Modeling the Effects of Software on Safety and Reliability in Complex Embedded Systems

Max Steiner, Patric Keller, and Peter Liggesmeyer

AG Software Engineering: Dependability, TU Kaiserslautern,
{steiner,pkeller,liggesmeyer}@cs.uni-kl.de

Abstract. The development of autonomous vehicle systems demands the increased usage of software based control mechanisms. Generally, this leads to very complex systems, whose proper functioning has to be ensured. In our work we aim at investigating and assessing the potential effects of software issues on the safety, reliability and availability of complex embedded autonomous systems. One of the key aspects of the research concerns the mapping of functional descriptions in form of integrated behavior-based control networks to State-Event Fault Tree models.

Keywords: safety analysis, reliability analysis, state-event fault trees

The final publication is available at http://link.springer.com/chapter/10.1007/978-3-642-33675-1_43

1 Introduction

Recent trends concerning the automation of high-tech products like cars, airplanes and trains lead to embedded systems of tremendous size and complexity. This development particularly affects the quality of those systems. Important (non-functional) quality characteristics often related to in this context are safety, reliability and availability. The compliance with predefined quality goals is ensured by the increased usage of software-based mechanisms. Depending on the nature of the considered systems, this may be crucial. Especially, in cases where software is applied to fulfill certain safety requirements.

The way software affects safety-critical systems is manifold. But not only safety is a factor influencing the final quality of a product. Faults introduced due to misinterpretation, wrong design decisions, or implementation errors may lead to unreliable services, or to reduced functionalities, which might affect the availability of the whole system. The other way around, unreliable or non-available services may again end up in safety critical events, e.g., the failure of anti-blocking systems of cars in emergency cases. In this regards, important questions focus on determining the impact of software faults on safety, reliability and availability of complex software-intensive embedded systems.

In general, the range of the considered systems only differs slightly with respect to its principle composition and its mission profiles: Data generated from sensors, responsible for the acquisition of environmental information, is collected and prepared for further processing. Based on the provided data, decisions are inferred determining the behavior of the system, e.g., whether or not a collision detection system should trigger an alarm in case of an imminent collision with other objects. Depending on the sort of system, the decision may also control actuators like engines and steering devices, as it is the case in autonomous vehicles.

Our efforts center on finding a way to draw conclusions about influences of software on quality characteristics like reliability, safety and availability of complex software-intensive embedded systems of autonomous vehicles. The main focus lies on how to map corresponding relations of software-artifacts of functional descriptions of integrated Behavior-Based Control (iB2C) networks [9] to safety and reliability analysis models using State Event Fault Trees (SEFT) [3]. Analyzing these mappings shall facilitate answering questions like:

- How do software-artifacts react on corrupted input data?
- At which point do software components fail?
- Given assumptions about the reliability of the input data, how imminent is the occurrence of certain unwanted/critical system-level events with respect to safety, reliability and/or availability?

Our research relies on the description of an autonomous vehicle demonstrator system called RAVON (Robust Autonomous Vehicle for Off-Road Navigation), which has been developed by the Robotics Research Lab at the University of Kaiserslautern.

The remainder is structured as follows. In Sect. 2 we provide an overview about the type of the demonstrator system used to conduct the studies. In Sect. 3 we review some of the related modeling techniques. Subject of Sect. 4 is the discussion of how to use the modeling elements of State-Event Fault Trees to map the interrelations between reliability and safety. Sect. 5 describes the way of using the technique to assess the overall system safety based on the information about interference to sensor (data) failure. In Sect. 6 we provide the results of an analysis of safety subsystems of our demonstrator system. We conclude with Sect. 7 and provide a short outlook of future work.

2 Demonstrator: RAVON

RAVON is a mobile robot developed by the Robotics Research Lab at the University of Kaiserslautern to research off-road navigation of autonomous vehicles [11]. It uses several different sensor systems to perceive its environment. Laser scanners are used for front and back distance measuring and obstacle detection. Additional information about the environment is added with cameras. Pressure sensitive bumper systems at the front and the back of the robot cause an emergency stop if triggered. All sensor systems are processed by the software part of the system, which then generates control values for the actuators. The control software is realized using the behavior-based control architecture iB2C [9].

RAVON has four electric wheel hub motors bringing it to a maximum velocity of 10 km/h. Its total weight of 750 kg brings up a potential risk of serious damage to itself and the environment including injuries in collisions with humans. It is therefore imperative that the system is analyzed for residual risks despite of the built-in safeguards. Unfavorable environmental conditions could lead to a non-detection of an obstacle by one or more sensors. In a safety analysis it has to be examined how such a fault affects the driving behavior and if it leads to an accident.

2.1 Behavior-based Network Description.

In RAVON a behavior-based control framework named iB2C [9] is used. The control architecture is a layer-based network consisting of several behavior modules. An iB2C behavior has several behavior-specific input and output signals to determine its influence on the behavior network. In addition to the behavior signals a behavior module has an input and output vector for values controlled by the behavior.

Interaction between behavior modules is realized by fusion behaviors, which combine outputs from different modules to one single output. Two different kinds of fusions are used in RAVON: maximum fusion and weighted fusion. The maximum fusion forwards the output vector of the behavior with the highest activity. The weighed fusion merges the output vectors of the input behaviors, which are weighted with their corresponding activity. All driving motions can be altered by safety behaviors by slowing down or changing direction.

In [9] some more complex design patterns for iB2C are described. As they are used frequently, it is planned to implement direct translations or at least translation templates for SEFT generation. The most important and most frequently used pattern is the behavioral group pattern. A behavioral group is a hierarchical abstraction of several behaviors, which manipulate the same output values. They are combined into a group, which looks like a behavior from the outside.

3 Related Work

Cheung et al. [1] provided a framework for reliability prediction of software components at the architectural level. Their model consists of three phases: First, determine system states including normal and abnormal behavior. Second, determine transitions with help of hidden Markov models and other sources of information, which is still a challenge. Third, solve the model and compute reliability via steady state analysis and the calculation of the probability of being in a failure state. They only consider reliability in their analysis. It might be possible to extend the technique to analyze safety and availability. We did not use this approach, because with SEFTs we can model everything Markov-models can describe, and additionally SEFTs can be used to analyze safety. Also, we have a tool, with which we can create and analyze SEFTs.

Min et al. [7] propose template software fault trees for Ada95 code for safety analysis on a very low abstraction level. They use standard fault trees, which do not take into account the component structure of larger systems. Only implementation errors are analyzed, and McDermid states in [6] "that many safety problems relate to requirements faults, not mistakes in coding". As a consequence, additional analyses are needed to check for faults in earlier development stages.

Lano et al. [5] propose HAZOP (hazard and operability study) guide words for the analysis of object-oriented models in UML. They provide new guide word interpretations for state transition, class and sequence diagrams. HAZOP is used to find system failures, which themselves have to be analyzed further. It can be used before a fault tree analysis to determine possible top events.

Förster and Trapp [2] developed a Fault Tree based method to cope with the problem of few or no information about failures early in the development. They base the analysis on Component Fault Trees (CFT) [4] to model system composition. The uncertainty of (software) safety probabilities for basic events is modeled with the help of probability distributions over intervals instead of single probabilities. For calculating an overall probability distribution, every interval is sampled many times and the overall probability per sample is calculated. CFTs offer the possibility to model big systems with many components in a structured way. In embedded systems the state of the system changes as a reaction on events. Therefore a reliability or safety analysis has to be able to consider states as well as events. CFTs do not support the distinction of states and events, which is possible in Petri-nets.

Sacha proposes in [10] an analysis called Transnet using structured Petri-nets. The analysis based on Petri-nets is similar to ours as we also conduct a reachability analysis. One drawback with Petri-nets is that the nets are considerably large, even for relatively small systems. SEFTs combine the ability to model states and events of Petri-nets with the ability to model component-wise like CFTs.

4 State-Event Fault Trees

In our approach we use State-Event Fault Trees (SEFT) [3] to analyze safety and reliability from a functional system description. SEFTs are a combination of deterministic state machines, Markov chains and Fault Trees [14], and can model system states as well as timing constraints. Quantitative evaluation is done by translating SEFTs into deterministic and stochastic Petri-nets (DSPN [8]). DSPNs are an extension of Generalized Stochastic Petri-nets (GSPN) to additionally model deterministic delay. A DSPN is a timed variant of Petri-nets, which means the time a transition waits, before it fires, is specified.

In [3] a SEFT-to-DSPN translation algorithm, which we use, is described. A SEFT has a finite state space, and a component is in exact one state at a time – the active state – and stays in that state for some time. The state of a component is described as a state expression: "component c is in state s

at time t". A probability can be assigned to a state expression for each point in time. Contrary to a state, an event has no duration. State transitions are events, but additional events are possible. As in DSPNs events can occur after a deterministic or exponentially distributed probabilistic delay. They can also be triggered by other events. In addition to the standard boolean gates used in Fault Trees (AND, OR), several new gates are introduced. For example `History-AND` remembers events that have occurred in the past, and `Priority-AND` additionally remembers if they have occurred in a given order.

As a result, one receives a directed acyclic graph (one cause can trigger multiple effects). In the graph causal loops are forbidden, except if some explicit delay is introduced into the cycle. Event ports allow triggering relations across component borders. SEFTs are constructed like Fault Trees: start with an undesired state or event, and find influences or causes. Basic events of FTs correspond to solitary exponentially distributed events in SEFTs. Subcomponents can be modeled by other means like Markov chains and easily included into the SEFT.

5 Modeling Approach

In this Section, the approach to apply SEFTs on functional system models is described on the example of RAVON. Starting with the general process chain, a description of all steps from iB2C behavior modules to failure probabilities is given.

5.1 Process Chain

Fig. 1 shows the different steps of our approach to analyze the safety and reliability of a functional model. First, a SEFT of the behavior network is developed describing how failures are propagated through the system hierarchy. After modeling the structure, probabilities for basic events are entered. This is done using the tool ESSaRel [13]. The SEFT can not be analyzed directly, so it is converted into a DSPN via an export function to the TimeNET [12] format. TimeNET is a tool for analyzing DSPNs. Now, interesting places have to be identified, for which a reachability analysis will be done. Usually, such places are the failure states of the topmost component in the model. In TimeNET probabilities for system states can be calculated. It can be expressed if a desired system state is reached with a certain probability, or an undesired state (failure state) is reached with a certain probability. This probability can be compared to a threshold to see if predefined constraints are met.

5.2 Translation from iB2C to SEFT

Before an analysis can be done, the system has to be modeled with regard to failure propagation. In ESSaRel a system can be modeled component-wise. Single iB2C behavior modules, fusion behaviors, whole behavior groups or behavior

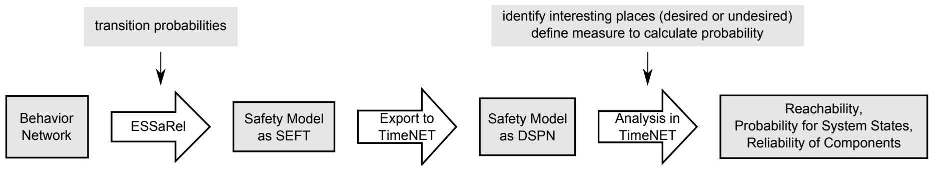
transition probabilities
identify interesting places (desired or undesired) define measure to calculate probability
Behavior Network
ESSaRel
Safety Model as SEFT
Export to TimeNET
Safety Model as DSPN
Analysis in TimeNET
Reachability, Probability for System States, Reliability of Components

Fig. 1. Process chain

patterns can be seen as components. In the following, two architectural components seen in iB2C are briefly described with a translation scheme into a SEFT. More detailed descriptions of iB2C components can be found in [9].

Behavior Modules. A behavior usually controls one or more control values like a velocity or a joint angle. These values have to be within certain boundaries in certain situations for the system to be in a state, in which no safety condition is violated. Additionally, the activity of a behavior module is closely related to the environmental situation. A model of such a behavior observes deviations from desired values of control values and activities.

Some modeling rules can be formulated: First, determine possible states of control values and/or activity. Possible states are one or more failure states and a state for normal operation. Next, determine causes for state transitions and model them as events. Events that are caused by the environment of the system should be modeled with a failure rate (either deterministic or exponentially distributed events). Events that are triggered by other events from other components are of the type "Triggered Event".

In Fig. 2 an example SEFT of a simple behavior module is shown. It consists of one OK state (out_vel_ok) and one failure state (out_vel_high). The init event is pointed at the OK state. Two events (no_obstacle_detected and sensors_working) switch between the states. The events are connected to event outports (black triangles), which are used to connect this component to others.

Fusion Behaviors. As stated in Sect. 2.1, behaviors that influence the same control values are merged with fusion behaviors. The failure modes of a fusion behavior result from the failure modes of its input behaviors. For a maximum fusion only the failure modes of the behavior with the highest activity are propagated. The maximum fusion results in an "OR" gate for the failure modes and an "History-AND" gate for the normal modes. That means if at least one of the behaviors is in the failure mode, the output of the fusion behavior is also in the failure mode. On the other hand, if both input behaviors are in normal mode, the fusion behavior is also in normal mode.

For the weighted fusion there are a few ideas that still have to be implemented: As for the maximum fusion the failure modes of the weighted fusion depend on the ones from the input behaviors. In contrast to the maximum fusion the weighted fusion has an infinite number of states. To be able to model

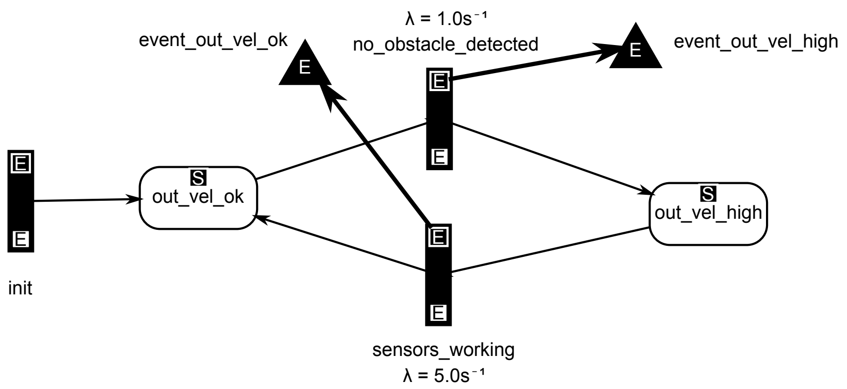

Fig. 2. Example SEFT for a simple behavior

these with a SEFT, they have to be classified. For the first modeling attempt, the weighted fusion can be abstracted as a maximum fusion.

In Fig. 3 an example SEFT of a maximum fusion behavior with two input behaviors is shown. The states are determined by the merged behaviors. At least one OK state and one failure state is needed. Events are triggered events in contrast to the exponentially distributed events of behavior modules. State changes depend on the inputs of this component (white triangles are event inports). Inputs of failure events are merged with an OR gate (≤ 1), inputs of OK events with a History-AND (H&). Like in the previous example, the events are also connected to event outports (black triangles).

5.3 Translation from SEFT to DSPN

The translation from SEFT to DSPN is done using the translation rules described in [3], which are implemented as an export function in ESSaRel. SEFT models are translated component-wise, events become transitions and states become places. The init node transforms into an initial marking of the place corresponding to the state, to which the init node is connected. Gates are translated according to a dictionary in [3]. Component ports also have a special translation.

At the bottom of Fig. 4 an example SEFT and its corresponding DSPN are shown. The SEFT model depicts one component with two states, two events and two event outports to connect the component to others. This SEFT is translated into the DSPN shown below the SEFT. The states are directly translated into places. Same names indicate corresponding states and places. The places have an additional prefix indicating the component. This is needed to distinguish places with the same name that come from different components. The same holds for events and transitions. The exponentially distributed trigger rates of the events in this example are translated into exponentially distributed delays of the DSPN transitions. The two event outports are translated to one place and one transition each. Other components can be attached to these transitions.

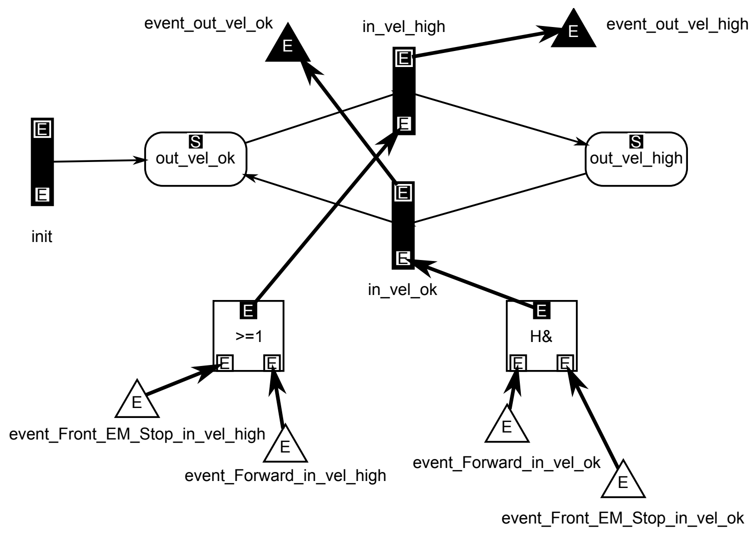

Fig. 3. Example SEFT for a maximum fusion with two input behaviors (Front_EM_Stop and Forward)

5.4 Analysis in TimeNET

When the SEFT is translated into a DSPN, it can be analyzed with TimeNET. The translation is done, because with DSPNs it is possible to calculate the reachability of states and the probability for the system to end up in a certain state. It is possible to draw conclusions about system reliability by calculating the probability that the system is in the OK state. If there exists a repair transition to leave each of the failure states, availability can be measured by calculating the probability of the system to be in the OK state at the top component at a given time. Also, safety can be analyzed by calculating the probability for the system to be in a safety-critical failure state. The same can be done for each single component, if desired.

Before the analysis can be done, a separate measure has to be defined for each observed state. An example measure `result = P{#P1=1};` would result in the probability that place `#P1` contains one token. TimeNET then calculates the probabilities for the system to be in the defined states during runtime.

6 Results

As an example a small part of the RAVON system is modeled. The modeled part controls the forward velocity of RAVON. Two different behaviors can slow down the robot by overwriting the velocity value. Fig. 4 shows the translation of the whole example. In the top-left part the SEFT model is shown consisting of five components, which are connected via their component ports. Two components are connected via a fusion behavior (second component to the top). On the right, the corresponding DSPN can be seen. Below, one component is enlarged to depict the translation in more detail:

`(G) Front Emergency Stop` is a behavior group, abstracted as a module, which could be modeled in more detail later on. It uses the bumper sensors to detect obstacles by touch and reduces the output velocity down to zero if an obstacle is detected. We identified “output velocity too high” as the only failure mode of this behavior at this abstraction level, making this one failure state in the SEFT model. With the OK state (out_vel_ok), we need events, which enable the transitions between the failure state and the OK state. As initial state, the OK state is selected, because it is assumed that the system is OK at the beginning.

In the SEFT model of `(G) Front Emergency Stop` each event has a trigger rate (λ) assigned, which indicates how often the events are happening per second. At the moment, this rate has been arbitrarily chosen, because it is not yet interesting to have accurate values for this small example. In case of the event “no obstacle detected”, it is the failure rate of the obstacle detection, the rate of the other event is the rate, with which the system detects obstacles, depending on sensor speed.

At the bottom of Fig. 4 the enlarged translation of the previous SEFT is shown. States are translated to places, events to transitions, the init event to an initial marking, and the component ports to transit structures consisting of places and triggered events. The trigger rate of the modeled events is translated into a transition with an exponentially distributed delay.

With this model, the probability, that the modeled components end up in a failure state, can be determined. For example, the reliability of the component `(G) Front Emergency Stop` depends on the probability, that it is in the OK state. This probability depends on the failure and repair rates given in the model and can be determined by an analysis with TimeNET. TimeNET can do a reachability analysis and calculate probabilities for the system to be in certain states.

Using this method, it is possible to analyze safety, reliability and availability of complex behavior-based systems. The propagation of failures from sensors through the control software can be modeled and quantitatively analyzed, provided reliability data for the sensors is available. The same could be achieved with DSPNs, but with greater modeling effort for larger systems.

The advantage of the proposed method is, that is possible to model software-intensive embedded systems with system states and externally triggered events (sensor inputs). The modeling in SEFT is easier than with Petri-nets, because complex systems can be modeled component-wise, instead of building one large net. Also, system states and events, that contribute to a failure, can be combined using logic gates like in fault trees.

7 Conclusion

We proposed a method to apply State-Event Fault Trees on functional descriptions of hardware and software to analyze safety, reliability and availability. The main contribution are translation rules from the functional description of behavior-based systems to an SEFT model. In this SEFT model desired and undesired system states are selected. The analysis of the SEFT is done via transformation into a DSPN. The results are probabilities that the system or system components are in previously selected states. From this probabilities the system reliability, availability or, in case of safety related components, the system safety can be measured.

The next steps will be to develop translation schemes for all of the patterns, and to expand the existing translations.

Acknowledgement. This work was funded by the German Ministry of Education and Research (BMBF) in the context of the "Virtuelle und Erweiterte Realität für höchste Sicherheit und Zuverlässigkeit Eingebetteter Systeme – Zweite Phase" (ViERforES II) project. We also thank our colleagues from the Robotics Research Lab of the University of Kaiserslautern for providing us with the input data.

References

1. Cheung, L., Roshandel, R., Medvidovic, N., Golubchik, L.: Early prediction of software component reliability. In: Proceedings of the 30th international conference on Software engineering. pp. 111–120. ICSE '08, ACM, New York (2008)
2. Förster, M., Trapp, M.: Fault tree analysis of software-controlled component systems based on second-order probabilities. In: ISSRE 2009 proceedings (2009)
3. Kaiser, B., Gramlich, C., Förster, M.: State/event fault trees–a safety analysis model for software-controlled systems. Reliability Engineering & System Safety 92(11), 1521 – 1537 (2007), sAFECOMP 2004, the 23rd International Conference on Computer Safety, Reliability and Security
4. Kaiser, B., Liggesmeyer, P., Mäckel, O.: A new component concept for fault trees. In: 8th Australian Workshop on Safety Critical Systems and Software. Canberra (October 2003)
5. Lano, K., Clark, D., Androutsopoulos, K.: Safety and security analysis of object-oriented models. In: Safecomp (2002)
6. McDermid, J.: Software hazard and safety analysis. In: Formal Techniques in Real-Time and Fault-Tolerant Systems. LNCS, vol. 2469, pp. 23–34. Springer-Verlag Berlin Heidelberg (2002)
7. Min, S.Y., Jang, Y.K., Cha, S.D., Kwon, Y.R., Bae, D.H.: Safety verification of ada95 programs using software fault trees. In: Computer Safety, Reliability and Security (1999)
8. Priese, L., Wimmel, H.: Petri-Netze. Springer (2008)
9. Proetzsch, M., Luksch, T., Berns, K.: Development of complex robotic systems using the behavior-based control architecture iB2C. Robotics and Autonomous Systems 58(1), 46–67 (January 2010)

10. Sacha, K.: Safety verification of software using structured petri nets. In: Computer Safety, Reliability and Security (1998)
11. Schäfer, B.H.: Robot Control Design Schemata and their Application in Off-road Robotics. Ph.D. thesis, TU Kaiserslautern (2011)
12. TU Berlin, R.: Timenet 4.0. `www.tu-ilmenau.de/TimeNET` (2007)
13. TU Kaiserslautern, A.s., IESE, F.: Embedded system safety and reliability analyzer (essarel). `http://www.essarel.de` (2009)
14. Vesely, W., Goldberg, F., Roberts, N., Haasl, D.: Fault Tree Handbook. U.S. Nuclear Regulatory Commission (1981)

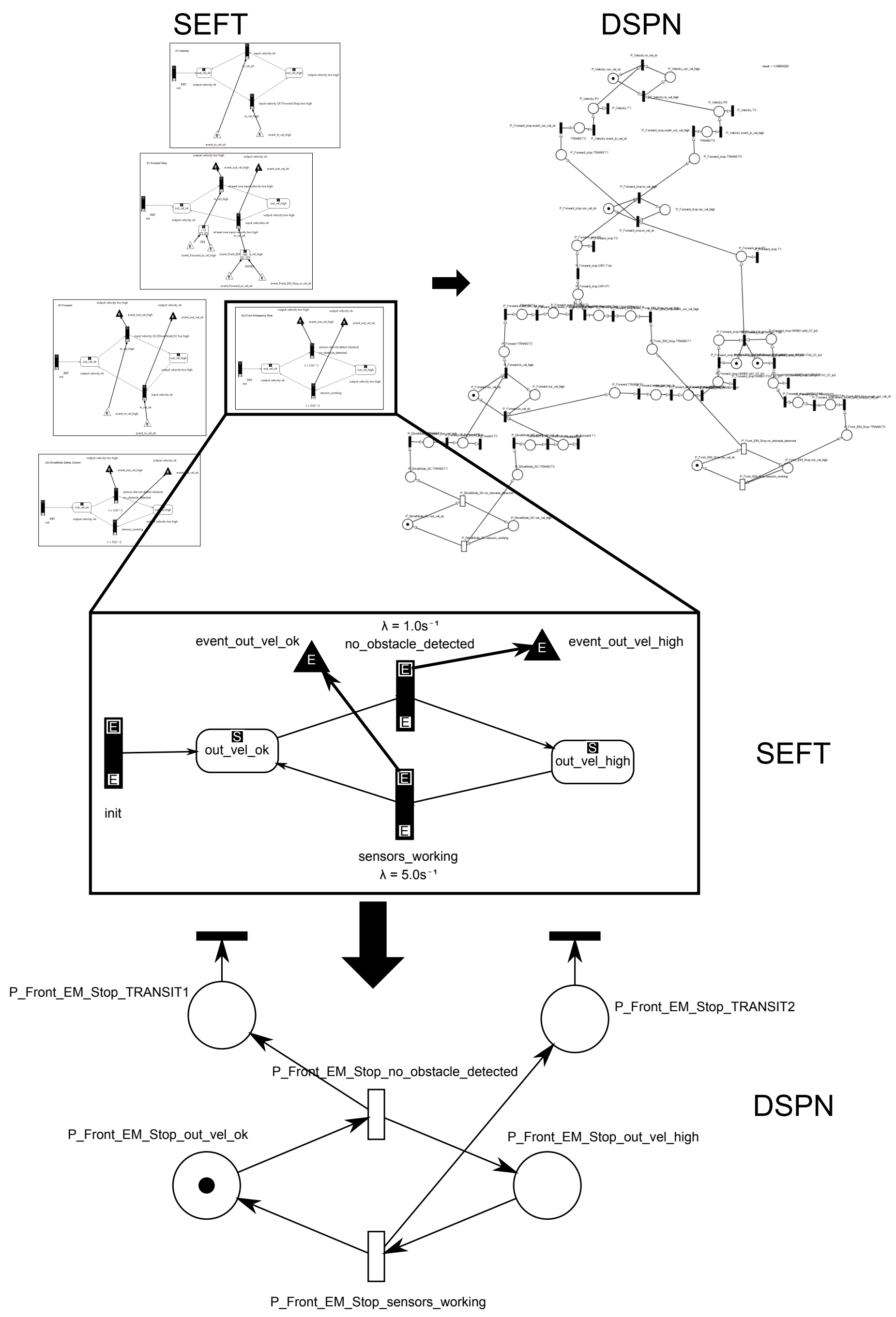

Fig. 4. Top-left: the SEFT model of the example system, consisting of five components. Top-right: the corresponding DSPN. Middle: one enlarged component of the SEFT. Bottom: the enlarged translation to DSPN of the SEFT