Quantifying model quality for supervisory control synthesis - an experimental study [⋆]

Martijn A. Goorden, Michel A. Reniers, and Joanna M. van de Mortel-Fronczak

Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands (e-mail: {m.a.goorden, m.a.reniers, j.m.v.d.mortel}@tue.nl).

Abstract: Supervisory control synthesis is a model-based engineering method to design supervisory controllers for high-tech and cyber-physical systems. Recent advances in synthesis techniques and modelling formalisms allow for synthesis of supervisors for large-scale industrial applications. Yet, the synthesis results depends on the quality and validity of the models used as input. Other model-based techniques such as simulation, testing, and verification provide complementary support in the design process to increase the quality and validity of the models. In this paper, we propose, in addition to the other supporting techniques, eleven modeling aspects to assess the model quality in the context of supervisory control synthesis. Examples of modeling aspects are the interdependency between component models, whether independent subsystems are modeled, and whether the model is annotated with comments. For each modeling aspect, we discuss its importance and describe how it can be quantified. We report on an experiment where 21 models of automated guided vehicles, created by students during a course on Supervisory Control Theory, are evaluated with the proposed modeling aspects. This experiment demonstrates the applicability of the modeling aspects.

Keywords: Discrete-event systems, modeling, supervisory control theory, education

1. INTRODUCTION

In cyber-physical systems, the task of supervisory control is to coordinate the large number of components such that the specified system functionality can be achieved in a safe manner. Supervisory control theory of Ramadge-Wonham (Ramadge and Wonham, 1987) provides means to synthesize a model of the supervisor from a model of the uncontrolled plant and a model of the control requirements. Such a supervisor interacts with the plant by dynamically disabling some controllable events. Then synthesis guarantees by construction that the closed-loop behavior of the supervisor and the plant adheres to all requirements and is furthermore nonblocking, controllable, and maximally permissive.

Advanced synthesis techniques have been introduced recently to overcome the notorious state-space explosion problem. On the one hand, there have been advances in synthesis techniques, e.g. interface-based (Leduc et al., 2009), distributed (Cai and Wonham, 2010), aggregative (Su et al., 2010), multilevel (Komenda et al., 2016), and compositional supervisory control synthesis (Mohajerani et al., 2014). Furthermore, there have been advances in the modeling formalism and the representation, e.g. state-tree structures (Ma and Wonham, 2005), Extended Finite Automata (Skoldstam et al., 2007), and state-based expressions (Ma and Wonham, 2005), resulting in efficient BDD-based implementation, see Miremadi et al. (2012).

The number of industrial applications of supervisory control theory reported in literature is low, see Wonham et al. (2018). Some notable industrial applications include the following ones. A supervisory controller has been synthesized for a theme park vehicle (Forschelen et al., 2012), a patient support table of an MRI scanner (Theunissen et al., 2014), a manufacturing system (Fabian et al., 2014), a waterway lock (Reijnen et al., 2017), and driver assistance systems (Korssen et al., 2017). While these papers propose models of different systems, there is also the question when a model can be considered a 'high quality' model, as there would probably be multiple views on what 'high quality' entails in the context of supervisory control synthesis.

The objective of this paper is to asses the quality of a model by introducing eleven different modeling aspects. Examples of modeling aspects are the interdependency between component models, whether independent subsystems are modeled, and whether the model is annotated with comments. For each modeling aspect, we discuss its importance and describe how it can be quantified. An experiment is performed to determine whether models of industrial applications can be assessed with the proposed modeling aspects. To this end, 21 models of automated guided vehicles, created by students during a course on Supervisory Control Theory, are evaluated. These models are created in the CIF modeling language, see van Beek et al. (2014).

[⋆] This work is supported by Rijkswaterstaat, part of the Ministry of Infrastructure and Water Management of the Government of The Netherlands.

The paper is structured as follows. Section 2 provides the preliminaries of this paper. Section 3 continues by introducing the eleven modeling aspects, discussing their relevance to model quality, and describing how they are quantified. Section 4 describes the experiment and presents the results. The paper concludes with Section 5.

2. PRELIMINARIES

This section provides a brief summary of concepts related to finite automata, extended finite automata, state-based expressions, and supervisory control theory relevant for this paper.

2.1 Finite automata

A finite automaton (FA) is a five-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the (finite) state set, Σ the set of events, $\delta : Q \times \Sigma \to Q$ the partial transition function, $q_0 \in Q$ the initial state, and $Q_m \subseteq Q$ the set of marked states. The event set Σ is partitioned into set Σ_c containing the controllable events and set Σ_u containing the uncontrollable events.

For large systems, it is not feasible to model their behavior by a single FA, as the state space is often too large. Instead, a system can be modeled by a set of several interacting automata G_i (referred to as component models). The combined behavior of the set of automata is given by the synchronous product $G = G_1 \parallel \ldots \parallel G_n$, see Cassandras and Lafortune (2008), which requires synchronization between automata of transitions labeled by the same event.

2.2 Extended finite automata

In Skoldstam et al. (2007), extended finite automata (EFAs) are introduced for modeling systems, which are FAs augmented with bounded discrete variables. An EFA is a seven-tuple $E = (L, V, \Sigma, \to, l_0, v_o, L_m)$, where L is the (finite) location set, V the set of variables, Σ is the set of events, $\to$ the extended transition relation, $l_0 \in L$ the initial location, v_0 the initial valuation, and $L_m \subseteq L$ the set of marked locations. The state of an EFA is the combination of the active location and current variable valuation.

In an EFA, the transition relation is enhanced with guard expressions (conditions) and variable assignments (updates). Formally, the extended transition relation is $\to \subseteq L \times C \times \Sigma \times U \times L$, where C is the set of all conditions and U the set of all updates. A transition is enabled if the associated condition evaluates to true with respect to the current valuation. After taking a transition, the variable valuation is updated according to the associated update. Subsequently, two EFAs can be combined by computing the synchronous product as defined in Skoldstam et al. (2007).

State-based expressions are introduced in Ma and Wonham (2005), and later generalized in Markovski et al. (2010), as a modeling formalism more closely related to the textual formulation of control requirements. This modeling formalism is available in the CIF modeling language van Beek et al. (2014).

2.3 Supervisory control synthesis

The objective of supervisory control synthesis is to construct, based on a plant model and a requirement model, an automaton called a supervisor which function is to dynamically disable controllable events so that the closed-loop system of the plant and the supervisor obeys the following control properties, see Ramadge and Wonham (1987); Cassandras and Lafortune (2008); Wonham and Cai (2019).

- *Safety*: all possible behavior of the closed-loop system should always satisfy the imposed requirements.
- *Controllability*: uncontrollable events may never be disabled by the supervisor.
- *Nonblockingness*: the closed-loop system is able to reach a marked state from every reachable state.
- *Maximal permissiveness*: the supervisor does not restrict more behavior than strictly necessary to enforce safety, controllability, and nonblockingness.

Monolithic supervisory control synthesis results in a single supervisor S derived from a single plant model and a single requirement model, see Ouedraogo et al. (2011). When the plant model and the requirement model are given as a set of models $\mathcal{P}$ and $\mathcal{R}$, respectively, the monolithic plant model P and the requirement model R are obtained by performing the synchronous products.

For large systems, returning a supervisor represented by a single automaton becomes infeasible. The method of Miremadi et al. (2011) allows for a compact representation of the synthesis result. It characterizes the restrictions of the supervisor as guards, extracted during the synthesis procedure. The result is an EFA with a single location and for each controllable event in the plant a selfloop with the derived guard. The supervisor is then represented by the original set of component models, the original set of requirement models, and the extracted guards.

3. MODEL QUALITY

In this section, we describe the model aspects to evaluate the model quality. For each model aspect, we discuss its importance and describe how it is quantified.

(1) Is each component model an elementary part of the system?

The first modeling aspect is one that is sensative to interpretation, as it depends on the system to be modeled. Typically, systems are decomposed into subsystems. Decomposing the system until elementary subsystems are reached eases modeling, as only a small part of the system has to be captured in a single component model. Furthermore, creating component models of elementary parts of the system may result in asynchronous component models (i.e., having no shared events or variables), which can benefit synthesis, see Goorden et al. (2019b).

To quantify this modeling aspect, we report the number of component models that are deemed to be elementary based on expert judgement relative to the total number of component models. A higher number implies more elementary component models.

(2) How strong is the interdependency between component models?

There are several reasons to investigate the interdependency between the component models. For this purpose, two component models are related to each other if they share events or variables. First, the level of interdependency is a better indicator for computational effort of synthesis compared to the uncontrolled state-space size, as argued in Vahidi et al. (2006). Second, more related component models result in less gain in computational effort with synthesis approaches like modular and multilevel synthesis, see Goorden et al. (2019c). Third, assessing the correctness of the model can be more complicated with higher interdependency between component models, as argued in Swartjes (2018).

Several measures exist to quantify the interdependency between components. In this paper, we use the level-n dependency sets of Vahidi et al. (2006) to measure the strength of the interdependency, yet generalized and adapted to EFAs. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ be the set of component models. The level-1 dependency set of an automaton $P_i \in \mathcal{P}$, denoted by $D^1(P_i)$, contains all automata, including itself, to which it is connected by shared events or variables: $D^1(P_i) = \{P \in \mathcal{P} \mid \Sigma_P \cap \Sigma_{P_i} \neq \emptyset \vee V_P \cap V_{P_i} \neq \emptyset\}$. Recursively, we can construct the level-n dependency set with $n \leq 2$, denoted by $D^n(P_i)$, from the level-$(n-1)$ dependency set, indicating the set of automata related to each other in at most n steps. Formally, $D^n(P_i) = \bigcup_{P \in D^{n-1}(P_i)} D^1(P)$. The normalized cardinality of the level-n dependency set for $|\mathcal{P}| > 1$, defined as $d^n(P_i) = \frac{|D^n(P_i)|-1}{|\mathcal{P}|-1}$, quantifies the interdependency between component models. Therefore, $d^n(P_i) = 0$ indicates that component model P_i is not related to any other component model after n steps, while $d^n(P_i) = 1$ indicates that it is related to all other component models after n steps.

(3) Are requirement models elementary, i.e., can they not be split any further?

By splitting requirement models, more elementary ones are created each describing a single control objective. Having smaller requirement models turns out to be beneficial for the applicability of modular and multilevel synthesis, as shown in Goorden et al. (2019c). Furthermore, understanding elementary requirement models is easier, similar to the first aspect of the component models.

To quantify this modeling aspect, we report the number of requirement models that are elementary based on expert judgement relative to the total number of requirement models. The higher the number, the more elementary requirement models are present.

(4) Are there references in requirement models to other requirement models?

Using the EFA modeling formalism it is possible to restrict the behavior of the plant model by using a location reference of another requirement model. By referring to the location of another requirement model, understanding such intertwined requirement models may be difficult. Often, the requirement referred to also acts as an observer to add more complex states to the plant. Furthermore, module-based synthesis approaches, like modular and multilevel synthesis, may fail to cope with such a dependency between requirement models, see Proposition 4 in Goorden et al. (2019d).

The dependency between requirement models can also be quantified by the level-n dependency sets, yet now based on the set of requirement models instead of on the set of component models. For requirements formulated as state-based expressions, we define that these requirements only have a dependency with other requirements if they refer to events or variables introduced in requirements.

(5) Do requirement models introduce new events or variables?

This modeling aspect is related to the previous one in the sense that when requirements introduce new events or variables, state-based requirement can refer to these new events or variables. Furthermore, Supervisory Control Synthesis is built upon the assumption that the desired behavior as described by the requirement models should be part of all possible behavior as described by the plant model. This implies that a requirement should not introduce events or variables. Yet, tooling like CIF and Supremica allow the engineer to introduce new events and variables in requirements and its interpretation is inconsistent throughout the tools.

This modeling aspect is reported as a binary outcome defining whether requirements introduce new events or variables. Special attention is paid to location variables, as these are implicitly constructed for each automaton in CIF. We only count those if they are explicitly used by other requirement automata.

(6) Are there independent subsystems modeled?

By analyzing the dependencies between component models and requirement models as proposed in Goorden et al. (2019a), one is able to verify whether independent subsystems are modeled. Yet, as the modeler has combined these independent subsystems into a single model, the most likely explanation is that requirement models are missing, which should capture dependencies between them.

This modeling aspect is reported as a binary outcome defining whether independent subsystems are present or not.

(7) Are uncontrollable events not unnecessarily blocked in automaton-based requirements?

A common mistake in modeling automaton-based requirements is the omission of uncontrollable events in some locations, leading to controllability issues that have to be solved by synthesis.

We report the number of requirement models that may have unnecessary omitted uncontrollable events relative to the total number of requirements. The higher the number, the more potential controllability issues are present.

(8) What is the length of the guards in the synthesized supervisors, with and without forward reachability analysis?

As noted in Fabian et al. (2014) and Reijnen et al. (2019), representing the synthesis result as a guard for each controllable event is useful for validating the obtained result. The more compact this result, the easier it is to interpret the guards.

We quantify this modeling aspect by the average number of binary operators in the guard for each controllable event.

(9) How many component models, requirement models, event declarations, and variable declarations are provided with comments describing their meaning?

In software engineering, it is common practice to comment the source code, as experiments have shown that commented source code is better maintainable than not commented code, see Tenny (1988). Therefore, we examine whether the implemented models also contain comments to explain the model. As model maintainability relates to the interpretation of the modeled features, like component models, requirement models, events and variables, we examine whether they are provided with such interpretation using comments.

We report the number of commented model features (component models, requirement models, events, and variables) relative to the total number of model features. We do not examine whether the actual content of the comment is sufficient to understand the model feature, as is shown to be possible for software source code in Steidl et al. (2013).

(10) Are templates used?

Templates allow for the re-use of models. In Grigorov et al. (2011), it has been shown that using templates reduces the modeling effort and time. Furthermore, adapting the model is more straightforward, as only the templates have to be understood and changed instead of each component model separately.

We report a binary outcome defining whether templates are used or not to model component or requirement models.

(11) Are groups used to structure the model?

Groups can be used to combine related models. While groups add no modeling expressiveness, they ease the understandability of the model. For example, groups can be used to indicate the system decomposition the modeler had in mind while modeling the system.

This modeling aspect is reported as a binary outcome defining whether groups are used or not.

4. EXPERIMENT

In this section, we describe the experiment performed to analyze models with the model quality aspects from Section 3. As a final assessment for the course on Supervisory Control, groups of two to four students had to provide a model of the plant and a model of the requirement from a system description, perform supervisor synthesis, and validate the controlled system using simulation, all performed in the CIF toolset. The course is given at the graduate level to a varied population of student, most of which are pursuing a master's degree in mechanical engineering (with a specialization in control systems technology or manufacturing systems engineering) and systems & control. There are no specific prerequisites for the course besides a relevant bachelor education. Students were graded for this assignment based on expert judgement.

Fig. 1. Amazon Robotics, or formerly known as Kiva System. Picture from Guizzo (2008)

The goal of the experiment is to determine whether the quality of the models provided by the students can be assessed by the model aspects from Section 3. To this end, we quantify the different aspects for each model. The experiment is set up after running the assignment in the course.

First, the system to be modelled is explained briefly. Subsequently, the results of the experiment are presented. Finally, this section is concluded with a discussion on the results.

4.1 System description

In the assignment, connected automated guided vehicles (AGVs) in warehouses and distribution centers need to be modeled. Several examples of these systems include Amazon Robotics (formerly known as Kiva Systems) (Robotics, 2019), Symbotic (formerly known as CasePick Systems) (Symbotic, 2019), Adapto (Adapto, 2019), and Fleet (Fleet, 2019). The full assignment description can be accessed at a GitHub repository [1].

The Amazon Robotics system consists of numerous AGVs moving along a grid in a warehouse and a centralized server. The centralized server solves the resource allocation problem, i.e., it determines for each AGV which product rack it should move around, which picking station to drive to, and where to store the rack after service by the warehouse worker. The AGVs are differential-drive two-wheeled robots equipped with numerous sensors, as described in D'Andrea and Wurman (2008); Guizzo (2008). An AGV is able to drive forward along a grid and to make turns of 90 degrees. Each AGV can determine its location by reading barcode stickers on the floor with a camera. Arriving at a location in the storage area, the AGV is able to lift or lower a product rack from the ground. As they move autonomously, several sensors are present in each AGV to observe the nearby area and to detect collisions.

The proper functioning of these connected AGVs requires solving several control challenges on different abstraction levels, as described in D'Andrea and Wurman (2008). On

[1] https://github.com/magoorden/WODES2020

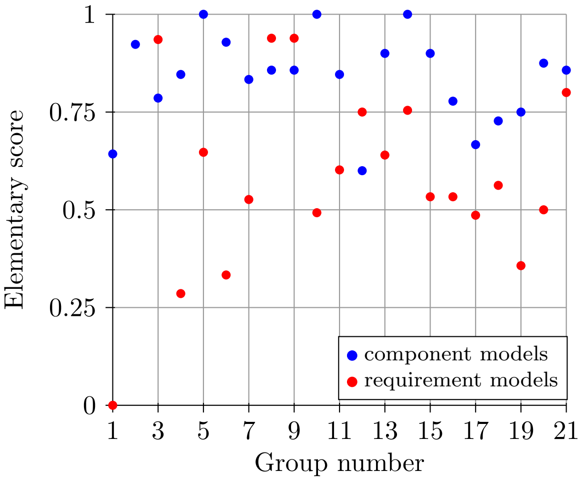

Fig. 2. The score on modeling aspect (1) and (3), which are related to whether the models are elementary. A higher score is better.

the low level, proper feedback loops need to be designed to drive the AGVs with high precision. A level higher, AGVs needs to operate safely in a warehouse environment, ensuring no collisions with each other, humans, or other equipment, and a correct execution order of the different tasks (like raising the rack, turning 90 degrees). Again, one level higher a path planning problem needs to be solved to calculate the best route from the current position to the desired destination. Finally, at the highest level, a dynamic resource allocation problem is solved constantly to assign to each AGV a goal it autonomously needs to fulfill.

In the assignment, the students focused on developing a supervisory controller for AGVs to ensure their safe behavior. We assumed that all other control systems are in place and designed correctly. Several details and design choices were not specified in the assignment to endow the project with a significant level of flexibility also on the formulation side. An important part of the project is therefore to make assumptions. Moreover, the students were highly encouraged to propose other functionalities to the system and changes to the present project. Therefore, we do not assess in this paper the correctness of the models provided by the students.

4.2 Results

In total, 21 models of different groups of students have been assessed. All collected data can be accessed at a GitHub repository [2] . The scores related to the 11 modeling aspects of Section 3 are presented here.

Figure 2 shows the score of each group on the modeling aspects (1) and (3), which assess whether the component models and the requirement models, respectively, are elementary. A model with a higher score is considered to be a higher quality model. For the component models, all scores are above 0.5, which indicates that more than half of the component models describe elementary parts of the system. Three models achieve the highest score of 1, meaning all component models are elementary. For requirement models, the scores are more diverse. The lowest score is 0 and the highest score is 0.94. No score is available for the model of group number 2, as the model did not contain any automata labeled as requirement.

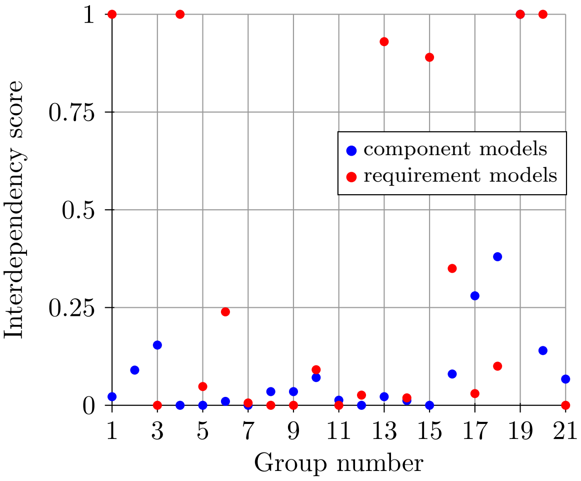

Fig. 3. The score $d^3(P_i)$ and $d^3(R_j)$ on modeling aspect (2) and (4), respectively, which are related to the interdependency between models. The lower a score is the better.

For most of the models, it holds that the score for the requirement models is lower than the score of the component models. Only four models have a higher score for the requirement models than the component models. A low score is often the result of writing down *how* the system should be controlled, i.e., partly specifying the supervisor, in requirement models instead of describing *what* the system should do. Furthermore, several groups tried to provide a single requirement automaton model for each textual requirement given in the assignment description, which also results in non-elementary requirement models.

Figure 3 shows the normalized level-3 dependency score for both the components model and the requirement models, which are the measures for the modeling aspects (2) and (4), respectively, describing the interdependency between models. The normalized level-1 dependency and normalized level-2 dependency scores can be found in the repository. Up to level-3 scores are determined, as higher scores rarely differs from the level-3 score. A model with a lower score is considered to be a higher quality model. No requirement models score is available for the model of group number 2, as the model did not contain any automata labeled as requirement. The scores of group 19 are 1 for both the component models and the requirement models. Therefore, the red mark overlaps the blue mark in the figure.

Most of the normalized level-3 dependency scores are below 0.25, which indicates that each component or requirement model is connected with less than 25% of the other component or requirement models in at most three steps. The higher scores are almost all for the interdependency between requirement models, with the exception of the component models score of group 19. The high component model score is due to shared breakdown and repair events. The high requirement model scores are due to intricate FA requirement models that share numerous events.

[2] https://github.com/magoorden/WODES2020

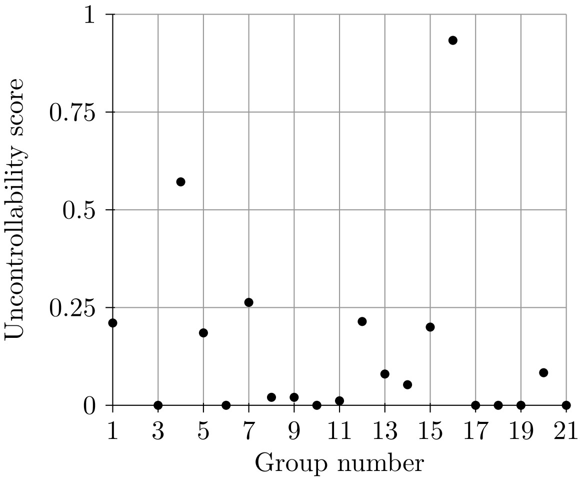

Fig. 4. The score on modeling aspect (7), which is related to potentially unnecessary omitted uncontrollable events. The lower a score is the better.

The perfect score of 0 for the normalized level-3 dependency score for the component models is achieved by five groups, which indicates that their component models do not share any events or variables. The perfect score of 0 for the normalized level-3 dependency for the requirement models is achieved by five other groups. This is achieved by having (almost) only requirements in the form of state-based expressions, such that there is no dependency between requirement models by definition of the dependency score.

Figure 4 shows the score of each group on modeling aspect (7), which is related to potentially unnecessary omission of uncontrollable events in requirement models. A model with a lower score is considered to be a higher quality model. Again, no score is available for the model of group number 2, as this modeling aspect also relates to the requirement models.

Most of the groups managed to keep the score lower than 0.25, which indicates that less than 25% of the requirement models may have unnecessary omitted uncontrollable events. For those groups that only have requirements in the form of state-based expressions, it is easier to have a score of 0. It feels natural to formulate this form of requirement models only for controllable events, as this form explicitly describes the enablement or disablement of an event and a synthesized supervisor may never disable uncontrollable events. The high score of group 16 results in a supervisor which blocks almost all controllable events permanently due to omitting selfloops labeled with uncontrollable events in over 90% of the requirement models.

Table 1 shows the result of the other modeling aspects from Section 3. Modeling aspects (8) and (9) are reported here as true or false questions, while in Section 3 they are quantified differently. The data of the original quantification can be found in the GitHub repository. We report these modeling aspects differently in this table for the following reasons. For modeling aspect (8), which assesses the length of the guards in the synthesized supervisor, we observed that for most groups the average guard length is either exactly 0 or close to 0. For modeling aspect (9), which assesses the comments in the model, we observed that no group has commented events or variables (when present) and that 9 groups also did not comment component or requirement models.

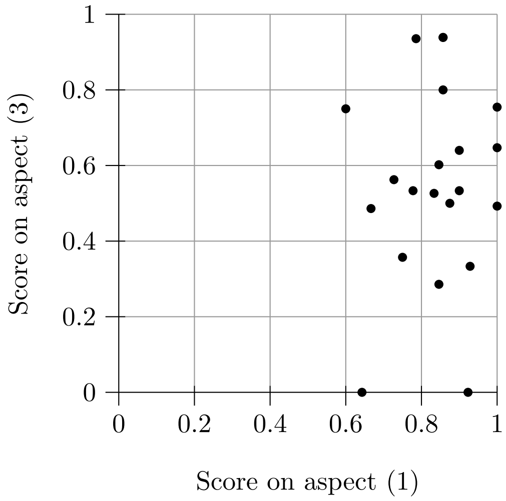

Fig. 5. The relation between the score of elementary component models (modeling aspect (1)) and the score of elementary requirement models (modeling aspect (3)).

Six out of the 21 groups have requirement models that introduce new events or variables not defined in the plant models. An even larger number of groups, to be exact 9, have modeled independent subsystems, i.e., in a single model there are multiple sets of component and requirement models that have no relationship with other parts of the system. In these cases, the students forgot to remove component models they no longer wanted to include in the model or they missed requirements. When focussing on the length of the guards, modeling aspect (8), from those that have an average larger than zero 7 groups have an average of less than 2 binary operators per controllable event. The 'largest' supervisor has an average number of binary operators of 37.3 per controllable event, which has a single guard with 398 binary operators.

The CIF modeling concepts mentioned in modeling aspects (9), (10), and (11) are not taught actively in the course on Supervisory Control. Therefore, lower scores were expected on these aspects. Several groups used groups and templates as they (tried to) model a warehouse system with multiple autonomous vehicles. In such a situation, using groups and templates reduces the modeling effort.

4.3 Discussion

The data reveals that scoring high on one modeling aspect does not imply a high score on the other modeling aspects. There is a low correlation between the scores. Figure 5 illustrates this observation. In this figure, the relation between the score of elementary component models (modeling aspect (1)) and the score of elementary requirement models (modeling aspect (3)) is shown. No clear correlation can be observed. For example, the three models that have the perfect score on modeling aspect (1) have only an average score on modeling aspect (3). Furthermore, their

Table 1. The score on modeling aspects (5), (6), (8), (9), (10), and (11). No scores could be obtained for group number 2 on modeling aspects (5), (6), and (8).

Group number	(5) Do requirement models introduce new events or variables?	(6) Are independent subsystems modeled?	(8) Is length of guard larger than zero?	(9) Are some modeling elements annotated with comments?	(10) Are templates used?	(11) Are groups used to structure the model?
1	no	yes	no	no	no	no
2	–	–	–	no	no	no
3	no	no	no	yes	no	no
4	no	no	yes	no	no	no
5	no	yes	yes	yes	no	no
6	yes	yes	yes	no	yes	yes
7	no	no	no	yes	no	no
8	no	yes	yes	yes	yes	yes
9	no	yes	yes	yes	yes	yes
10	yes	yes	yes	yes	yes	yes
11	no	no	no	yes	no	yes
12	no	yes	yes	yes	no	no
13	no	no	yes	yes	no	no
14	yes	no	yes	yes	no	yes
15	yes	yes	no	yes	yes	no
16	no	yes	no	no	no	no
17	yes	no	yes	no	no	no
18	yes	no	yes	no	no	no
19	no	no	no	no	no	no
20	no	no	no	no	no	no
21	no	no	yes	yes	no	no

score on modeling aspect (3) ranges from 0.49 to 0.75. Similar analyses between other pairs of modeling aspects result in the same conclusion.

The data on modeling aspect (6) about independent subsystems is surprising. Nine out of 21 models contain independent subsystems, i.e., the model can be divided into groups of component and requirement models that have no shared events or variables with any other group. Spotting independent subsystems manually is not straightforward, while an automated analysis reveals this fact immediately. The expectation is that during modeling a system oversight on the model is lost, such that conceptual requirements between component models are forgotten.

Finally, this experiment shows that a model can be assessed with different modeling aspects, yet the question remains how to combine the scores on the individual modeling aspects into a total score reflecting the overall model quality. There are three questions to consider: how to combine binary with non-binary scores, how to weigh each individual score in the total score, and is the set of proposed modeling aspects complete. Answering these questions turns out to be harder than expected, so it is left open in this paper for future research.

5. CONCLUSION

In this paper, we propose eleven modeling aspects to assess the quality of a model in the context of supervisory control synthesis. These modeling aspects have been applied to assess 21 different models of a problem with automated guided vehicles. This experiment has demonstrated the applicability of the proposed modeling aspects to quantify model quality.

Future research may focus on the question how to combine the scores of the individual modeling aspects into a total score that reflects the overall model quality. Furthermore, if the proposed modeling aspects can be quantified automatically, reporting on these during the modeling process may benefit the final model.

ACKNOWLEDGEMENTS

The authors thank Han Vogel and Maria Angenent from Rijkswaterstaat for their support.

REFERENCES

Adapto (2019). URL `https://www.vanderlande.com/warehousing/innovative-systems/storage-asrs/adapto`.

Cai, K. and Wonham, W.M. (2010). Supervisor localization: a top-down approach to distributed control of discrete-event systems. *IEEE Trans. on Automatic Control*, 55(3), 605–618.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer, 2 edition.

D'Andrea, R. and Wurman, P. (2008). Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities. In *IEEE International Conference on Technologies for Practical Robot Applications*, 80–83.

Fabian, M., Fei, Z., Miremadi, S., Lennartson, B., and Åkesson, K. (2014). Supervisory control of manufacturing systems using extended finite automata. In *Formal Methods in Manufacturing*, 295–314. Taylor & Francis Inc.

Fleet (2019). URL `https://www.vanderlande.com/airports/evolutions/fleet`.

Forschelen, S.T.J., van de Mortel-Fronczak, J.M., Su, R., and Rooda, J.E. (2012). Application of supervisory control theory to theme park vehicles. *DEDS*, 22(4), 511–540.

Goorden, M.A., van de Mortel-Fronczak, J.M., Etman, L.F.P., and Rooda, J.E. (2019a). DSM-based analysis for the recognition of modeling errors in supervisory controller design. In *21st Int. DSM Conference*, 121–129.

Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Fokkink, W.J., and Rooda, J.E. (2019b). Component-based modeling of cyber-physical systems for supervisory control synthesis. In *Proc. of FACS*. Springer.

Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Fokkink, W.J., and Rooda, J.E. (2019c). The impact of requirement splitting on the efficiency of supervisory control synthesis. In *Proc. of FMICS*, 76–92. Springer.

Goorden, M.A., van de Mortel-Fronczak, J.M., Reniers, M.A., Fokkink, W.J., and Rooda, J.E. (2019d). Structuring multilevel discrete-event systems with dependency structure matrices. *IEEE Trans. on Automatic Control*. Early access.

Grigorov, L., Butler, B.E., Cury, J.E.R., and Rudie, K. (2011). Conceptual design of discrete-event systems using templates. *DEDS*, 21(2), 257–303.

Guizzo, E. (2008). Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, 45(7), 26–34.

Komenda, J., Masopust, T., and van Schuppen, J.H. (2016). Control of an engineering-structured multilevel discrete-event system. In *Proc. of WODES*, 103–108.

Korssen, T., Dolk, V., van de Mortel-Fronczak, J.M., Reniers, M.A., and Heemels, M. (2017). Systematic model-based design and implementation of supervisors for advanced driver assistance systems. *IEEE Trans. on Intelligent Transportation Systems*, 19(2), 533–544.

Leduc, R.J., Dai, P., and Song, R. (2009). Synthesis method for hierarchical interface-based supervisory control. *IEEE Trans. on Automatic Control*, 54(7), 1548–1560.

Ma, C. and Wonham, W.M. (2005). *Nonblocking supervisory control of state tree structures*. Number 317 in Lecture Notes in Control and Information Sciences. Springer.

Markovski, J., Jacobs, K.G.M., van Beek, D.A., Somers, L.J.A.M., and Rooda, J.E. (2010). Coordination of resources using generalized state-based requirements. *IFAC Proceedings Volumes*, 43(12), 287–292.

Miremadi, S., Åkesson, K., and Lennartson, B. (2011). Symbolic computation of reduced guards in supervisory control. *IEEE Trans. on Automation Science and Engineering*, 8(4), 754–765.

Miremadi, S., Lennartson, B., and Åkesson, K. (2012). A BDD-based approach for modeling plant and supervisor by extended finite automata. *IEEE Trans. on Control Systems Technology*, 20(6), 1421–1435.

Mohajerani, S., Malik, R., and Fabian, M. (2014). A framework for compositional synthesis of modular nonblocking supervisors. *IEEE Trans. on Automatic Control*, 59(1), 150–162.

Ouedraogo, L., Kumar, R., Malik, R., and Åkesson, K. (2011). Nonblocking and safe control of discrete-event systems modeled as extended finite automata. *IEEE Trans. on Automation Science and Engineering*, 8(3), 560–569.

Ramadge, P.J.G. and Wonham, W.M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1), 206–230.

Reijnen, F.F.H., Goorden, M.A., van de Mortel-Fronczak, J.M., and Rooda, J.E. (2017). Supervisory control synthesis for a waterway lock. In *Proc. of CCTA*, 1562–1568.

Reijnen, F.F.H., Goorden, M.A., van de Mortel-Fronczak, J.M., and Rooda, J.E. (2019). Modeling for supervisor synthesis - a lock-bridge combination case study. Submitted.

Robotics, A. (2019). URL `https://www.amazonrobotics.com/`.

Skoldstam, M., Åkesson, K., and Fabian, M. (2007). Modeling of discrete event systems using finite automata with variables. In *Proc. of CDC*, 3387–3392.

Steidl, D., Hummel, B., and Juergens, E. (2013). Quality analysis of source code comments. In *Proc. of ICPC*, 83–92.

Su, R., van Schuppen, J.H., and Rooda, J.E. (2010). Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Trans. on Automatic Control*, 55(7), 1627–1640.

Swartjes, L. (2018). *Model-Based Design of Baggage Handling Systems*. Ph.D. thesis, Eindhoven University of Technology.

Symbotic (2019). URL `https://www.symbotic.com/`.

Tenny, T. (1988). Program readability: procedures versus comments. *IEEE Trans. on Software Engineering*, 14(9), 1271–1279.

Theunissen, R.J.M., Petreczky, M., Schiffelers, R.R.H., Beek, D.A.v., and Rooda, J.E. (2014). Application of supervisory control synthesis to a patient support table of a magnetic resonance imaging scanner. *IEEE Trans. on Automation Science and Engineering*, 11(1), 20–32.

Vahidi, A., Fabian, M., and Lennartson, B. (2006). Efficient supervisory synthesis of large systems. *Control Engineering Practice*, 14(10), 1157–1167.

van Beek, D.A., Fokkink, W.J., Hendriks, D., Hofkamp, A., Markovski, J., van de Mortel-Fronczak, J.M., and Reniers, M.A. (2014). CIF 3: model-based engineering of supervisory controllers. In *Proc. of TACAS*, 575–580.

Wonham, W.M., Cai, K., and Rudie, K. (2018). Supervisory control of discrete-event systems: a brief history. *Annual Reviews in Control*, 45, 250–256.

Wonham, W.M. and Cai, K. (2019). *Supervisory Control of Discrete-Event Systems*. Communications and Control Engineering. Springer.