21st INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2019
MONTEREY, CA, USA, 23 – 25 September, 2019

DSM-based Analysis for the Recognition of Modeling Errors in Supervisory Controller Design

Martijn Goorden, Joanna van de Mortel-Fronczak, Pascal Etman, Jacobus Rooda

Eindhoven University of Technology

Abstract: The design of supervisory controllers for cyber-physical systems is steadily becoming harder as increasingly more functionality needs to be automated, the systems become larger, and safe operation becomes more important. Model-based systems engineering incorporating formal methods such as supervisory control synthesis can be used to synthesize these supervisory controllers based on models of the uncontrolled system components and models of the control requirements. Although synthesis is an automatic procedure, creating these models is still a manual activity prone to modeling errors. In this paper, we propose to use several DSM-supported analysis techniques to identify potential modeling errors. Analyzing the dependencies between uncontrolled system component models and requirement models with both a domain mapping matrix and a dependency structure matrix reveals potential modeling errors. We present several examples of models from literature to show the potential effectiveness of the DSM-supported analysis of the uncontrolled system and the associated control requirements.

Keywords: DSM, MBSE, supervisory control theory, formal methods

1 Introduction

Currently, the fourth industrial revolution is taking place, called Industry 4.0, where the physical world and the digital world are intertwined resulting in cyber-physical systems, see (Lasi et al., 2017). In these cyber-physical systems, more and more functionality is automated. This results in the ever increasing responsibility of the supervisory control systems for a proper and safe execution of these automated functions.

Model-based systems engineering (MBSE) is often proposed as a design method used to increase the quality of the system, decrease the development cost, and decrease time-to-market, see (Bahill and Botta, 2008) and (Ramos et al., 2012). Combining MBSE with a mathematical formalism opens up the possibility to even further improve the design as desired properties can be analyzed by algorithms, see for example (Waymore, 1993).

Supervisory Control Synthesis (SCS) of (Ramadge and Wonham, 1987, 1989) provides means to automatically derive (i.e., synthesize) supervisory controllers based on a model of the system (in control theory called the plant) and a model of the requirements. These supervisory controllers are proven to ensure that the behavior of the plant always satisfies the imposed requirements, i.e., the supervisory controllers are correct-by-construction. In (Baeten et al., 2016), SCS is integrated into MBSE to benefit from synthesis in the development of supervisory controllers for cyber-physical systems.

For SCS, the synthesized supervisory controllers are correct for the provided plant model and requirement model. However, if the quality of these models is insufficient, the guarantee of a correct supervisory controller is meaningless when it is implemented on the actual system, which undermines all the benefits of MBSE. As modeling is still a human activity, the quality of these models is susceptible to modeling errors. As systems to be automated become larger and larger, identifying modeling errors becomes cumbersome.

The contribution of this paper is showing how Dependency Structure Matrix (DSM) based techniques, introduced in (Steward, 1981) and reviewed in (Eppinger and Browning, 2012), can be utilized to analyze the plant and requirement models in order to recognize potential modeling errors. As SCS deploys a mathematical formalism, the interactions (dependencies) between plant and requirement models can be automatically recorded in a Domain Mapping Matrix (DMM). From this DMM, a DSM can be constructed. Analysis of both the DMM and the DSM can reveal modeling errors previously unseen by the engineer.

This paper is structured as follows. Section 2 introduces concisely the basic concepts of SCS to be able to interpret the DMM and the DSM derived from the plant and requirement models. Section 3 describes the derivation of the DMM and shows which modeling errors can be identified from the DMM. Subsequently, in Section 4, it is shown how the analysis of the DSM constructed from the DMM contributes to finding modeling errors. Examples of modeling errors of large cyber-physical systems are presented in Sections 3 and 4. The paper concludes with Section 5.

2 Design of Supervisory Control Systems

Supervisory Control Synthesis (SCS) as initiated by (Ramadge and Wonham, 1987, 1989) provides the means to automatically derive (i.e., synthesize) a model of a supervisory controller based on the formal models of the plant and the requirements. The models of the plant describe all possible behavior of the system, i.e., what the system *can* do. The models of the requirements formulate the desired behavior of the system, i.e., what the system *should* do.

Automata are one of the modeling formalisms utilized by SCS. An automaton describes the possible states of a system (e.g., a lamp can have the states `On` and `Off`) and events that change the state of a system (e.g., event `go_on` turns the lamp on and event `go_off` turns it off). Typically, for each component in the system, such as actuators, sensors, and buttons, an automaton is constructed to act as the plant model. Automata can be also used to formulate the requirements, one for each requirement. For example, a requirement may express that a lamp may only go on after the operator pushed a specific button.

The synthesized supervisory controller can be used to control the uncontrolled plant. Based on events observed in the plant, it may disable events such that they cannot be performed next. For example, the supervisor may disable the event `go_on` as long as it has not observed the pushed event of the button. The method guarantees that the system consisting of the uncontrolled plant and the synthesized supervisor together adhere to the (modelled) requirements.

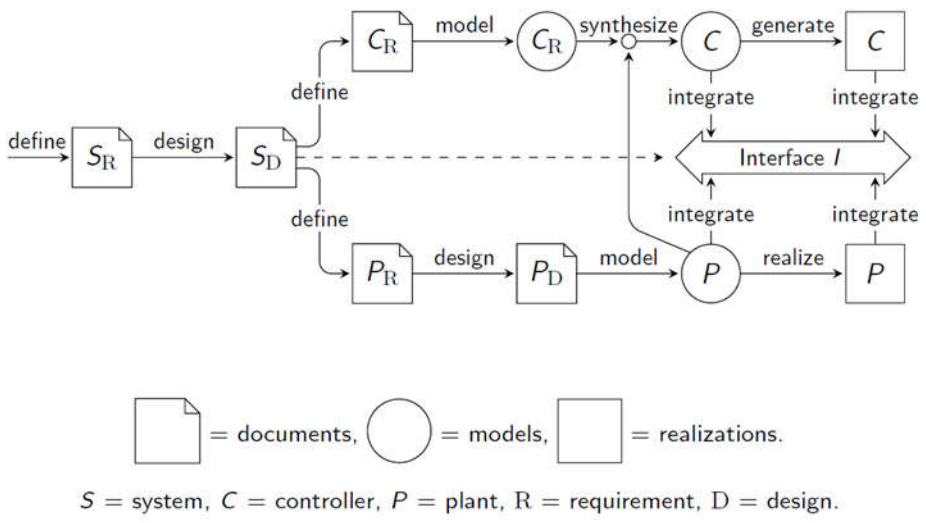


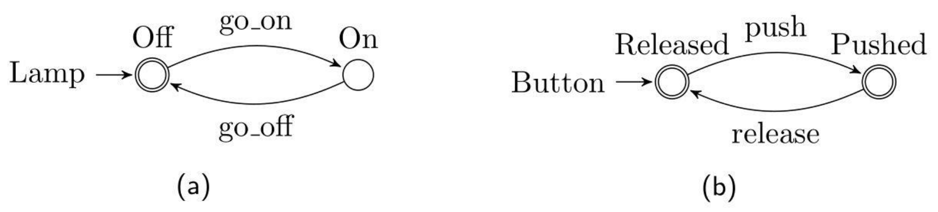
Figure 1. SCS in combination with MBSE (Baeten et al., 2016).

Figure 1 shows the integration of SCS in MBSE, from (Baeten et al., 2016). In general, the systems engineering process starts with system requirements followed by a system design partitioned into subsystems or modules. For each module, requirements are defined based on the system design. Figure 1 shows an example where all modules of the plant are displayed as one module (bottom row) and the supervisory controller as the second module (top row). For the plant, a design is defined which is translated into a model. For the purpose of supervisory control synthesis, the plant is modeled with automata as described above. For the supervisory controller, no design is defined, but the requirements are formulated directly as a model. From the model of the plant and the model of the requirements, a supervisory controller can be synthesized with SCS. The model of the supervisory controller together with the plant model can be used, for example, for simulation-based validation. Finally, the actual plant can be realized from the model and the actual supervisory control code can be generated from the model of the supervisory controller.

In practice, instead of creating a single large automaton as the model of the plant, the plant is modeled with a set of smaller automata, which are called the plant models. Similarly, the model of the requirement consists of a set of small requirement models. In the rest of the paper, we use this notion of a set of plant models and a set of requirement models, so a model will be an element of one of these sets.

3 Domain Mapping Matrix Analysis

To obtain a supervisory controller with SCS, two kinds of models are needed: plant models and requirement models. The roles of these two kinds of models are different in the synthesis algorithms. Therefore, a Domain Mapping Matrix (DMM) suits the analysis of the models where the plant models are the elements on the one axis and the requirement models are the elements on the other axis.



requirement Lamp.go_on **needs** Button.Pushed

Figure 2. Examples of two plant models, with (a) a model of a lamp and (b) a model of a button, and a requirement expressing that the lamp may only go on when the button is pushed.

As automata are used as a modeling formalism in SCS, we define a dependency between a plant model and a requirement as follows. There is a dependency between plant model P_i and requirement model R_j if and only if requirement model R_j uses a state or an event from plant model P_i, as formalized in (Goorden et al., 2017). For example, consider again the simple system of a lamp and a button, and the requirement stating that the lamp may only go on when the button is pushed, shown in Figure 2. There is a dependency between this requirement and the lamp, as the requirement uses the event `go_on` from the lamp. Furthermore, there is also a dependency between this requirement and the button, as the requirement uses the state Pushed of the button. The benefit of using a mathematical formalism is that these dependencies can automatically be identified from the model.

With this definition of a dependency between plant models and requirement models, a DMM PR can be constructed. Since we construct this DMM automatically, any error observed in the DMM can be related to an error in the provided models, not in the method of constructing the DMM, as could be the case when it was constructed manually. In this paper, we place the plant models along the rows and the requirements along its columns. A binary DMM (i.e., the entries of the DMM are either 0 or 1) is sufficient for the purpose of identifying modeling errors, as we show next.

Figure 3 shows the DMM of the first two workstations of a real production line (which consists of 6 subsequent workstations in total). A model of this production line has been provided in (Reijnen et al., 2018). An earlier (incomplete and incorrect) version of the model, not the final version published in (Reijnen et al, 2018), is used here to show how modeling errors can be identified. The modeling errors in this and subsequent examples are errors encountered during the development process, not errors artificially injected in the model to demonstrate the presented method. For readability of the large DMM, the names of the plant and requirement models are replaced by numbers. An entry in row i and column j of the DMM indicates a dependency between plant model P_i and requirement model R_j. For example, entry $(10, 1)$ indicates that requirement R_1 mentions a state or event from plant P_{10}.

The following potential modeling errors can be identified from the DMM. First, an empty row indicates that no requirement mentions a state or an event from that particular plant model. This means that the behavior of this subsystem is not restricted by any requirement or the behavior of this subsystem does not influence the behavior of another subsystem.

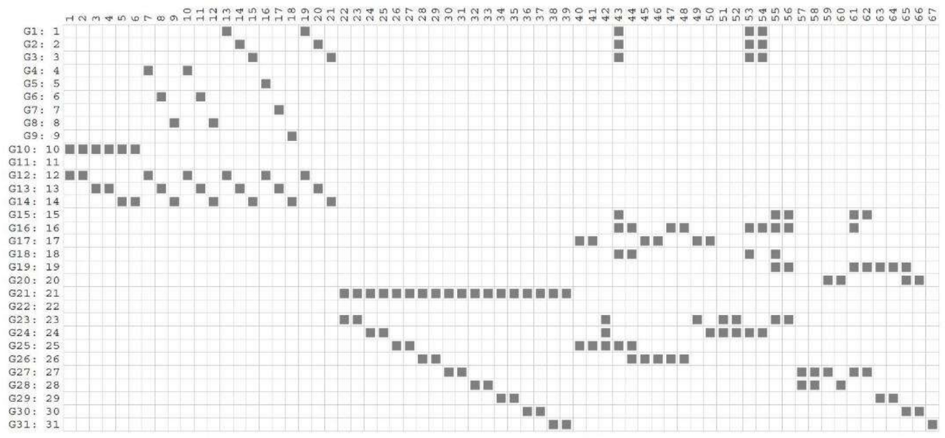


Figure 3. The DMM of the first two workstations of a production line.

Probably, this is not the intention of the modeler. Either requirements are missing or the subsystem is obsolete and should not be modeled. Consider the example shown in Figure 3. Row 11 is an empty row and therefore indicates a potential modeling error. After analysis, it turned out that the modeled sensor of the production line is indeed obsolete for the intended functioning and the plant model P_{11} was removed from the system model.

Second, an empty column also indicates a modeling error. Some modeling tools for SCS allow for the modeling of requirements that refer to other requirements, while this was never the intention of SCS. A DMM could help in identifying these modeling errors. As in the DMM the dependencies between plant models and requirement models are captured, and not the dependencies between requirement models themselves, such a modeling error would result in an empty column.

Third, a column with just a single nonzero entry may also indicate a modeling error, but that should be confirmed by the modeler. It may be the case that a requirement is only restricting the internal behavior of a component and not its interaction; only in this case a single nonzero entry is expected. If the modeler cannot confirm this, a single nonzero entry probably indicates a missing dependency or an obsolete dependency. In Figure 3, the last column of requirement R_{67} has a single nonzero entry. It turned out that this was indeed a modeling error, as this requirement was stating that component P_{31} should not be used. Another way of expressing this is simply by removing this component from the model, which happened in the revised model of the production line as published in (Reijnen et al., 2018).

4 Dependency Structure Matrix analysis

From DMM PR, a square Dependency Structure Matrix P^2 can be constructed by the matrix transformation $P^2 = PR \cdot PR^T$, where PR^T is the transpose of matrix PR (Maurer, 2007). In DSM P^2, the plant models are the elements along its axes and a dependency between two plant models indicates that there exists a requirement that mentions a state or an event from both plant models. While this DSM construction would create potential dependencies (which should be checked to verify whether each potential dependency is an actual dependency), within the context of SCS all potential dependencies are actual dependencies. Consider the requirement from Section 2 that expresses that a lamp may only go on after the operator pushed a specific button. SCS can only synthesize a supervisor for this requirement if both the model of the lamp and the model of the button are provided as input. Therefore, no matter how the system is clustered (or portioned), at some point both models need to be together. This dependency between the plant models is exactly obtained by the matrix transformation described by (Maurer, 2007).

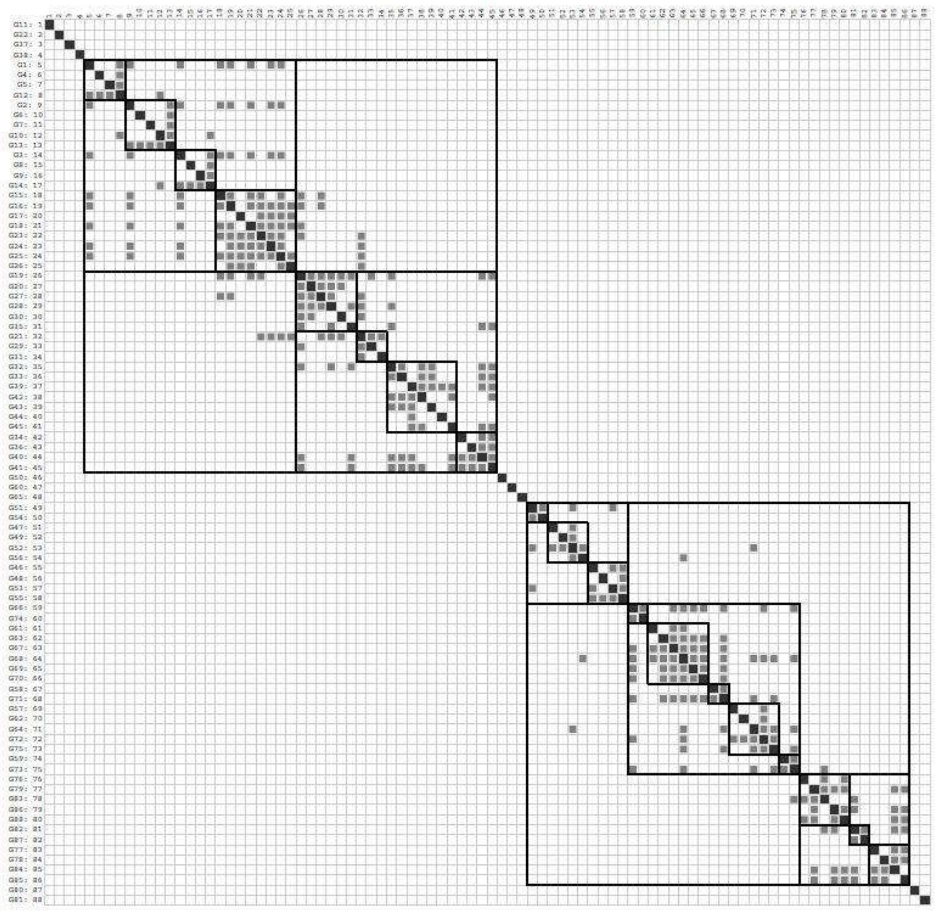


Figure 4. The DSM of the complete production line, based on an earlier version of the model.

When the DSM P^2 is subsequently clustered, another opportunity to identify modeling errors is obtained. The DSMs in this paper have been clustered with the Markov-based clustering algorithm of (Wilschut et al., 2017). Other clustering algorithms may also be used, like k-means clustering (Hartigan and Wong, 1979), spectral clustering (Sarkar et al., 2014), and hierarchical clustering (AlGeddawy and ElMaraghy, 2013).

First, clustering P^2 may reveal disjoint subsystems. An example is shown in Figure 4, where the clustered DSM of the full production line of (Reijnen et al., 2018) is shown. The identified plant model with an empty row in the DMM PR of Figure 3 shows up in the DSM as also having an empty row (and column by construction). More interesting, there are two large disjoint subsystems present in the model of the system. If the modeler can argue that they are indeed two independent subsystems, there is no need from the perspective of SCS to combine them into a single system model. That is, the synthesis algorithms of SCS can be applied on each subsystem independently to derive two separate supervisory controllers. More probably, the modeler missed requirements that combine the two subsystems together resulting in a single system. The latter was the case in the development of the models for the production line. Using the DSM in Figure 4, the missing requirements describing the desired interaction between third and fourth workstation have been identified easily.

Second, the clustered DSM P^2 can also be analyzed more deeply by inspecting how plant models are actually clustered. Often, large cyber-physical systems contain multiple similar components performing similar functions. Therefore, one expects to see similar clusters for these components. Consider the DSM shown in Figure 5. This DSM depicts the dependencies within a model for supervisory control of a waterway lock, as described in (Reijnen et al., 2017). Despite the size of the system, analyzing the DSM P^2 helped in finding modeling errors. In Figure 5, the buttons of the user interface related to stopping the system are shown. The operator has five buttons: an emergency button and four buttons stopping parts of the system. In the model, G_{21} is the emergency button, and G_{22} through G_{25} are the normal stops. The clustering result shows that all stop buttons are clustered together, but one hierarchical level lower stop button G_{24} is clustered with the emergency stop and not with the other stop buttons, which is counter intuitive based on system knowledge. Further inspection of the DSM shows that G_{24} has fewer dependencies than all other stop buttons, while all of them have similar functions. After inspecting the actual model, it turned out that quite some requirements were referring to the stop button G_{25} while they should have referred to G_{24}. Such typing errors are not identified by the modeling formalism, as G_{25} is also a model in the system, but by analyzing the DSM this modeling error is recognized.

5 Conclusion

The success of MBSE in the design of supervisory controllers for cyber-physical systems depends on the quality of the provided models. In this paper, we propose to use DSM-based analysis of these models to reveal potential modeling errors. By analyzing the dependencies between plant models and requirement models, different kinds of errors may be identified, like missing requirements, obsolete plant models, and wrongly formulated requirements. Both the DMM, with plant models along one axis and the requirement models along the other axis, and the DSM constructed from the DMM, with plant models along both of its axes, are useful in analyzing the large system model. Creating the DMM and the DSM during the modeling process allows the control engineer to reflect on the current models and eventually conclude with confidence the correctness of the final model.

Future work includes clustering of the DMM and investigating the DSM $R^2 = PR^T \cdot PR$ with requirement models along both of its axis (instead of plant models) to see whether the clustered DMM and this DSM also contain features that may indicate modeling errors.

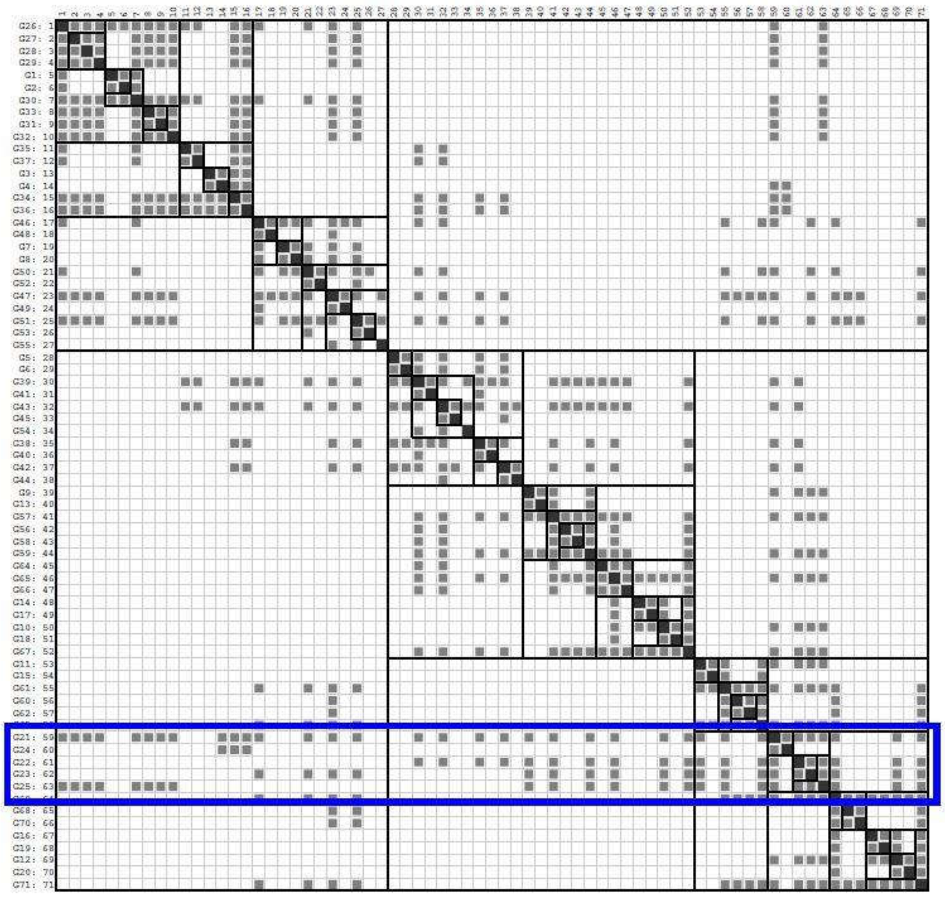


Figure 5. The DSM of the model of Lock III where the part indicated within the blue rectangle shows the stop buttons in the user interface.

Acknowledgement

The authors thank Ferdie Reijnen for providing the different versions of the models including the modeling errors created before the published version. The authors thank Rijkswaterstaat, part of the Dutch Ministry of Infrastructure and Water Management, for providing funding for this research. In particular, the authors thank Maria Angenent, Bert van der Vegt, and Han Vogel for their feedback on the results.

References

AlGeddawy, T., ElMaraghy, H., 2013. Optimum granularity level of modular product design architectures. CIRP Annals 62, 151-154.

Baeten, J.C.M., van de Mortel-Fronczak, J.M., Rooda, J.E., 2016. Integration of supervisory control synthesis in model-based systems engineering. Complex Systems 55, 39-58.

Bahill, A.T., Botta, R., 2008. Fundamental principles of good system design. Engineering Management Journal 20, 9-17.

Eppinger, S.D., Browning, T.R., 2012. Design structure matrix methods and applications. MIT press.

Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Rooda, J.E., 2017. Structuring multilevel discrete-event systems with dependency structure matrices. IEEE Annual Conference on Decision and Control, 558-564.

Hartigan, J.A., Wong, M.A., 1979. Algorithm AS 136: a k-means clustering algorithm. Journal of the Royal Statistical Society, Series C (Applied Statistics) 28, 100-108.

Lasi, H., Fettke, P., Kemper, H.G., Feld, T., Hoffmann, M., 2017. Industry 4.0. Business & Information Systems Engineering 6, 239-242.

Maurer, M.S., 2007. Structural awareness in complex product design. PhD thesis, Universität München.

Ramadge, P.J.G., Wonham, W.M., 1987. Supervisory control of a class of discrete event processes. SIAM Journal on Control and Optimization 25, 206-230.

Ramadge, P.J.G., Wonham, W.M., 1989. The control of discrete event systems. Proceedings of the IEEE 77, 81-98.

Ramos, A.L., Ferreira, J.V., Barceló, J., 2012. Model-based systems engineering: an emerging approach for modern systems. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 42, 100-111.

Reijnen, F.F.H., Goorden, M.A., van de Mortel-Fronczak, J.M., Rooda, J.E., 2017. Supervisory control synthesis for a waterway lock. IEEE Conference on Control Technology and Applications, 1562-1568.

Reijnen, F.F.H., Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Rooda, J.E., 2018. Application of dependency structure matrices and multilevel synthesis to a production line. IEEE Conference on Control Technology and Applications, 458-464.

Sarkar, S., Dong, A., Henderson, J.A., Robinson, P.A., 2014. Spectral characterization of hierarchical modularity in product architectures. ASME Journal of Mechanical Design 136, 240-252.

Steward, D.V., 1981. The design structure system: a method for managing the design of complex systems. IEEE Transactions on Engineering Management EM-28, 71-74.

Waymore, A.W., 1993. Model-based systems engineering. CRC Press.

Wilschut, T., Etman, L.F.P., Rooda, J.E., Adan, I.J.B.F., 2017. Multilevel flow-based Markov clustering for design structure matrices. ASME Journal of Mechanical Design 139, 121402.

Contact: M. Goorden, Eindhoven University of Technology, Department of Mechanical Engineering, PO BOX 513, 5600 MB, Eindhoven, the Netherlands, m.a.goorden@tue.nl