrmann, G., David, A., Larsen, K.G., Håkansson, J., Pettersson, P., Yi, W., Hendriks, M.: UPPAAL 4.0. In: Proc. 3rd Conference on the Quantitative Evaluation of Systems (QEST). pp. 125–126. IEEE (2006). https://doi.org/10.1109/QEST.2006.59
3. Bunte, O., Groote, J.F., Keiren, J.J.A., Laveaux, M., Neele, T., de Vink, E.P., Wesselink, W., Wijs, A., Willemse, T.A.C.: The mCRL2 toolset for analysing concurrent systems - Improvements in expressivity and usability. In: Proc. 25th Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 11428, pp. 21–39. Springer (2019). https://doi.org/10.1007/978-3-030-17465-1_2
4. Eclipse Foundation: Eclipse ESCET v0.7 for Linux (2022). https://doi.org/10.5281/zenodo.7296616
5. Eppinger, S.D., Browning, T.R.: Design Structure Matrix Methods and Applications. MIT Press (2012)
6. Feng, L., Wonham, W.M.: TCT: A computation tool for supervisory control synthesis. In: Proc. 8th Workshop on Discrete Event Systems (WODES). pp. 388–389. IEEE (2006). https://doi.org/10.1109/WODES.2006.382399
7. Fokkink, W.J., Goorden, M.A., van de Mortel-Fronczak, J.M., Reijnen, F.F.H., Rooda, J.E.: Supervisor synthesis: Bridging theory and practice. Computer **55**(10), 48–54 (2022). https://doi.org/10.1109/MC.2021.3134934
8. Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Fokkink, W.J., Rooda, J.E.: Structuring multilevel discrete-event systems with dependence structure matrices. IEEE Transactions on Automatic Control **65**(4), 1625–1639 (2020). https://doi.org/10.1109/TAC.2019.2928119
9. Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Fabian, M., Fokkink, W.J., Rooda, J.E.: Model properties for efficient synthesis of nonblocking modular supervisors. Control Engineering Practice **112**, 104830 (2021). https://doi.org/10.1016/j.conengprac.2021.104830
10. Komenda, J., Masopust, T., van Schuppen, J.H.: Control of an engineering-structured multilevel discrete-event system. In: Proc. 13th Workshop on Discrete Event Systems (WODES). pp. 103–108. IEEE (2016). https://doi.org/10.1109/WODES.2016.7497833

[5] See https://eclipse.org/escet/download.html.

11. Lousberg, S., Thuijsman, S.B., Reniers, M.A.: DSM-based variable ordering heuristic for reduced computational effort of symbolic supervisor synthesis. IFAC-PapersOnLine **53**(4), 429–436 (2020). https://doi.org/10.1016/j.ifacol.2021.04.058
12. Malik, R., Åkesson, K., Flordal, H., Fabian, M.: Supremica–An efficient tool for large-scale discrete event systems. IFAC-PapersOnLine **50**(1), 5794–5799 (2017). https://doi.org/10.1016/j.ifacol.2017.08.427
13. Markovski, J., van Beek, D., Theunissen, R., Jacobs, K., Rooda, J.: A state-based framework for supervisory control synthesis and verification. In: Proc. 49th IEEE Conference on Decision and Control (CDC). pp. 3481–3486 (2010). https://doi.org/10.1109/CDC.2010.5717095
14. McMillan, K.L.: Symbolic Model Checking. Springer (1993). https://doi.org/10.1007/978-1-4615-3190-6
15. Miremadi, S., Åkesson, K., Lennartson, B.: Extraction and representation of a supervisor using guards in extended finite automata. In: Proc. 9th Workshop on Discrete Event Systems (WODES). pp. 193–199. IEEE (2008). https://doi.org/10.1109/WODES.2008.4605944
16. Moor, T., Schmidt, K., Perk, S.: libFAUDES — An open source C++ library for discrete event systems. In: Proc. 9th Workshop on Discrete Event Systems (WODES). pp. 125–130. IEEE (2008). https://doi.org/10.1109/WODES.2008.4605933
17. Moormann, L., van Hegelsom, J., Maessen, P., van de Mortel-Fronczak, J.M., Fokkink, W.J., Rooda, J.E.: Advantages of using digital twins in the validation of road tunnel supervisory controllers. In: Proc. ITA/AITES World Tunnel Congress (WTC). pp. 573–578 (2022)
18. Moormann, L., van de Mortel-Fronczak, J.M., Fokkink, W.J., Maessen, P., Rooda, J.E.: Supervisory control synthesis for large-scale systems with isomorphisms. Control Engineering Practice **115**, 104902 (2021). https://doi.org/10.1016/j.conengprac.2021.104902
19. Ouedraogo, L., Kumar, R., Malik, R., Åkesson, K.: Nonblocking and safe control of discrete-event systems modeled as extended finite automata. IEEE Transactions on Automation Science and Engineering **8**(3), 560–569 (2011). https://doi.org/10.1109/TASE.2011.2124457
20. van Putten, B.J.C., van der Sanden, L.J., Reniers, M.A., Voeten, J.P.M., Schiffelers, R.R.H.: Supervisor synthesis and throughput optimization of partially-controllable manufacturing systems. Discrete Event Dynamic Systems **31**, 103–135 (2021). https://doi.org/10.1007/s10626-020-00325-x
21. Ramadge, P.J., Wonham, W.M.: Supervisory control of a class of discrete event processes. SIAM Journal on Control and Optimization **25**(1), 206–230 (1987). https://doi.org/10.1137/0325013
22. Reijnen, F.F.H., Goorden, M.A., van de Mortel-Fronczak, J.M., Rooda, J.E.: Modeling for supervisor synthesis - a lock-bridge combination case study. Discret. Event Dyn. Syst. **30**(3), 499–532 (2020). https://doi.org/10.1007/s10626-020-00314-0
23. Reijnen, F.F.H., Leliveld, E.B., van de Mortel-Fronczak, J.M., van Dinther, J., Rooda, J.E., Fokkink, W.J.: Synthesized fault-tolerant supervisory controllers, with an application to a rotating bridge. Computers in Industry **130**, 103473 (2021). https://doi.org/10.1016/j.compind.2021.103473
24. Reniers, M.A., van de Mortel-Fronczak, J.M.: An engineering perspective on model-based design of supervisors. IFAC-PapersOnLine **51**(7), 257–264 (2018). https://doi.org/10.1016/j.ifacol.2018.06.310
25. Ricker, L., Lafortune, S., Genc, S.: DESUMA: A tool integrating GIDDES and UMDES. In: Proc. 8th Workshop on Discrete Event Systems (WODES). pp. 392–393. IEEE (2006). https://doi.org/10.1109/WODES.2006.382402

26. van der Sanden, L.J., Blankenstein, Y., Schiffelers, R.R.H., Voeten, J.P.M.: LSAT: Specification and analysis of product logistics in flexible manufacturing systems. In: Proc. 17th Conference on Automation Science and Engineering (CASE). pp. 1–8. IEEE (2021). https://doi.org/10.1109/CASE49439.2021.9551412
27. van der Sanden, L.J., Reniers, M.A., Geilen, M.C.W., Basten, T., Jacobs, J., Voeten, J.P.M., Schiffelers, R.R.H.: Modular model-based supervisory controller design for wafer logistics in lithography machines. In: Proc. 18th Conference on Model Driven Engineering Languages and Systems (MODELS). pp. 416–425. IEEE (2015). https://doi.org/10.1109/MODELS.2015.7338273
28. Schiffelers, R.R.H., van Beek, D.A., Man, K.L., Reniers, M.A., Rooda, J.E.: A hybrid language for modeling, simulation and verification. IFAC Proceedings Volumes **36**(6), 199–204 (2003). https://doi.org/10.1016/S1474-6670(17)36431-5
29. Swartjes, L., van Beek, D.A., Fokkink, W.J., van Eekelen, J.A.W.M.: Model-based design of supervisory controllers for baggage handling systems. Simul. Model. Pract. Theory **78**, 28–50 (2017). https://doi.org/10.1016/j.simpat.2017.08.005
30. Thuijsman, S.B., Hendriks, D., Theunissen, R., Reniers, M.A., Schiffelers, R.R.H.: Computational effort of bdd-based supervisor synthesis of extended finite automata. In: Proc. 15th International Conference on Automation Science and Engineering (CASE). pp. 486–493 (2019). https://doi.org/10.1109/COASE.2019.8843327
31. Verbakel, J.J., Vos de Wael, M.E.W., van de Mortel-Fronczak, J.M., Fokkink, W.J., Rooda, J.E.: A configurator for supervisory controllers of roadside systems. In: Proc. 17th Conference on Automation Science and Engineering (CASE). pp. 784–791. IEEE (2021). https://doi.org/10.1109/CASE49439.2021.9551485
32. Wilschut, T., Etman, L.F.P., Rooda, J.E., Adan, I.J.B.F.: Multilevel flow-based Markov clustering for design structure matrices. Journal of Mechanical Design **139**(12) (2017). https://doi.org/10.1115/1.4037626
33. Wonham, W.M., Cai, K., Rudie, K.: Supervisory control of discrete-event systems: A brief history. Annual Reviews in Control **45**, 250–256 (2018). https://doi.org/10.1016/j.arcontrol.2018.03.002

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.