Computer-Aided PHA, FTA and FMEA for Automotive Embedded Systems

Roland Mader[1,2], Eric Armengaud[1,3], Andrea Leitner[2], Christian Kreiner[2], Quentin Bourrouilh[1], Gerhard Grießnig[1,2], Christian Steger[2], and Reinhold Weiß[2]

[1] AVL List GmbH
{roland.mader,quentin.bourrouilh,gerhard.griessnig}@avl.com
[2] Institute for Technical Informatics, Graz University of Technology
{andrea.leitner,christian.kreiner,steger,rweiss}@tugraz.at
[3] Virtual Vehicle Competence Center (ViF)
eric.armengaud@v2c2.at

Abstract. The shift of the automotive industry towards powertrain electrification introduces new automotive sensors, actuators and functions that lead to an increasing complexity of automotive embedded systems. The safety-criticality of these systems demands the application of analysis techniques such as PHA (Preliminary Hazard Analysis), FTA (Fault Tree Analysis) and FMEA (Failure Modes and Effects Analysis) in the development process. The early application of PHA allows to identify and classify hazards and to define top-level safety requirements. Building on this, the application of FTA and FMEA supports the verification of a system architecture defining an embedded system together with connected sensors and controlled actuators. This work presents a modeling framework with automated analysis and synthesis capabilities that supports a safety engineering workflow using the domain-specific language EAST-ADL. The contribution of this work is (1) the definition of properties that indicate the correct application of the workflow using the language. The properties and a model integrating the work products of the workflow are used for the automated detection of errors (property checker) and the automated suggestion and application of corrective measures (model corrector). Furthermore, (2) fault trees and a FMEA table can be automatically synthesized from the same model. The applicability of this computer-aided and tightly integrated approach is evaluated using the case study of a hybrid electric vehicle development.

1 Introduction

Nowadays automotive embedded systems incorporate up to 70 microcontrollers that communicate via bus systems, gather sensor data and command actuators of the vehicle. This complexity still increases. One of the reasons is the shift of the automotive industry towards powertrain electrification that goes along with the introduction of new sensors, actuators and functions. The automotive embedded system is responsible for the management of the components (e.g.

F. Flammini, S. Bologna, and V. Vittorini (Eds.): SAFECOMP 2011, LNCS 6894, pp. 113–127, 2011.

high voltage battery, electric motor) that can be found in electrified vehicles and components (e.g. transmission, engine) which are parts of traditional vehicles as well. It is obvious that the correct and safe operation of an electrified vehicle depends on the correct operation of its embedded system.

Due to the safety-criticality of automotive embedded systems, they are developed according to rigorous development processes such as defined by ISO 26262, the functional safety standard for the automotive domain. These development processes incorporate the application of analysis techniques. Among the applied techniques are the following:

- **PHA (Preliminary Hazard Analysis):** PHA [12] is an analysis technique that is qualitatively applied early in the development process by a team of people with a wide variety of expert knowledge and skills. The purpose of PHA is the identification, classification and assessment of potential hazards of a newly developed vehicle, caused by failures. The early knowledge about these hazards allows to define top-level safety requirements, even if less detailed and quantitative information about the vehicle is available.
- **FTA (Fault Tree Analysis):** FTA [12] belongs to the group of deductive analysis techniques. FTA starts with the identified hazards and tracks them back to possible faults that can lead to the occurrence of the top faults. Relationships between effect and cause are defined using logical operators that combine the effects of events. This analysis technique can be applied to verify a system architecture defining an embedded system together with connected sensors and controlled actuators.
- **FMEA (Failure Modes and Effects Analysis):** FMEA [12] belongs to the group of inductive analysis techniques. Individual failures of system components are considered and their causes (e.g. fault of a component) are identified. Then the effects on the complete system in terms of hazards are determined. This analysis technique can be applied to verify a system architecture as well.

This work presents a modeling framework with automated analysis and synthesis capabilities. This modeling framework supports an ISO 26262-compatible automotive safety engineering workflow. Results are annotated using the domain-specific language EAST-ADL [1]. The contribution of this work is (1) the definition of properties that indicate the correct application of the workflow using this language. The properties and a model integrating the work products of the workflow are used for the automated detection of errors (property checker) and the automated suggestion and application of corrective measures (model corrector). Furthermore, (2) fault trees and a FMEA table can be automatically generated from the model allowing the qualitative application of FTA and FMEA. The fault trees and the FMEA table are consistent to the PHA results. Minimum cut sets can be automatically extracted from the synthesized fault trees.

The remainder of this work is organized as follows. Section 2 reviews related work. Section 3 describes the ISO 26262-compatible safety engineering workflow. Section 4 describes how the workflow can be supported by the property checker.

Section 5 describes how the model corrector can be used to correct errors and how fault trees and a FMEA table can be automatically generated. Section 6 describes the experimental evaluation of the approach using the case study of a hybrid electric vehicle development. Finally Section 7 concludes this work.

2 Related Work

Approaches that aim on supporting safety engineering by fault tree generation and/or FMEA generation are reviewed in this section. An approach that combines system architecture modeling and FTA is described in [14]. The approach allows continuous assessment of an evolving system design. A system model is input to HAZOP (Hazard and Operational Studies). Each component of the system model is analyzed and component failure modes are determined. The HAZOP result is a model that defines failure modes that can be observed at the component outputs as results of internal component malfunctions as well as deviating component inputs. In [13] an extension of [14] is presented that allows FMEA table generation. In [2] the extended approach is integrated with an EAST-ADL modeling tool using a model transformation technique. This allows synthesis of fault trees and FMEA tables from EAST-ADL models.

In [4] tool support for automated FMEA generation is presented. Input to the presented method is a component model of a system including so called safety interfaces that can be automatically generated. Safety interfaces can be seen as formal descriptions of the components in terms of failures affecting the components. From the safety interface descriptions cFMEAs (Component Failure Modes and Effects Analysis) can be created for each component. Subsequently the cFMEAs are input to the generation of a system-level FMEA.

A methodology that combines safety analyses and a component-oriented, model-based software engineering approach is described in [3]. The authors aim on supporting safety analyses in the earlier stages of development. A hierarchical model for component-based software engineering is available. The model allows to define a failure specification and a failure realization as well as a functional specification and a functional realization for each software component. Fault trees can be generated from the component model.

In [10] a computer-aided approach to fault tree generation is described. The approach requires the creation of a model of the system under investigation. This model describes system structure, system behavior as well as the flows of information and energy through the system. Moreover top events are defined for system parameters such as component inputs or component outputs. This model is input to a trace-back algorithm that generates a fault tree.

The authors of [11] integrate architectural modeling languages with safety analysis languages to improve consistency. When a safety-critical software architecture is developed an initial architecture is proposed. This architecture is annotated and enriched with safety-relevant information. Safety analysis of the architecture is carried out. Results influence the software architecture. This design and analysis process is cyclic. A meta model for component-based, safety-aware

architectures (SAA) is available allowing to complement architectural descriptions with safety-relevant information such as safety objectives and mitigation means. Meta models for FTA and FMECA (Failure Modes, Effects and Criticality Analysis) are proposed. A tool implementation is presented that allows the generation of FTA models and FMECA models from a SAA model.

Each of the reviewed approaches uses a system model describing the system components complemented with safety-relevant information (typically about faults and failures and their propagation). This underlying model is used by all approaches as input to fault tree generation and/or for FMEA table generation, supporting the application of FTA and/or FMEA. In none of these approaches the application of the workflow for creation of the underlying model is aided by automated checking or model correction. Our approach supports this, supporting fault tree generation and FMEA table generation and furthermore elaboration of the underlying model that integrates the work products of the presented workflow. This strongly supports coping with the complexity imposed by the embedded system of an electrified vehicle.

3 Safety Engineering Workflow

We present an ISO 26262-compatible, automotive safety engineering workflow that is based on the workflows described in [15] and [9]. The workflow can be subdivided into multiple phases. Iterations between phases are possible. The presented workflow is illustrated in Figure 1. In the course of the workflow an EAST-ADL model is annotated, systematically enhanced and refined using a modeling framework. The elaborated model integrates the work products (e.g. analysis results, requirements, system architecture) of the workflow phases. EAST-ADL is a domain-specific language and tailored to the needs of the automotive domain. It is *diagrammatic* [5] such as UML. It consists of syntactic elements such as boxes, ovals, lines or arrows. Its *abstract syntax* is defined by its meta model and its *semantic domain* and *semantic mapping* are defined using natural language [5]. The workflow phases are thereafter described:

1. **Definition of the Analysis Subject:** First information about the vehicle under development is collected and modeled. Functions of the vehicle (e.g. motoring or recuperative braking) are defined. Requirements to these functions are determined and allocated (e.g. conditions for activation or deactivation). In addition relevant modes (e.g. drive, creep or acceleration) are identified for each function and associated with the requirements.
2. **Identification of Hazards and Hazardous Events:** Based on the definition of the analysis subject, PHA (for more details see also [9]) is carried out. Possible malfunctions are identified. Hazards are derived for each malfunction (e.g. unintended acceleration of the vehicle). Thereafter operational situations such as traffic situations (e.g. oncoming traffic on a highway in a curve) and maintenance situations (e.g. vehicle at lifting ramp) are defined. Moreover use cases describing the behavior (e.g. overtaking or

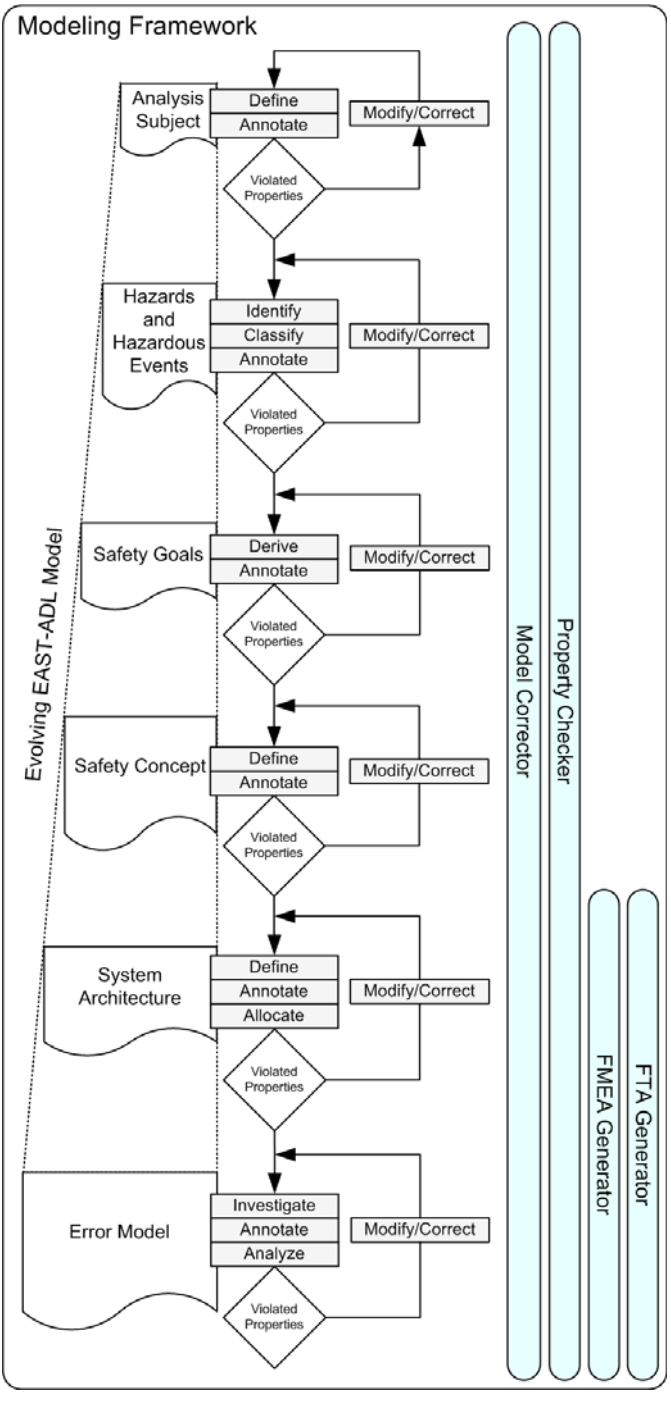

Fig. 1. Computer-Aided Safety Engineering Workflow

changing oil) of the related actors (e.g. driver or mechanic) are described. Hazardous events are determined for relevant combinations of hazards, use cases and operational situations. Moreover relevant modes are identified for each hazardous event. The criticality of each hazardous event is assessed in terms of its controllability, severity and exposure and an ASIL (Automotive Safety Integrity Level) [7] is determined.

3. **Derivation of Safety Goals:** For each hazardous event that has an ASIL assigned (ASIL A, ASIL B, ASIL C or ASIL D), a safety goal is derived and associated. Furthermore, a safe state is defined (e.g. switch open) for each safety goal. Alternatively a safe mode (e.g. limp home mode) is determined. The determined safety goals are top-level safety requirements.
4. **Definition of Safety Concept:** The safety concept is derived from the safety goals. This safety concept consists of functional and technical safety requirements to the automotive embedded system, connected sensors and controlled actuators. Traces are created between safety goals, functional safety requirements and technical safety requirements.
5. **Definition of System Architecture:** The system architecture is defined in terms of the embedded system, connected sensors and controlled actuators. Moreover the parts of the environment are modeled that interact with the sensors and actuators. Thereafter the functional and technical safety requirements are allocated to the components of the system architecture. Furthermore functions are allocated to the components of the system architecture.
6. **Investigation and Annotation of Faults and Failures:** Information flows and energy flows through the embedded system, connected sensors, controlled actuators and their environment are investigated. Possible faults and failures are estimated and their propagation is analyzed and annotated. Moreover it is investigated and annotated how the failures lead to the malfunctions that were identified during PHA. Thereafter FTA and FMEA are applied.

After the completion of these working steps, requirements to software and hardware are derived from the safety concept. Software and hardware are fully specified, implemented, integrated, verified and validated. However these working steps are beyond the scope of this work.

Due to the complexity of contemporary vehicles, the application of the workflow is cumbersome and error-prone. Therefore we propose to aid the safety engineering workflow defined above by automated property checking (property checker), automated model correction (model corrector), automated fault tree synthesis and automated FMEA table synthesis (see Section 4 and Section 5). This allows to early identify erroneously applied working steps and enables the automated suggestion and application of corrective measures. Moreover it is not necessary to construct fault trees and FMEA tables manually. Instead, they are consistently generated from the EAST-ADL model. While property checker and model corrector aid the entire workflow, FTA generator and FMEA generator are especially useful for the verification of the system architecture defining an

embedded system together with connected sensors and controlled actuators. The automated analysis and synthesis capabilities of the modeling framework provide guidance and strongly support the application of the workflow and the creation of a complete and consistent set of work products.

4 Computer-Aided Checking

Properties are defined that indicate the correct application of the activities of the safety engineering workflow (see Section 3). A property checker is part of the modeling framework (see Figure 1) and continuously checks the evolving EAST-ADL model and presents violating modeling elements to the user. If no properties are violated, the EAST-ADL model indicates the correct application of the workflow. If the property checker identifies violated properties, the erroneous application of the workflow is unveiled. The property checker does not only allow the early identification of errors, it is also a valuable guide, while the workflow is applied. In addition to properties for the earlier phases of the safety engineering workflow (see [9]), properties for the later phases are presented in Table 1.

Assume M is an EAST-ADL model, M_{MM} is the EAST-ADL meta model and P is the set of properties an EAST-ADL model is expected to hold. Assume e is a modeling element of the EAST-ADL model, t is a type defined by the EAST-ADL meta model and p is a property (Expression 1).

$$e \epsilon M, t \epsilon M_{MM}, p \epsilon P \tag{1}$$

Moreover, $I(e,t)$ pertains, if e is of type t, $D(t,p)$ pertains, if p is defined for t and $H(e,p)$ pertains if p holds for e. If M *indicates* the correct application of the workflow, Expression 2 is valid. In this case no modeling elements violate properties.

$$\neg\exists e \neg\exists t \neg\exists p(I(e,t) \wedge D(t,p) \wedge \neg H(e,p)) \tag{2}$$

If a model M *shows the erroneous application* of the workflow, Expression 3 is valid. In this case at least one modeling element violates a property.

$$\exists e \exists t \exists p(I(e,t) \wedge D(t,p) \wedge \neg H(e,p)) \tag{3}$$

5 Automated Synthesis

5.1 Model Correction

Correction rules are defined that impose possible solutions to problems indicated by the property checker (see also Section 4). A model corrector is part of the modeling framework (see Figure 1). Possible solutions are suggested on demand by the model corrector, based on the evolving EAST-ADL model and the correction rules. A user can decide to accept a suggestion of the model corrector

Table 1. Properties of the EAST-ADL model are automatically checked

ID	Meta Class	Property Definition
28	SafetyGoal	At least one safety requirement is derived
29	QualityRequirement	Traceable to a safety requirement or a SafetyGoal, if it is a safety requirement
30	QualityRequirement	Is allocated to at least one AnalysisFunctionPrototype, if it is a safety requirement
32	Environment	An environmentModel is defined
34	AnalysisLevel	A functionalAnalysisArchitecture has been defined
39	FunctionFlowPort	Has at most one FunctionConnector to a FunctionFlowPort of type out or inout associated, if type in
40	FunctionPort	A type is defined
40a	FunctionPort	Connected by at least one FunctionConnector
40c	FunctionPort	A complementary description has been defined
40b	FunctionConnector	Connector is connected to two FunctionPorts
41	AnalysisFunctionPrototype	A type is defined
42	AnalysisFunctionPrototype	Has a complementary description
42a	AnalysisFunctionPrototype	An ErrorModelPrototype is defined for every AnalysisFunctionPrototype
37	AnalysisFunctionType	At least one FunctionPort has been defined
42b	AnalysisFunctionType	An ErrorModelType is defined for every AnalysisFunctionType
48	FaultInPort	Has only one FaultFailurePropagationLink to a FailureOutPort associated
51	FaultFailurePort	A functionTarget_path is defined
51a	FaultFailurePort	A type is defined
52a	FailureOutPort	Has a complementary description
53	ErrorModelPrototype	A type is defined
54	ErrorModelPrototype	A functionTarget is defined
55	ErrorBehavior	An externalFailure is defined
56	ErrorBehavior	The defined failureLogic is legal and recognized
57	ErrorBehavior	An owner is defined
58	InternalFaultPrototype	Has a complementary description
59	InternalFaultPrototype	Is owned by at least one ErrorBehavior
60	VehicleFeature	Every function is allocated to at least one AnalysisFunctionPrototype
62	FeatureFlaw	Is mapped onto a FailureOutPort
63	EABoolean	A note is defined
64	RangeableDatatype	A note is defined
65	EAFloat	The lower threshold is defined
66	EAFloat	The upper threshold is defined

Table 2. Possible solutions to problems that are automatically suggested

ID	Meta Class	Suggested Solution
28	SafetyGoal	Creation and association of safety requirement
28	SafetyGoal	Associate one of the untraceable safety requirements
29	QualityRequirement	Associate to existing SafetyGoal, if it is a safety requirement
29	QualityRequirement	Associate to existing safety requirement, if it is a safety requirement
30	QualityRequirement	Allocation to existing AnalysisFunctionPrototype, if it is a safety requirement
32	Environment	Creation and association of AnalysisFunctionPrototype
34	AnalysisLevel	Creation and association of AnalysisFunctionPrototype
40	FunctionPort	Association of existing EAInteger
40	FunctionPort	Association of existing EAFloat
40	FunctionPort	Association of existing EABoolean
40c	FunctionPort	Creation and association of Comment
40b	FunctionConnector	Remove connector
41	AnalysisFunctionPrototype	Association of existing AnalysisFunctionType
42	AnalysisFunctionPrototype	Creation and association of Comment
42a	AnalysisFunctionPrototype	Association of existing ErrorModelPrototype
37	AnalysisFunctionType	Creation and association of FunctionPort
42b	AnalysisFunctionType	Creation and association of ErrorModelType
42b	AnalysisFunctionType	Association of existing, unassociated ErrorModelType
51	FaultFailurePort	Association of existing AnalysisFunctionPrototype
51a	FaultFailurePort	Association of existing EAInteger
51a	FaultFailurePort	Association of existing EAFloat
51a	FaultFailurePort	Association of existing EABoolean
52a	FailureOutPort	Creation and association of Comment
53	ErrorModelPrototype	Creation and association of ErrorModelType
53	ErrorModelPrototype	Association of existing ErrorModelType
54	ErrorModelPrototype	Association of existing AnalysisFunctionPrototype
55	ErrorBehavior	Association of existing, unassociated FailureOutPort
56	ErrorBehavior	Change to and (type OTHER)
56	ErrorBehavior	Change to or (type OTHER)
57	ErrorBehavior	Creation and association of ErrorModelType
57	ErrorBehavior	Association of existing ErrorModelType without ErrorBehavior
58	InternalFaultPrototype	Creation and association of Comment
59	InternalFaultPrototype	Association of existing ErrorBehavior
60	VehicleFeature	Allocation to existing AnalysisFunctionPrototype
62	FeatureFlaw	Allocation to existing FailureOutPort

or to solve the problem in another way. If a suggested, possible solution is accepted, the EAST-ADL model is automatically modified and corrected making a manual modification superfluous. If a suggestion is rejected the EAST-ADL model remains unchanged. In addition to correction rules for the earlier phases of the safety engineering workflow (see [9]), correction rules for the later phases are presented in Table 2.

Assume M is an EAST-ADL model, M_{MM} is the EAST-ADL meta model, P is the defined set of properties and S is the set of defined suggestions. Assume e_1 is a modeling element of the EAST-ADL model, t_1 is a type defined by the meta model, p_1 is a property and s_1 is a suggestion (Expression 4).

$$e_1 \epsilon M, t_1 \epsilon M_{MM}, p_1 \epsilon P, s_1 \epsilon S \tag{4}$$

Assume that before an automated model correction is carried out (precondition), e_1 is of type t_1 and violates p_1 that is defined for t_1 (Expression 5).

$$I(e_1, t_1) \wedge D(t_1, p_1) \wedge \neg H(e_1, p_1) \tag{5}$$

If the user accepts suggestion s_1, the EAST-ADL model M is automatically corrected and transformed to EAST-ADL model M' by function γ depending on M, e_1, t_1, p_1 and s_1 (Expression 6).

$$\gamma(M, e_1, t_1, p_1, s_1) \rightarrow M' \tag{6}$$

After the modification (postcondition) e_1 is an element of M', e_1 is still of type t_1 and does not violate p_1 any more (Expression 7).

$$e_1 \epsilon M', I(e_1, t_1) \wedge D(t_1, p_1) \wedge H(e_1, p_1) \tag{7}$$

5.2 Fault Tree and FMEA Table Synthesis

The modeling framework (see Figure 1) contains a FTA generator and a FMEA generator. The EAST-ADL model that is created in the course of the safety engineering workflow (see Section 3) is input to them. The FTA generator is able to synthesize fault trees (see Figure 2). The fault trees show, how each safety goal can be violated by the faults and failures of the embedded system, connected sensors or controlled actuators. The FMEA generator is able to synthesize a FMEA table (see Figure 3). The FMEA table shows failure modes of the components, causative faults for these failure modes and effects of these failure modes in terms of violated safety goals.

The FTA generator considers the recommendations of IEC 61025 [6]. Therefore the shapes of the symbols of the fault trees are adapted to the shapes of the symbols recommended by IEC 61025. Basic events (faults, failures) are represented by circles, complex events (faults, failures, malfunctions, hazards, hazardous events, violated safety goals) are represented by rectangles and gates (and, or) are represented by the corresponding logic symbols. The FTA generator uses the identified safety goals as top events of the generated fault trees (one

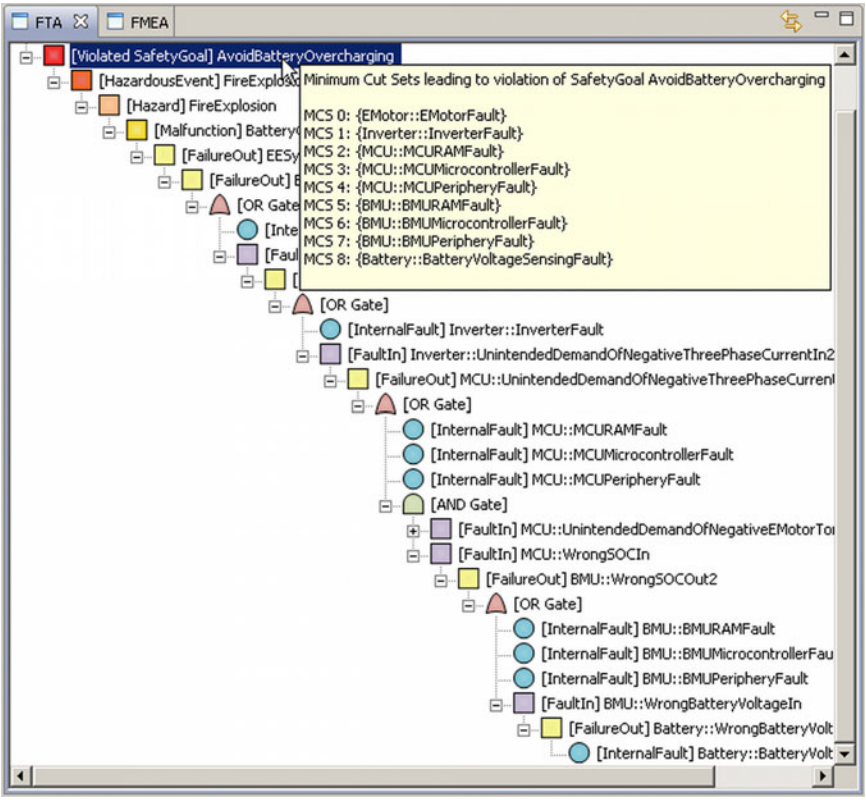

Fig. 2. Fault trees can be synthesized from the EAST-ADL model

fault tree per safety goal is generated) and adds the causative malfunctions, hazards and hazardous events that were identified during PHA as offsprings. Component faults, component failures and gates that were identified during the definition of the error model are used as offsprings of the malfunctions. Thus the generated fault trees are consistent to the earlier elaborated PHA results.

It is possible to automatically extract the minimum cut sets from each fault tree. A *minimum cut set* [12] is a set of basic events leading to the top event (violation of the safety goal) that cannot be reduced in number. For every violated safety goal the minimum cut sets can be displayed on demand (see Figure 2).

Assume M is an EAST-ADL model and S is the subset of M that contains hazards, hazardous events, safety goals, system architecture and the error model (see Expression 8).

$$S \subseteq M \tag{8}$$

Given that Expression 2 holds for all $e \epsilon S$, FTA generator $\rho(S)$ can generate graphical fault trees Υ that allow examining how component faults and failures can contribute to the violation of safety goals (see Expression 9).

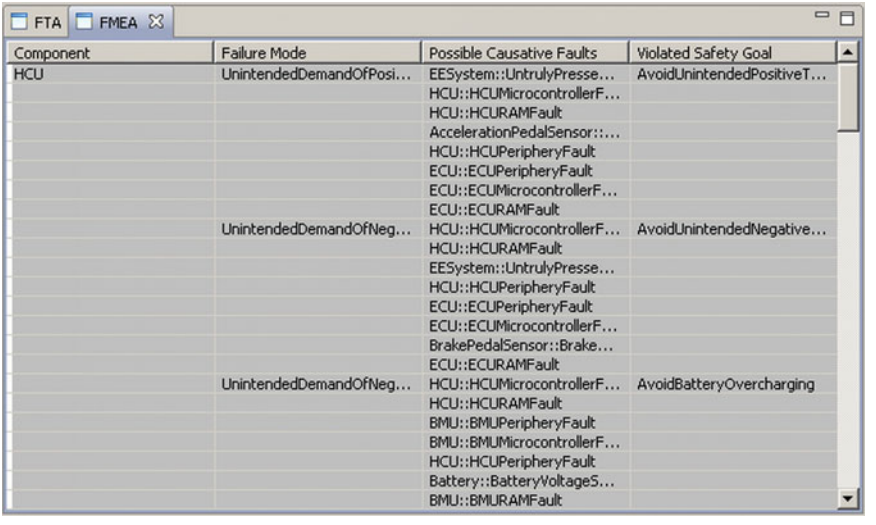

Fig. 3. A FMEA table can be synthesized from the EAST-ADL model

$$\rho(S) \rightarrow \Upsilon \tag{9}$$

The FMEA generator creates a FMEA table containing four columns denoting the names of the components (*Component*), the component failure modes leading to the violation of safety goals (*Failure Mode*), faults that potentially cause the component failure modes (*Possible Causative Faults*) and the violated safety goals (*Violated Safety Goal*). The generated FMEA table is consistent to the fault trees and the earlier elaborated PHA results, because FTA generator and FMEA generator use the same model S as input.

Given that Expression 2 holds for all $e \epsilon S$, FMEA generator $\alpha(S)$ can generate a graphical FMEA table Ξ that allows to examine how component failures can lead to the violation of safety goals (see Expression 10).

$$\alpha(S) \rightarrow \Xi \tag{10}$$

6 Experimental Evaluation

A plugin for the open source tool Papyrus [8] was created that allows property checking, model correction, fault tree generation and FMEA table generation such as described in Section 4 and Section 5. Thereafter the approach was experimentally evaluated using the case study of a hybrid electric vehicle development. This type of vehicle contains an additional electric motor that supplements the internal combustion engine providing substitutive or additive torque. This electric motor is controlled by the automotive embedded system. The safety engineering workflow such as defined in Section 3 was carried out for a part of a

hybrid electric vehicle powertrain being aided by the property checker and the model corrector.

Although the safety engineering workflow was carried out only for a part of the hybrid electric vehicle powertrain, the resulting EAST-ADL model contains 457 interconnected modeling elements. Each of them contains numerous attributes. The property checker identifying erroneously applied activities (see Section 4) and the model corrector suggesting and applying model corrections (see Section 5.1) strongly supported the application of the workflow and allowed coping with the complexity. Illustrations of property checker and model corrector can be found in [9].

During PHA the hybrid electric vehicle was identified to be safety-critical, because its failures can cause malfunctions such as battery overcharging (*BatteryOvercharging*). This malfunction can lead to hazards such as fire or explosion of the battery (*FireExplosion*). Fire or explosion of the battery during vehicle operation imposes a hazardous event (*FireExplosionDuringCityTraffic*). Therefore the safety goal *AvoidBatteryOvercharging* was defined to control or mitigate the corresponding hazard.

In later phases of the workflow, a part of the system architecture including networked ECUs (electronic control unit), connected sensors and controlled actuators was defined. Furthermore the relevant parts of the interacting environment were modeled. The propagation of faults and failures was estimated and annotated. The failure *UnintendedNegativeTorque2* of the component *EMotor* was identified to be causative for the malfunction *BatteryOvercharging*. This failure can occur due to a failure of the E-motor or faults propagated from a sensor and networked ECUs such as the BMU (Battery Management Unit).

Fault trees and a FMEA table were synthesized from the annotated model (see Section 5.2). Figure 2 depicts a fault tree that shows the relations between safety goal *AvoidBatteryOvercharging*, hazardous event *FireExplosionDuringCityTraffic*, hazard *FireExplosion*, malfunction *BatteryOvercharging* as well as the causative faults and failures of the components. The extracted minimum cut sets that can cause the violation of the safety goal are also depicted.

Figure 3 shows a part of the synthesized FMEA table. The table shows that a failure mode of the HCU (Hybrid Control Unit) can lead to the violation of the safety goal *AvoidBatteryOvercharging*. Moreover possible causative faults are listed.

7 Conclusion

This work presents a modeling framework with analysis and synthesis capabilities. This modeling framework supports a safety engineering workflow. In the course of the workflow a model is annotated using the domain-specific language EAST-ADL. This model integrates the work products of the workflow phases. The modeling framework contains a property checker that allows to unveil the incorrect application of the workflow and a model corrector that suggests and automatically performs corrections of the evolving model. Moreover fault trees and

a FMEA table can be automatically synthesized allowing the application of qualitative FTA and FMEA. This tightly integrated approach ensures consistency of PHA results, fault trees and FMEA table. The approach was evaluated using the case study of a hybrid electric vehicle development. While the analysis and synthesis capabilities of the modeling framework did not replace the intellectual process of applying the workflow, they strongly supported its application.

Acknowledgment. The authors wish to thank the "COMET K2 Forschungsförderungs-Programm" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economics and Labour (BMWA), Österreichische Forschungsförderungsgesellschaft mbH (FFG), Das Land Steiermark and Steirische Wirtschaftsförderung (SFG) for their financial support. Additionally we would like to thank the supporting company and project partner AVL List GmbH as well as Graz University of Technology. Further information about the MEPAS project can be found at http://www.v2c2.at/mepas.

References

1. ATESST2 Project Consortium: EAST-ADL Domain Model Specification, version 2.1, Release Candidate 3 (2010)
2. Biehl, M., DeJui, C., Törngren, M.: Integrating Safety Analysis into the Model-based Development Toolchain of Automotive Embedded Systems. In: Proc. of the Conference on Languages, Compilers and Tools for Embedded Systems, pp. 125–131 (2010)
3. Domis, D., Trapp, M.: Integrating Safety Analyses and Component-Based Design. In: Proc. of the 27th International Conference on Computer Safety, Reliability and Security, pp. 58–71 (September 2008)
4. Elmqvist, J., Nadjm-Tehrani, S.: Tool Support for Incremental Failure Mode and Effects Analysis of Component-Based Systems. In: Proc. of the Design, Automation and Test in Europe Conference and Exhibition (DATE 2008), pp. 921–927 (April 2008)
5. Harel, D., Rumpe, B.: Meaningful Modeling: What's the Semantics of "Semantics"? IEEE Transactions on Computers 37, 64–72 (2004)
6. International Electrotechnical Commission: IEC 61025 - Ed. 2.0 Fault tree analysis (FTA) (2006)
7. International Organization for Standardization: ISO/DIS 26262-3 Road vehicles - Functional safety - Part 3: Concept phase (2009)
8. Lanusse, A., Tanguy, Y., Espinoza, H., Mraidha, C., Gerard, S., Tessier, P., Schnekenburger, R., Dubois, H., Terrier, F.: Papyrus UML: an open source toolset for MDA. In: Proc. of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA 2009), pp. 1–4 (June 2009)
9. Mader, R., Grießnig, G., Leitner, A., Kreiner, C., Bourrouilh, Q., Armengaud, E., Steger, C., Weiß, R.: A Computer-Aided Approach to Preliminary Hazard Analysis for Automotive Embedded Systems. In: Proc. of the IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS), pp. 169–178 (2011)

10. Majdara, A., Wakabayashi, T.: A New Approach for Computer-Aided Fault Tree Generation. In: Proc. of the 3rd Annual IEEE Systems Conference, pp. 308–312 (2009)
11. de Miguel, M., Briones, J., Silva, J., Alonso, A.: Integration of safety analysis in model-driven software development. IET Software 2, 260–280 (2008)
12. Leveson, N.G.: Safeware: system safety and computers. Addison-Wesley Publishing Company, Reading (1995)
13. Papadopoulos, Y., Grante, C.: Evolving car designs using model-based automated safety analysis and optimisation techniques. The Journal of Systems and Software 76, 77–89 (2004)
14. Papadopoulos, Y., Maruhn, M.: Model-Based Synthesis of Fault Trees from Matlab - Simulink models. In: Proc. of the International Conference on Dependable Systems and Networks (DSN 2001), pp. 77–82 (July 2001)
15. Sandberg, A., Chen, D.J., Lönn, H., Johansson, R., Feng, L., Törngren, M., Torchiaro, S., Kolagari, R.T., Abele, A.: Model-Based Safety Engineering of Interdependent Functions in Automotive Vehicles Using EAST-ADL2. In: Proc. of the 29th International Conference on Computer Safety, Reliability and Security, pp. 332–346 (September 2010)