ISF

Technische Universität Braunschweig

Applying Higher-Order Delta Modeling for the Evolution of Delta-Oriented Software Product Lines

Informatik-Bericht Nr. 2018 – 01

Sophia Nahrendorf, Sascha Lity, and Ina Schaefer

June 5, 2019

Acknowledgment

The author Sascha Lity is supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution.

Changelog

Version	Date	Author	Change Description
1.0	12.02.2018	Sascha Lity	Version 1.0 finalized
1.1	05.06.2019	Sascha Lity	Typos and failure in categorization fixed

Abstract

In this technical report, we define and document the evolution of four delta-oriented software product lines based on the application of higher-order delta modeling. Higher-order delta modeling is a variability modeling technique for capturing the variability and evolution of a variant-rich system by the same means, where differences between variants and versions of variants are specified in terms of (higher-order) deltas. For every product line, we describe the complete evolution history including the original version and the higher-order deltas specifying the respective evolution steps. As underlying modeling formalism, we use finite state machines.

Keywords Delta-Oriented Software Product Lines, Software Evolution, Variability Modeling, Higher-Order Delta Modeling

Contents

List of Figures

List of Abbreviations

AL	Automatic Locking
AS	Alarm System
AutoPW	Automatic Power Window
BCS	Body Comfort System
CAP	Control Automatic Power Window
CAS	Control Alarm System
CLS	Central Locking System
DRL	Daytime Running Lights
EM	Exterior Mirror
FM	Feature Model
FP	Finger Protection
HB	High Beam
HMI	Human Machine Interface
HW	Heatable Windows
IM	Interior Monitoring
LB	Low Beam
LED	Status LED
LED AS	LED Alarm System
LED CLS	LED Central Locking System
LED EM	LED Exterior Mirror
LED FP	LED Finger Protection
LED Heat	LED Heatable
LED PW	LED Power Window
ManPW	Manual Power Window
PL	Parking Lights
PW	Power Window
RCK	Remote Control Key
SF	Safety Function
SM	State Machine
SPL	Software Product Line
WT	Window Temperature

1 Introduction

Software systems are subject to constant evolution due to changing demands of customers, bugfixes or to improve functionality [14]. Evolution is controlled by strict guidelines of software engineering, in order to efficiently and correctly realize it. Essential parts of the evolution process are the documentation and modeling of the system. Models provide an overview of the system as well as support an early identification of errors or inconsistencies, and ensure a better and more clear communication and understanding between the stakeholders involved in the development process [1].

Software product lines [17, 4] (SPLs), i.e., a family of similar product variants with explicit specification of the commonality and variability, are especially often subjected to evolution because of their diversity of variants [14]. There exist different established approaches for modeling SPLs [14, 19] as for instance the transformational approach delta modeling [2, 18]. Delta modeling represents a family of products based on a designated core product and a set of deltas [7]. Deltas describe sequences of changes, called operations, to the core product. These operations are additions, removals, or modifications of elements allowing for the creation of further product variants by transforming the core. In contrast, the modeling of SPL evolution is more challenging, but there also exist approaches in the literature [11, 13, 14]. For instance higher-order delta modeling [11] which is an extension of delta modeling to capture the evolution and variability of an SPL by the same means. The approach lifts the concept of delta modeling [2] to the next level in order to change complete delta models instead of core models. By specifying and applying so-called higher-order deltas to delta models, new deltas can be added to the delta model, existing but obsolete deltas can be removed from the delta model, or deltas can be modified. Modifications of deltas change the sequence of operations specified in the delta by adding, removing, or modifying operations. However, variability modeling techniques and other SPL approaches have to be evaluated in order to fix mistakes, show their correct function, or demonstrate suitability for daily use. In order to evaluate new approaches for modeling evolution in software product lines, a sufficient amount of systems offering an evolution history have to be available. So far, there are rather few model-based systems with evolution histories in the context of evolving software product lines.

Therefore, in this technical report we propose and document the evolution of four different delta-oriented product lines. The evolution scenarios are based on the project thesis of Nahrendorf [15]. For each of the four systems, several evolution scenarios are developed. Those evolution scenarios cover the different operations of adding, removing, and modifying deltas. The systems to be examined are a body comfort system of a vehicle [12], a wiper system, a vending machine, and a mine pump system [3]. The evolution scenarios for the different systems will be described, and higher-order delta modeling [11] is used to model the different scenarios. The evolution scenarios to be developed facilitate the evaluation of further approaches in the context of evolution of software product lines in the future.

Structure of the Report

The remainder of the report is organized as follows. The evolving product lines of the body comfort system, the vending machine, the wiper system, and the mine pump system are introduced in Chap. 2. For each system, the original version for the development of evolution scenarios is established by describing the structure and behavior of the systems. In Chap. 3, the evolution scenarios for the evolving SPLS are documented. To this end, the desired changes for the scenarios are described and graphically represented using higher-order delta modeling. The technical report is concluded in Chap. 4.

2 Delta-Oriented SPLs

In this chapter, the four different software product lines documented in this technical report and their original behavioral specifications are introduced. Therefore, we describe the original feature models as well as the corresponding delta models building the basis for the definition of evolution steps as proposed in Sect. 3. Each delta model consists of a core model and different deltas, which generate valid variants if applied to the core model. The four systems encompass a vending machine, a wiper system, and a mine pump system [3] as well as a body comfort system [12].

For deltas, we use the following graphical notation. Deltas are illustrated as box comprising its name, its application condition, and the specification of the captured change operations. For the change operations, a $+$ denotes an addition of an element, whereas a $-$ represents a removal and a $*$ a modification of an element. Elements that are illustrated with dashed lines represent references to elements which are required to understand the context of a change operation, e.g., the addition of a transition between a certain source and target state.

2.1 Vending Machine

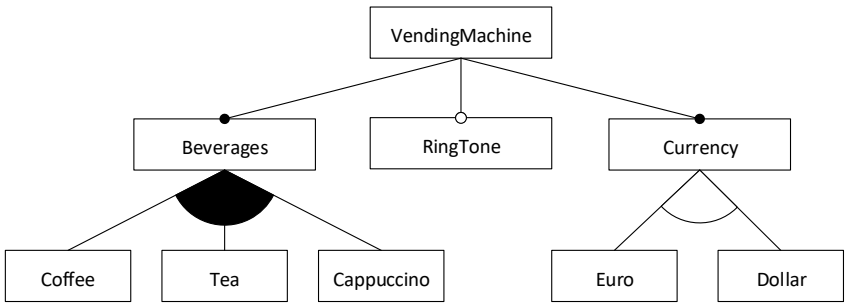

Figure 2.1: Feature Model of the Vending Machine SPL (according to [3])

The software product line of the vending machine [3] describes a machine selling hot beverages. The system was initially introduced by Fantechi and Gnesi [5] with two different variants and was extended by Classen [3]. The product line of the vending machine offers three different beverages (`Coffee`, `Tea` and `Cappuccino`) and accepts two different currencies (`Euro` and `Dollar`). This relation is depicted in the feature model shown in Fig. 2.1. For a valid product, at least one beverage has to be selected, where the beverages can be arbitrarily combined. Furthermore, exactly one currency has to be chosen. Finally, there exists the optional feature `RingTone` which represents a function emitting a tone as soon as a beverage has been delivered.

Behavioral Specification

The core state machine representing the behavior for the selected features `Coffee` and `Euro` is shown in Fig. 2.2. The vending machine is activated by inserting a coin. Afterwards, the custo-

mer has to decide if he wants sugar for his beverage or not. After this decision, the favored beverage has to be selected. If sugar was chosen, at first, the sugar will be poured into the cup, and after that, the chosen beverage. If sugar was not chosen, the beverage will directly be poured into the cup. After delivering the beverage, an event shows up on the display saying that the beverage is ready to be taken, and a tone is emitted. In case the vending machine variant does not have the feature `RingTone`, the tone emission is skipped. As soon as the cup is taken, the machine returns to its initial state and waits for the next inserted coin.

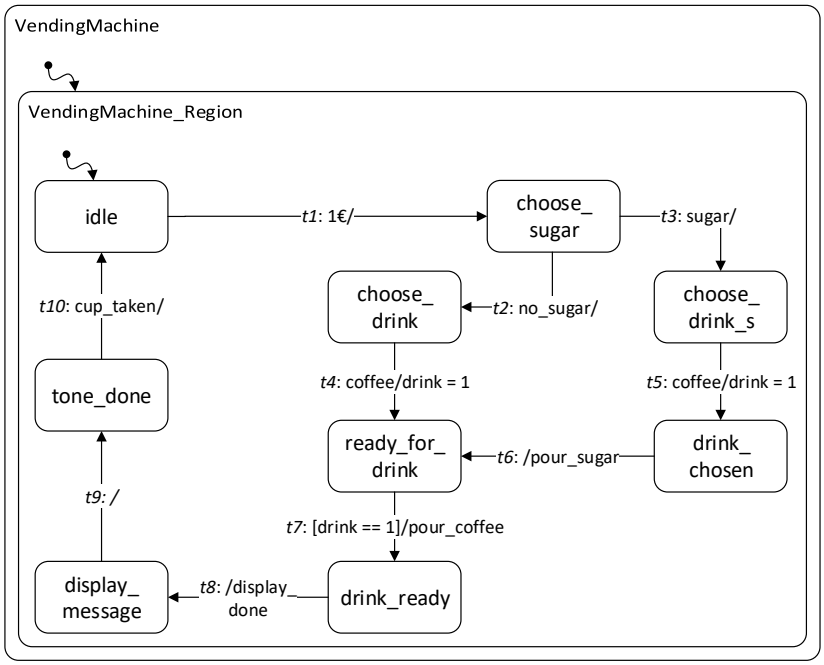

Figure 2.2: Core State Machine of the Vending Machine SPL

In Fig. 2.3 and Fig. 2.4, the deltas are depicted which are specified to transform the core into variant-specific state machines. To generate a variant emitting a ringtone when a beverage was delivered, the delta `Delta_Ringtone` is used. This delta replaces transition `t9` for skipping the tone emission with transition `t11` for emitting the tone. In case `Dollar` is selected as currency instead of `Euro`, `Delta_Dollar` removes transition `t1` between the states `idle` and `wait_for_sugar` for detecting an euro coin and adds transition `t12` for detecting dollar in its place.

`Delta_Tea` describes the changes needed to be made to the core model in order to generate a machine additionally offering tea as beverage. The transitions `t13` and `t14` for choosing tea as well as the transition `t15` for pouring tea are added to the model. In case `Coffee` is not selected as a beverage for the machine, the related elements can be removed by `Delta_Remove_Coffee`. Thus, this delta removes the transitions `t4`, `t5`, and `t7`. `Delta_Cappuccino` adds cappuccino to the beverage choice by adding the state `ca_ready_for_milk` as well as the transitions `t17`, `t18`, `t19`, and `t20` for choosing cappuccino and pouring coffee and milk into the cup.

To this end, the vending machine product line consists of 28 valid feature configuration and, accordingly, 28 different variants.

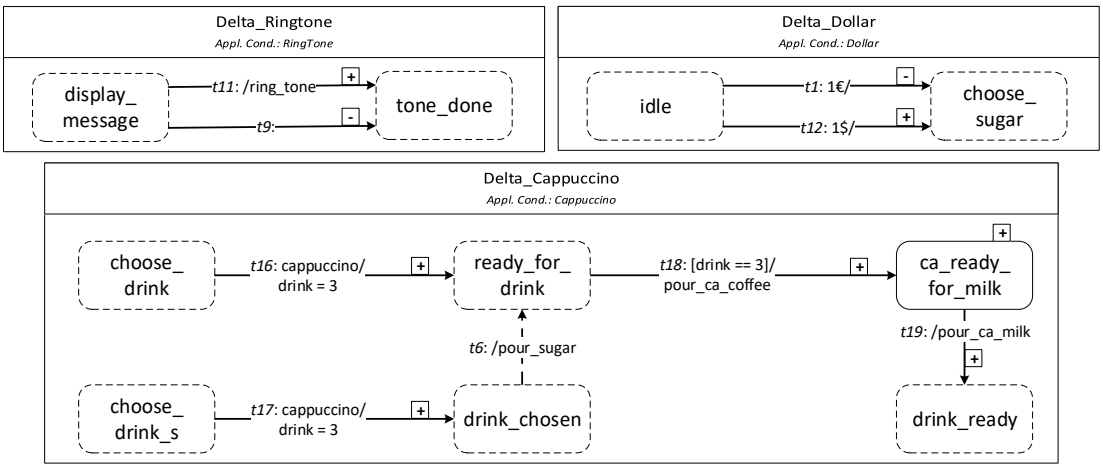

Figure 2.3: First Part of the Deltas of the Original Vending Machine SPL

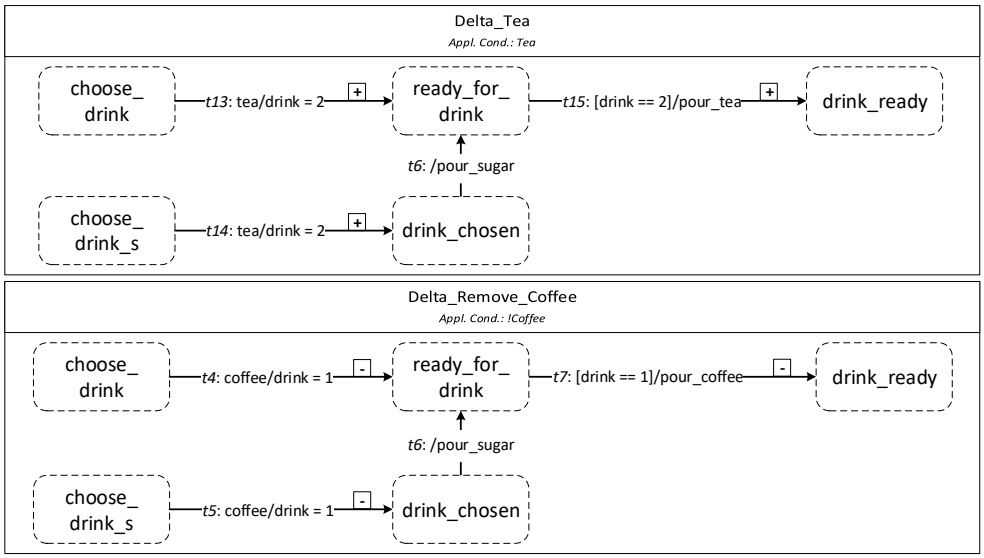

Figure 2.4: Second Part of the Deltas of the Original Vending Machine SPL

2.2 Wiper System

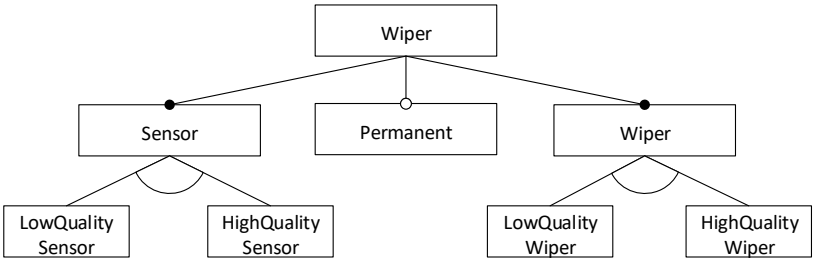

Figure 2.5: Feature Model of the Wiper System (according to [3])

The wiper system [3] describes simple rain-sensing wipers. It was initially introduced by Gruler et al. [6] and later adapted by Classen [3]. As shown in Fig. 2.5, the wiper system consists of a `Sensor` and a `Wiper`. The sensor as well as the wiper are available in the two quality levels, namely `High` and `Low`. Additionally, there is the optional feature `Permanent` allowing continuous wiping.

Behavioral Specification

The low quality sensor merely detects the difference between rain and no rain. The low quality wiper automatically reacts to rain with a slow wiping. The high quality sensor detects no rain as well as the difference between little rain and heavy rain. According to the rain intensity, the high quality wiper automatically activates a slow or fast wiping. In case the wiper system has the optional feature `Permanent` selected, a permanent wiping can be manually activated instead of the automatic wiping.

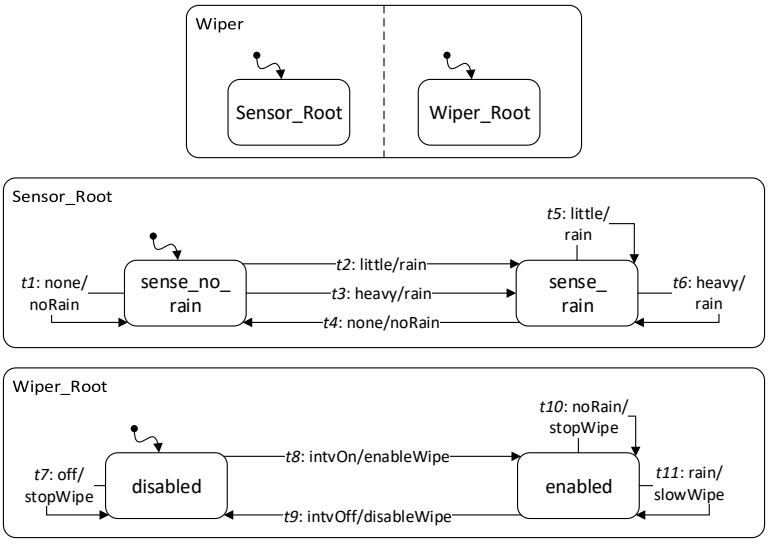

Figure 2.6: Core State Machine of the Wiper System SPL

The core model of the wiper system consists of the low quality wiper and the low quality sensor. The core is depicted in Fig. 2.6 and contains the two concurrent state machines `Wiper_Root` and

Sensor_Root representing the behavior of the wiper and the sensor, respectively. In case the sensor detects no rain, its state machine passes into the state sense_no_rain and sends the event noRain to the wiper. Both for little and for heavy rain, the low quality sensor sends the event rain and goes into the state sense_rain. As soon as the wiper is switched on, its state machine passes into the state enabled. In this state, the sensor activates a slow wiping for the event rain and stops the wiping again for the event noRain. In the state disabled the wiper is turned off and does not react to any events.

Figure 2.7 shows the deltas which are specified to derive further valid variant-specific models from the core state machine. Delta_HighSensor makes the necessary changes to transform the low quality sensor into the high quality sensor. For this purpose, the state sense_heavy_rain is added to Sensor_Root. The sensor passes into this state via the new transitions t13 or t15 whenever it detects heavy rain. Based on the new transition t14, the sensor stays in the same state if it is already in state sense_heavy_rain and detects heavy rain. All three transitions trigger the event heavyRain. Additionally, in case the rain stops when the system is in state sense_heavy_rain, the sensor passes into the state sense_no_rain via the added transition t12, sending the event noRain in the process. In case the rain only lower, the sensor transfers into state sense_rain via transition t16 sending the event rain. Furthermore, the obsolete transitions t3 and t6 are removed, because they would otherwise still send the event rain when dectecting heavy rain.

Delta_HighSensor_LowWiper adapts the low quality wiper to the high quality sensor by adding transition t17. Due to this transition, the enabled wiper reacts with slow wiping to the event heavyRain. Delta_HighSensor_LowWiper can only be applied after Delta_HighSensor.

Delta_HighSensor_HighWiper describes the necessary changes in order to transform the low quality wiper into the high quality wiper, if the low quality sensor was transformed into the high quality sensor by Delta_HighSensor before. For this purpose, the transition t18 is added to the model. This transition activates the fast wiping for the event heavyRain, if the wiper is enabled.

If a combination of low quality sensor and high quality wiper is desired, Delta_LowSensor_HighWiper is used. This delta removes transition t11 and replaces it with transition t19 which activates a fast instead of a slow wipe for the event rain.

For all four combinations of sensor and wiper, Delta_PermanentWiping adds a permanent wiping function. For this purpose, the state permanent is added to Wiper_Root. In case the permanent wiping is activated, the wiper can be switched off completely by the new transition t20. If the wiper is disabled, the permanent wiping can be activated again by the added transition t21. Furthermore, the new transition t22 switches from permanent wiping to the automatic sensor-based wiping.

In total, the wiper system has eight valid feature configurations and, consequently, eight different product variants.

2.3 Mine Pump System

The mine pump system [3] serves to pump water out off a mine shaft. It was already introduced in 1983 by Kramer et al. [9] and later extended by Classen [3]. The product line of the mine pump system comprises the mandatory feature WaterRegulation as well as the optional features Command and MethaneDetection shown in the feature model in Fig. 2.8. The feature WaterRegulation provides a sensor monitoring the water level and a pump for pumping out water. The handling of the water level can be varied according to the child features Low, Normal and High. The handling of a high

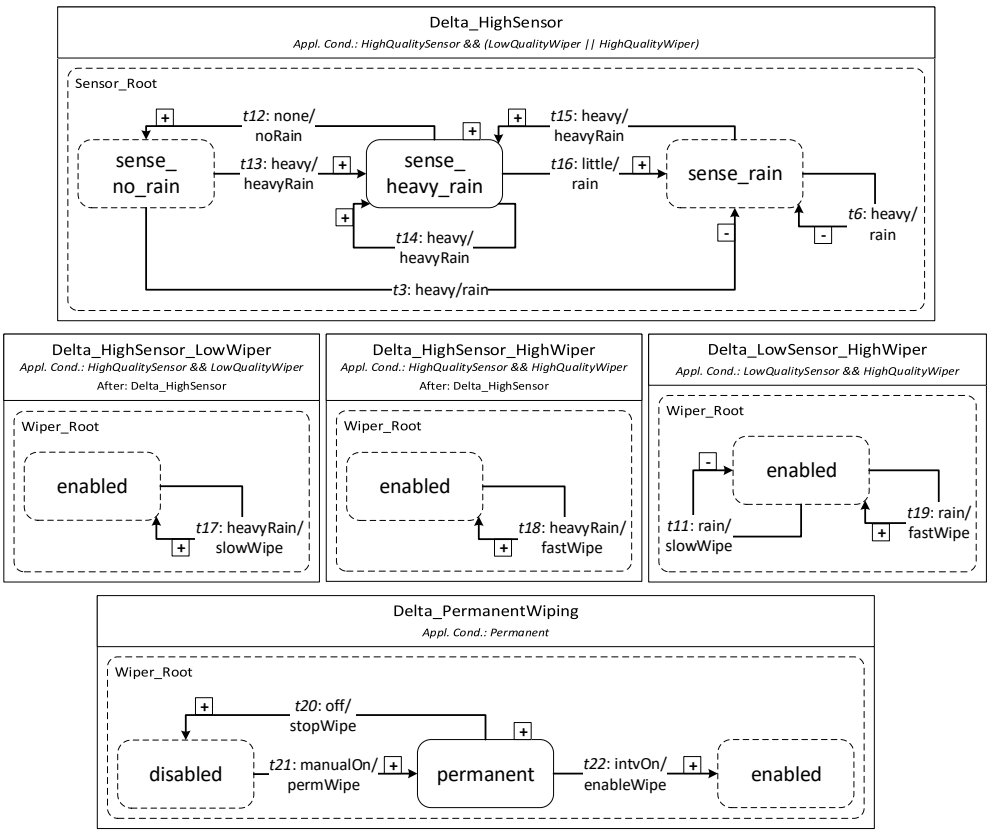

Figure 2.7: Deltas of the Wiper System SPL

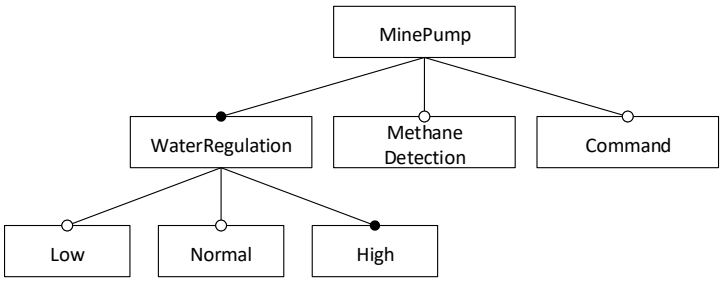

Figure 2.8: Feature Model of the Mine Pump System (according to [3])

water level based on feature `High` is absolutely necessary, whereas the handling of a normal or low water level due to features `Normal` and `Low` is optional selectable. Feature `Command` provides an opportunity to manually turn on and off the function of feature `WaterRegulation`. The feature `MethaneDetection` supplies an alarm interface as well as a methane sensor in order to prevent an explosion of methane gas induced by starting the pump.

Behavioral Specification

The mine pump system permanently monitors the water level with its sensor. In case of a high water level, the pump is started. After a certain period of time has elapsed, the pump is stopped. If the feature `Low` is selected, the pump automatically stops as soon as the water level is detected as low. Feature `Normal` does not perform any action in case of a normal water level. For selected feature `MethaneDetection`, the methane content in the air is permanently monitored. If, in this case, the water level is detected as high, the methane content is checked before starting the pump. In case a certain methane threshold is exceeded, the pump will not be started to prevent an explosion, even if the water level is high. Furthermore, the already running pump will be stopped if the permissable methane level is exceeded. In addition, the pump does not perform any monitoring or pump functions if it is manually turned off by the function of feature `Command`. This state holds until the system is manually turned on again.

The core model of the mine pump system is depicted in Fig. 2.9. It represents the state machine for the functionality for monitoring and handling a high water level. For this purpose, the state machine consists of the three concurrent substate machines `Pump_Ctrl_Region`, `Water_Monitoring_Region`, and `MinePump_System_Region`. `Water_Monitoring_Region` monitors the water level and sends events containing the information about a low, normal or high water level. `MinePump_System_Region` interprets the events about the water level in its substate machine `Water_Level_Control_Region`. In case of a high water level, the state machine checks if the pump is already running and, if not, sends the event to start the pump. Additionally, this state machine checks for how long the active pump is already running and stops it after a certain time span has elapsed. `Pump_Ctrl_Region` starts and stops the pump according to the events from `Water_Level_Control_Region`.

The deltas needed to derive further valid variants from the core state machine are shown in Fig. 2.10 and Fig. 2.11. `Delta_Add_Normal_Level_Handling` adds the transition `t18` to the core model for handling an event about a normal water level. `Delta_Add_Low_Level_Handling` adds the states `low_level_detected` and `pump_stopped_low` as well as the transitions `t19` for signaling a low water level, `t20` for handling an already stopped pump, `t21` for stopping the pump, and `t22` for returning to the initial state if the pump is already stopped.

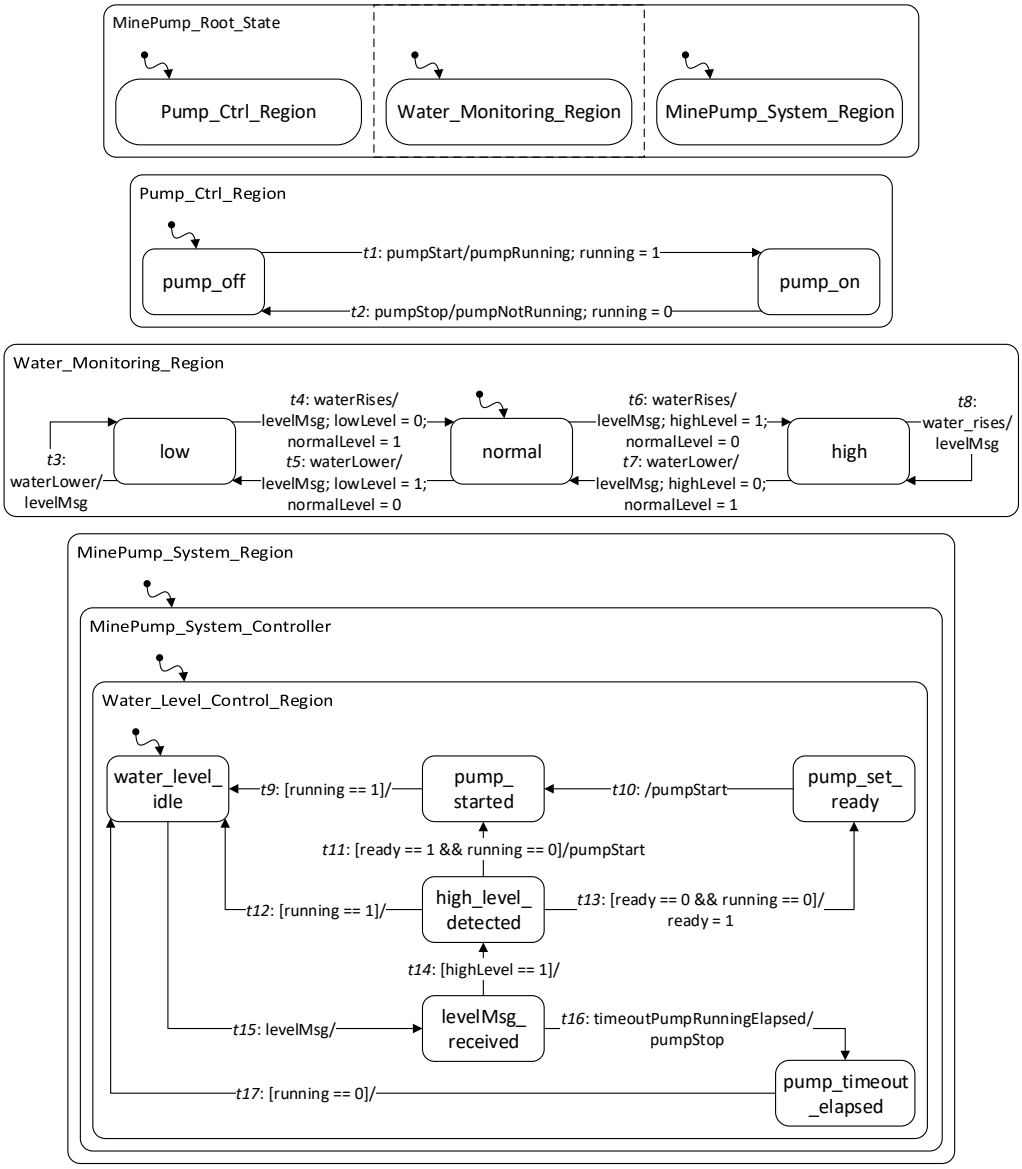

Figure 2.9: Core Model of the Mine Pump System SPL

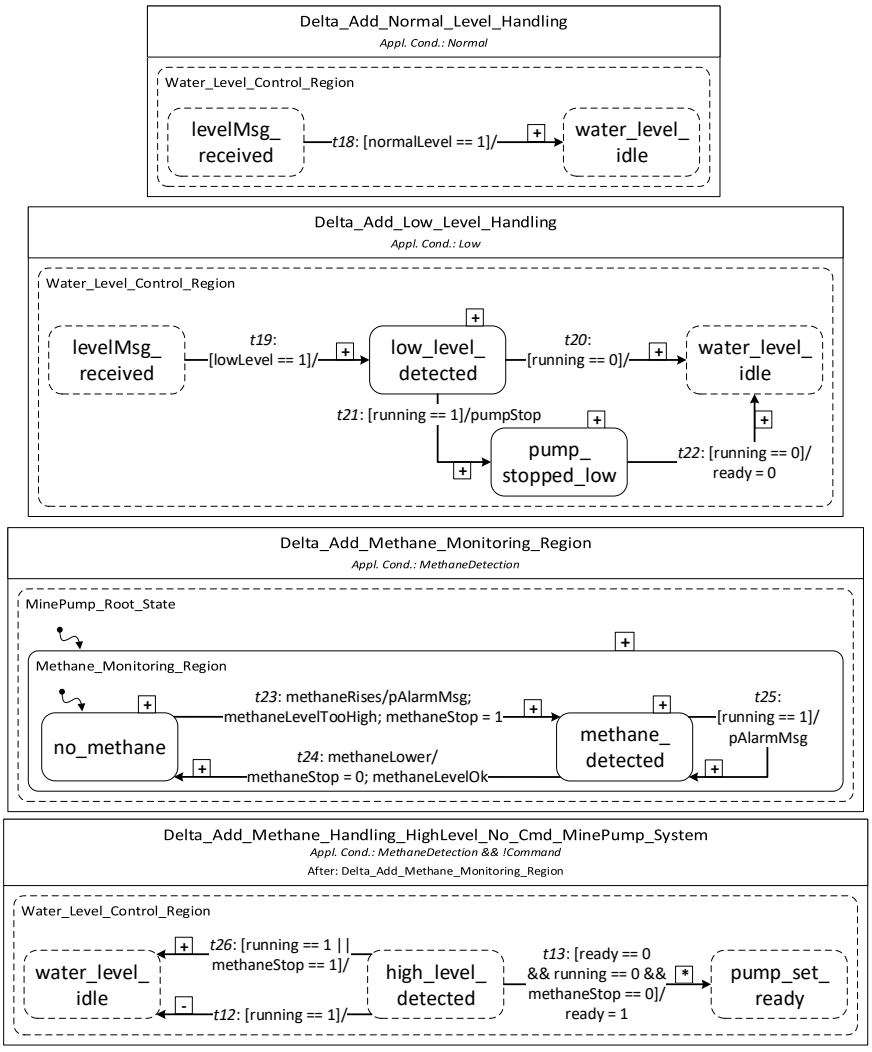

Figure 2.10: First Part of the Deltas of the Mine Pump System SPL

`Delta_Add_Methane_Monitoring_Region` adds the substate machine `Methane_Monitoring_Region` to `MinePump_Root_State`. `Methane_Monitoring_Region` is responsible for the methane monitoring and contains the states `no_methane` and `methane_detected` as well as the transitions `t23` releasing an alarm event when exceeding the methane threshold, `t24` for detecting a normal methane level, and `t25` for releasing further alarm events in case the pump is still running despite previous alarm events.

If feature `MethaneDetection` is selected, but not feature `Command`, `Delta_Add_Methane_Handling_HighLevel_No_Cmd_MinePump_System` replaces transition `t12` with transition `t26`. This transition returns to the initial state without an action not only when the pump is running, but also when the pump is blocked because of a high methane level. The delta can only be applied after `Delta_Add_Methane_Monitoring_Region`.

`Delta_Add_Methane_Handling_MinePump_System` adds the substate machine `Methane_Level_Control_Region` to `MinePump_System_Controller`. This substate machine consists of the states `methane_level_ctrl_idle`, `methane_detected_minepump_region`, and `pump_stopped_by_me-`

thane as well as the transitions t27, t28, t29, and t30. Transition t27 detects an alarm event, t28 and t29 are responsible for checking if the pump is running and if so, stopping it, and t30 returns to the initial state. Delta_Add_Methane_Monitoring_Region has to be applied prior to Delta_Add_Methane_Handling_MinePump_System.

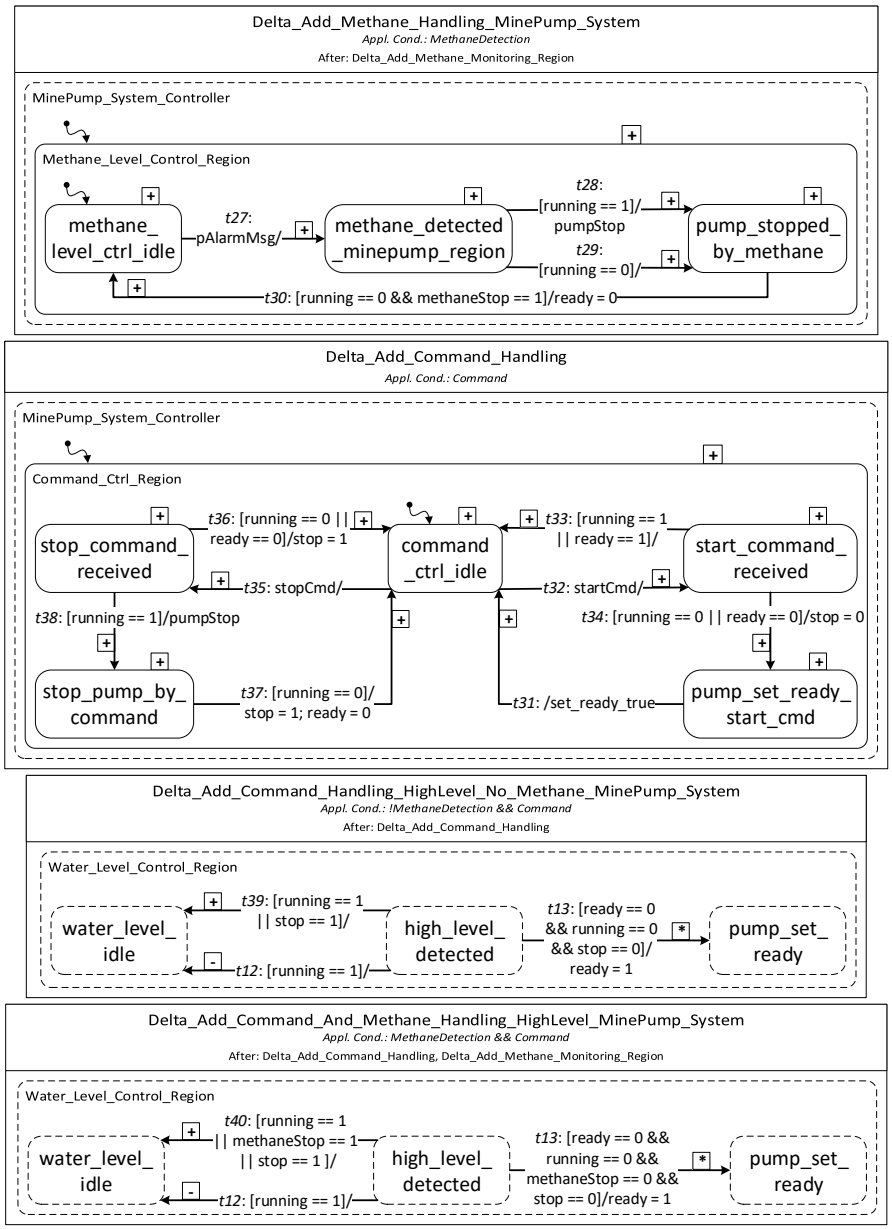

Figure 2.11: Second Part of the Deltas of the Mine Pump System SPL

For the selected feature Command, Delta_Add_Command_Handling adds the substate machine Command_Ctrl_Region to MinePump_System_Contoller. Command_Ctrl_Region contains the states command_ctrl_idle, start_command_received, pump_set_ready_start_cmd, stop_command_ received, and stop_pump_by_command as well as the transitions t31 to t34 for turning on the monitoring function, and transitions t35 to t38 for turning off the monitoring function.

If feature Command is selected, but not feature MethaneDetection, Delta_Add_Command_Handling_HighLevel_No_Methane_MinePump_System replaces transition t12 by transition t39. Despite a detected high water level, this transition additionally returns without action to the initial state if the pump was turned off by command. In order to use this delta, Delta_Add_Command_Handling has to be applied before.

In case both features MethaneDetection and Command are selected, Delta_Add_Command_And_Methane_Handling_HighLevel_MinePump_System replaces transition t12 with transition t40. This transition additionally returns without action to the initial state, if the pump is either turned off by command or blocked because of a high methane level. This delta can only be applied after Delta_Add_Command_Handling and Delta_Add_Methane_Monitoring_Region.

The product line of the mine pump system comprises 16 valid feature configurations and, thus, 16 different product variants.

2.4 Body Comfort System

The body comfort system [12] (BCS) describes various comfort functions of a vehicle. This case study was originally introduced by Müller et al. [**cdm2009**], later extended to a software product line by Oster et al. [16], and re-engineered to a delta-oriented SPL by Lity et al. [12]. An overview over all features of the body comfort system is depicted in the feature model in Fig. 2.12. Each product of the product line consists at least of a human machine interface and a door system. Additionally, the optional feature Security may be chosen.

Feature DoorSystem is the parent of the features ExteriorMirror and PowerWindow. Due to the mandatory child feature Electric, the exterior mirror is always electrically adjustable. Furthermore, it is heatable if feature Heatable is selected. The power window always comes with an anti-trap protection because of the mandatory feature FingerProtection. In addition, the power window can be operated manually (ManualPowerWindow) or automatically (AutomaticPowerWindow).

In case feature Security is selected, the customer has the possibility to chose an arbitrary combination of the features CentralLockingSystem, RemoteControlKey und AlarmSystem. If feature CentralLockingSystem is requested, the feature AutomaticLocking can be selected additionally. If feature RemoteControlKey is chosen, CentralLockingSystem has to be selected in any case. Furthermore, the features SafetyFunction, AdjustExteriorMirror, ControlAutomaticPowerWindow, and ControlAlarmSystem can be chosen for the selected feature RemoteControlKey. Feature ControlAlarmSystem requires AlarmSystem to be selected, whereas ControlAutomaticPowerWindow excludes the selection of ManualPowerWindow. The alarm system can be supplied with feature InteriorMonitoring.

If feature StatusLED is requested of feature HumanMachineInterface, at least one of the LEDs LEDFingerProtection, LEDPowerWindow, LEDExteriorMirror, LEDAlarmSystem, LEDCentralLockingSystem, and LEDHeatable are to be chosen. Feature LEDAlarmSystem needs AlarmSystem to be selected, LEDCentralLockingSystem is dependent on feature CentralLockingSystem, and for LEDHeatable feature Heatable has to be selected.

Behavioral Specification

Based on the described features, the body comfort system offers various functions simplifying the handling of the vehicle. Feature PowerWindow allows the electrical control of the window. In case of

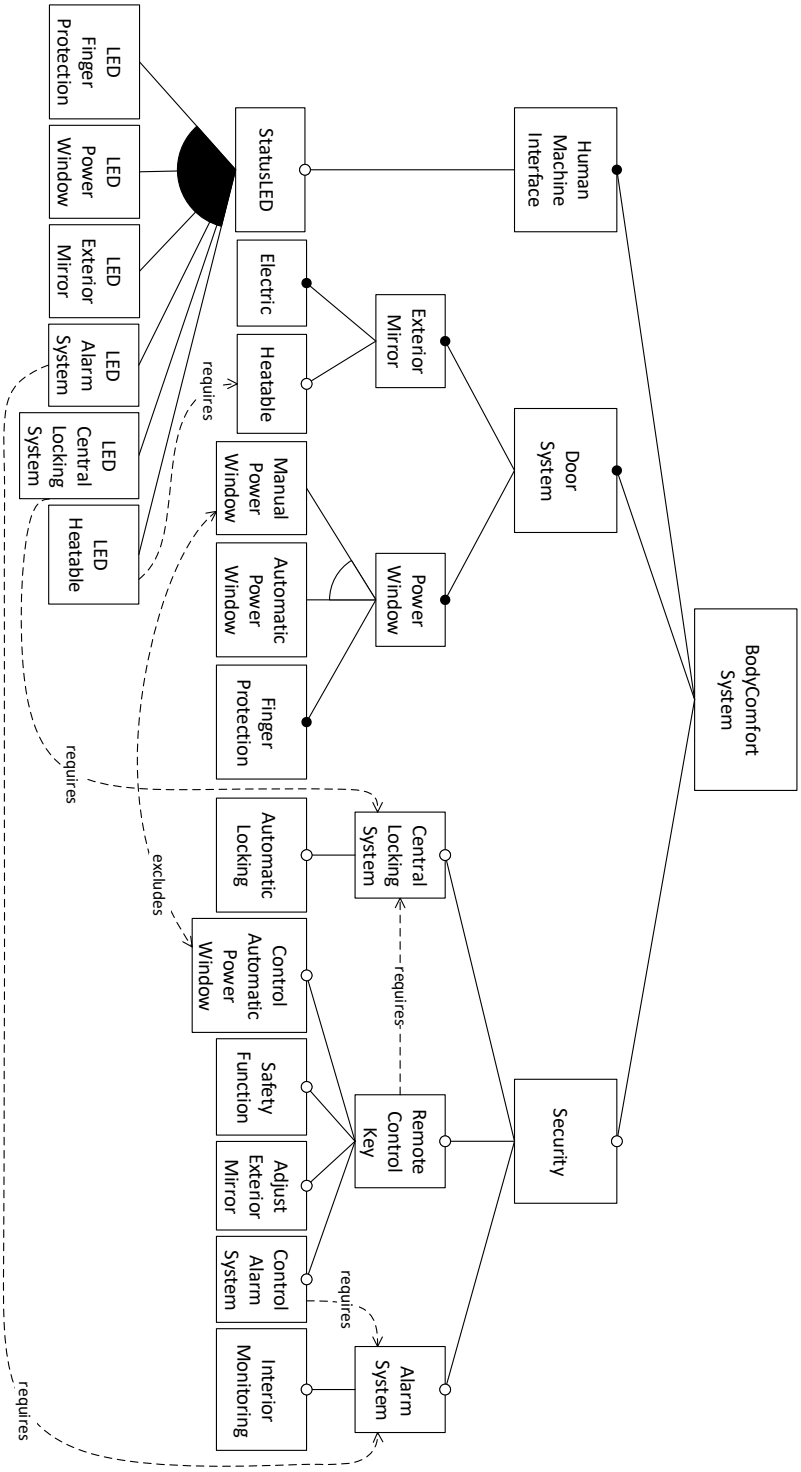

Figure 2.12: Feature Model of the Body Comfort System (according to [12])

manual control, the button has to stay pressed for the movement, whereas in case of automatic control, it is sufficient to press the button once in order to trigger an automatic movement of the window. If the button is pressed in the other direction during the movement, the movement will stop. Otherwise, if there is no other action during the movement, the movement will stop automatically as soon as the window reaches the uppermost, or alternatively lowermost, position. In case feature `FingerProtection` detects a resistance during the upwards movement, regardless if controlled manually or automatically, the movement of the window is stopped. Feature `LEDFingerProtection` turns a LED on, if a resistance was detected. If feature `LEDPowerWindow` is selected, a LED is active while the window is being opened or closed, no matter `PowerWindow` is chosen as manual or automatic. If `AutomaticPowerWindow` is selected, there is another LED glowing, when the window is being closed because of locking the car.

The position of the electrical exterior mirror can be adjusted with a button and, accordingly, moved left, right, up or down. If feature `Heatable` is selected, the heater of the mirror is automatically turned on if the outside temperature gets too low. After a certain time limit has elapsed, the heater is automatically turned off again. In case the according LED is selected for the product, the LED glows if the mirror heater is turned on. Furthermore, for the selected feature `LEDExteriorMirror`, there also exist LEDs glowing if the mirror has reached its stop position for the left, right, up or down direction.

The central locking system ensures that all doors are locked simultaneously if the vehicle is locked with the key. Here, the key may be a mechanical key or the remote key coming with feature `RemoteControlKey`. In case feature `AutomaticLocking` is selected additionally, the vehicle also automatically locks all doors as soon as it starts moving. If a variant contains the LED for the central locking system, the LED is active if the vehicle is locked. In case the body comfort system is equipped with feature `AutomaticPowerWindow` and there are still windows open when the car is locked with any key variant, the windows get closed automatically.

The remote control key requires the presence of the central locking system and serves to lock or unlock the vehicle at the press of a button. If feature `ControlAutomaticPowerWindow` is also selected, the key further has buttons for automatically opening and closing the windows from a certain distance. In case feature `SafetyFunction` is selected, its function monitors if a door is opened within a certain time span after unlocking the vehicle. If this is not the case, the vehicle automatically locks itself again to ensure the safety in case the vehicle was accidentally unlocked. If feature `ControlAlarmSystem` is selected, the alarm system is automatically activated as soon as the vehicle is locked. Hence, `ControlAlarmSystem` requires feature `AlarmSystem` to be selected.

The alarm system can be activated by locking the door with the mechanical key or manually by the use of the human machine interface. If the alarm is triggered, an acustic alarm signal is emitted. This signal will be automatically turned off after a certain time has elapsed. The alarm can also be manually turned off by using the human machine interface or, in case feature `ControlAlarmSystem` is existent, by unlocking the vehicle with the remote control key. Feature `LEDAlarmSystem` comes with an LED glowing if the alarm system is active. Furthermore, a LED glows if an alarm signal has been triggered, and another one as indication of the silent alarm when the audible alarm has been turned off after a certain time span. The alarm system can be enhanced by feature `InteriorMonitoring` with the result that the interior of the vehicle is also monitored and an alarm is triggered if a movement is detected inside the car. There also glows an LED in this case.

The core model of the body comfort system consists of the substate machines HMI (Human Machine Interface), Man_PW (Manual Power Window), FP (Finger Protection) und EM (Exterior Mirror). The core model is shown in Fig. 2.13 and Fig. 2.14.

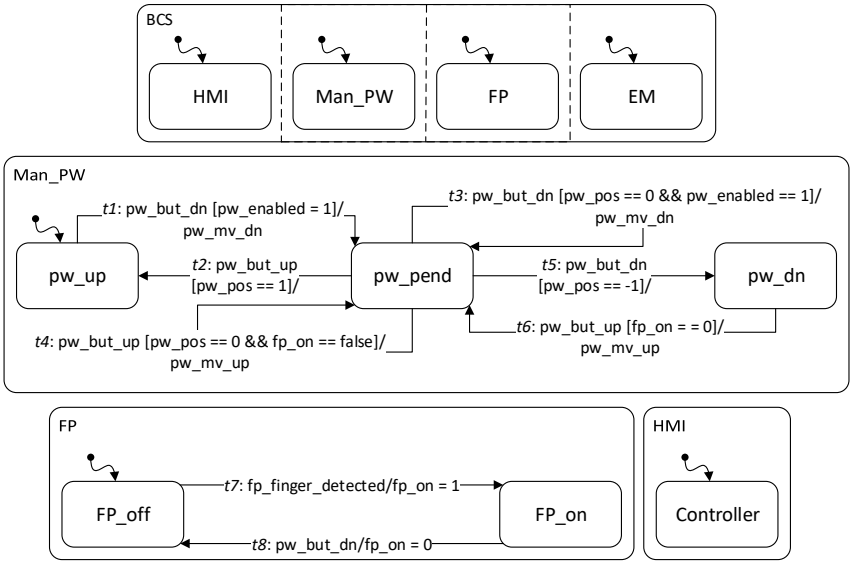

Figure 2.13: First Part of the Core State Machine of the Body Comfort System SPL

Substate machine Man_PW contains the states pw_up representing a closed window, pw_dn representing a completely opened window and pw_pend representing an arbitrary position in between. If the button for opening the window is pressed, a downward movement is executed, provided the window is closed (t1) or opened arbitrarily wide other than completely (t3). Once the window reaches the lowermost position (t5), the movement stops despite the pressed button. In contrast, if the button for closing the window is pressed, no resistance is detected by the finger protection, and the window is opened completely (t6) or arbitrarily wide (t4), a closing movement is executed. As soon as the window is closed (t2), the movement stops, even if the button for closing is pressed. Substate machine FP detects if the window hits a resistance and activates the finger protection stopping the upwards movement of the window. Substate machine HMI solely consists of the empty substate machine Controller.

Substate machine EM comprises nine states representing the mirror position. These states show if the mirror is positioned completely at the right, left, top, bottom, top left, bottom left, top right, or bottom right end position. Furthermore, state em_pending represents all positions in between. Movements executed in between the end positions are embodied by loops (e.g. t11 or t30). All other transitions describe movements inducing the mirror to leave an end position (e.g. t21 or t37) or to reach an end position (e.g. t13 or t29).

Since the complete body comfort system is very large, only a few relevant deltas from the delta model changed in Chap. 3.4 are introduced. An overview over the complete set of deltas is documented by Lity et al. [10, 12].

The deltas described in this section are the ones responsible for adding the LED status display and the central locking system. The deltas are depicted in Fig. 2.15. Delta DAddStatusLED adds the

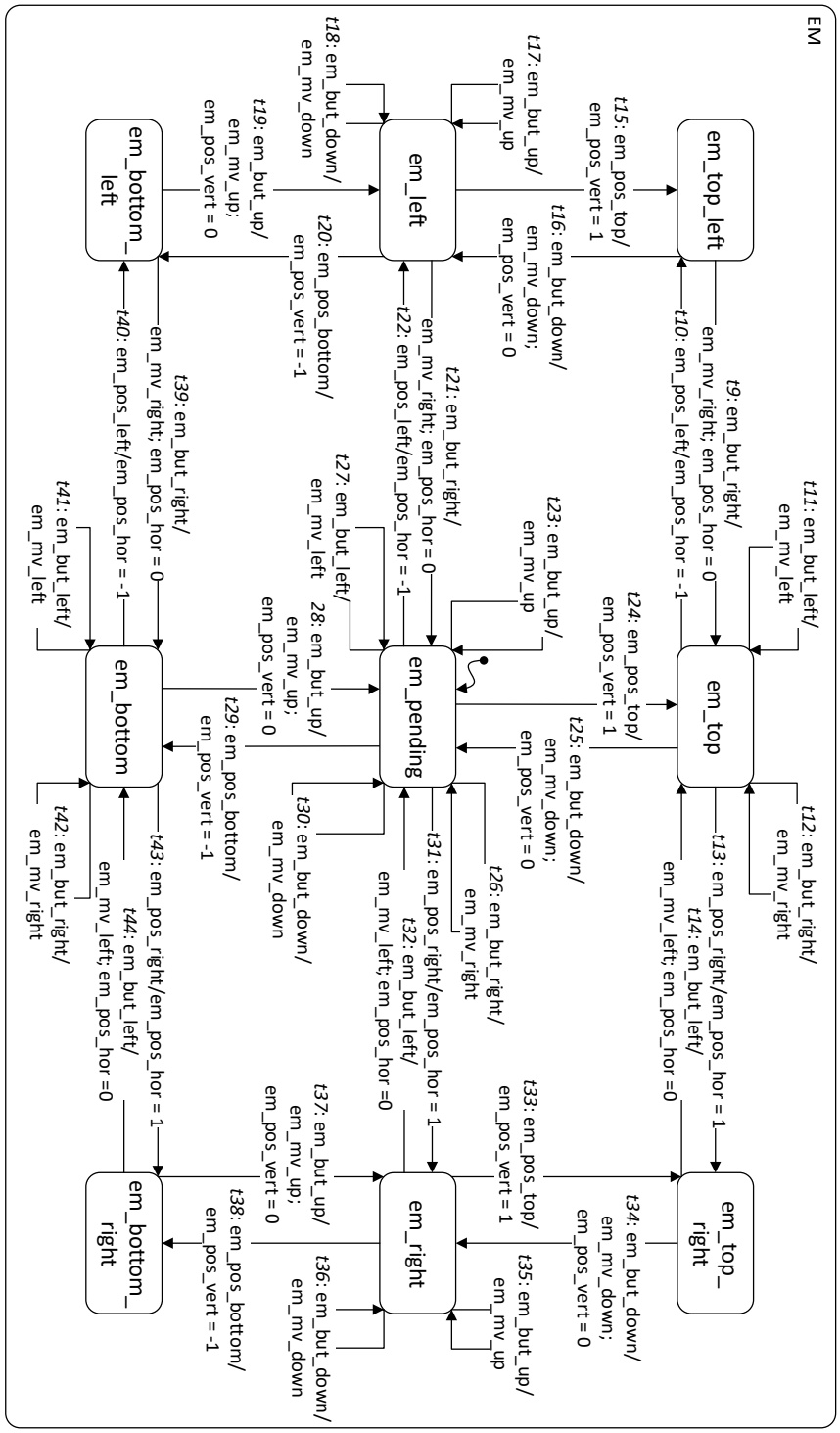

Figure 2.14: Second Part of the Core State Machine of the Body Comfort System SPL

empty substate machine LED to the core state machine. All further substate machines responsible for the control of any LED are inserted into LED. Delta DAddCLS adds the empty substate machine CLS to the core. The states CLS_unlock and CLS_lock as well as the transition t48 are added to CLS by delta DAddCLSBSM. Transition t48 unlocks the vehicle and allows for the opening of the windows, once the door was manually unlocked by the mechanical key. DAddCLSBSM can only be applied after delta DAddCLS.

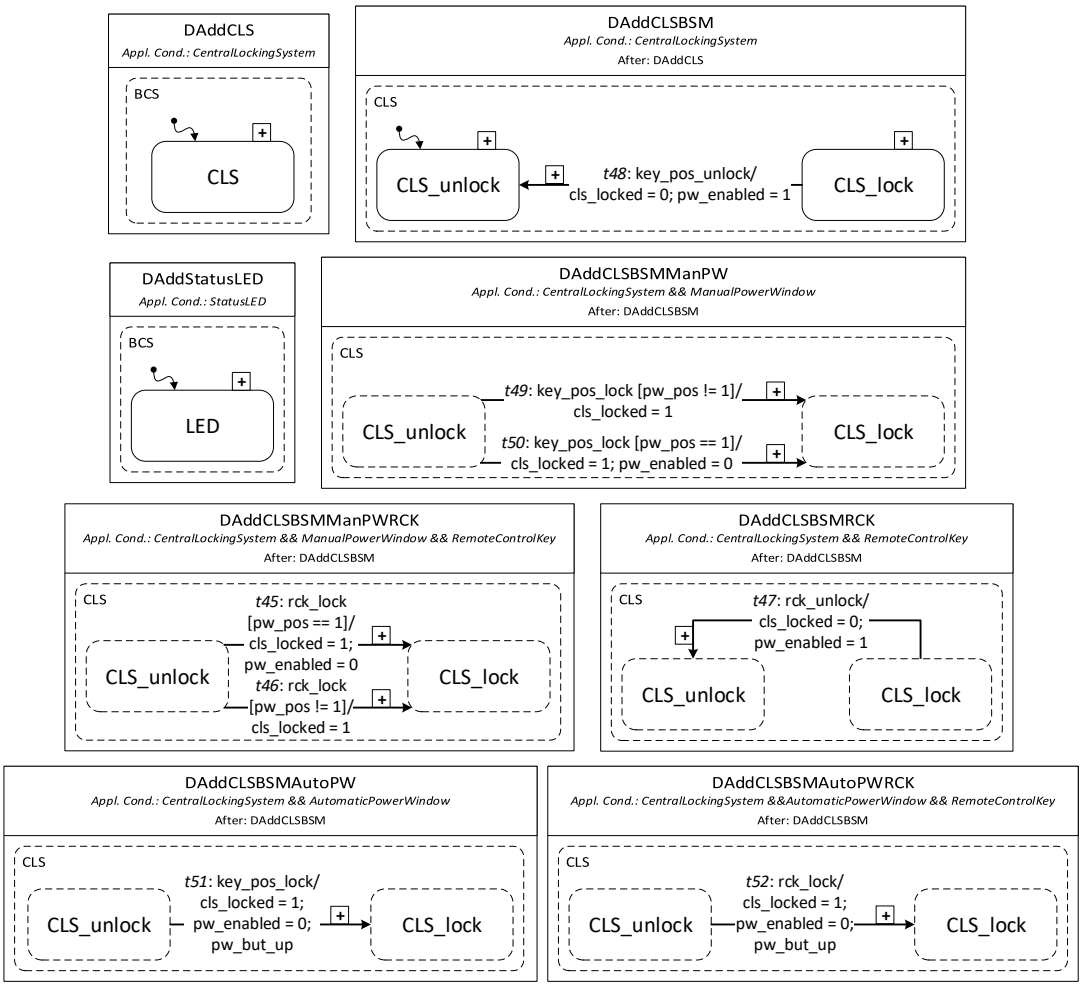

Figure 2.15: Selected Examples of the Deltas of the Body Comfort System SPL

Delta DAddCLSBSMManPW is used if feature ManualPowerWindow is selected in addition to feature CentralLockingSystem. In this case, the transitions t49 and t50 are additionally added to substate machine CLS. Transition t49 locks the vehicle if the door is manually locked by the mechanical key while not all windows are closed, and t50 locks the vehicle and blocks the usage of the power window if the door is manually locked by the mechanical key while all windows are closed. If feature AutomaticPowerWindow is selected, delta DAddCLSBSMAutoPW adds transition t51. This transition locks the vehicle and automatically closes the windows if any window is still open when locking the vehicle with the remote control key. It simultaneously prevents any further usage of the power window.

Delta DAddCLSBSMManPWRCK adds the transitions t45 and t46 to substate machine CLS. These transitions perform the same functions as t49 and t50, but for locking the vehicle with the remote control key instead of the mechanical key. If delta DAddCLSBSMRCK is used, transition t47 is also added to CLS. Transition t47 shows the same behavior as t48 when locking the vehicle with the remote control key. Delta DAddCLSBSMAutoPWRCK adds transition t52 performing the same actions as t51 for locking the vehicle with the remote control key. All five deltas DAddCLSBSMManPW, DAddCLSBSMAutoPW, DAddCLSBSMRCK, DAddCLSBSMManPWRCK, and DAddCLSBSMAutoPWRCK can only be applied after delta DAddCLSBSM.

The product line of the body comfort system consists of 11,616 feature configurations and, therefore, product variants.

3 Evolution Scenarios

In this chapter, various evolution scenarios for the systems described in the previous chapter are defined and the changes made by the scenarios using higher-order delta modeling [11] are presented. The scenarios are defined in a chronological order w.r.t. each SPL. For each scenario, the initial situation is outlined, followed by the illustration of the changes to be made. The visualization of the changes is supported by at most one sample variant-specific state machine, regardless of the number of new variants resulting due to the evolution. The complete number of new variants is presented in the description of the changes of the corresponding delta models.

3.1 Vending Machine

This section describes the evolution scenarios for the vending machine SPL. In total, there are seven scenarios including the introduction of different beverage sizes, the variable selection of beverage sizes, the removing of one beverage size, the addition of a milk counter, a milk display, and an offering of milk to tea and coffee as well as the selection of different milk types.

3.1.1 Different Beverage Sizes

This evolution scenario describes the addition of a choice between different beverage sizes.

Description

Initial Situation: In order to get a valid vending machine, the beverages coffee, tea and cappuccino can be mixed arbitrarily and the currency can be either euro or dollar. Furthermore, there is the possibility to emit a ring tone once the beverage is ready if the corresponding option is selected.
Scenario: In europe, the requests increasingly arise for different beverage sizes. Thus, different choices for the size are to be added to the european machines. Additionally to the regular size available up to now, a small and a large size will be added.

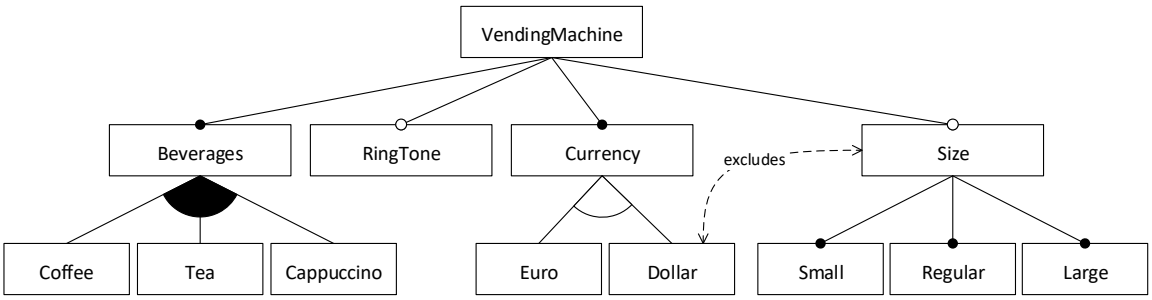

Figure 3.1: Feature Model of the Vending Machine with new Feature `Size`

Realization: The new feature `Size` is added to the product line of the vending machine. Since the choice of a beverage size should only be available for european machines, the optional feature `Size`

is added to the feature model in Fig. 3.1 and has an `excludes`-relation to the feature `Dollar`. In case the choice of beverage sizes gets selected for a product, all three sizes are supposed to be offered. Thus, the child features `Small`, `Regular` and `Large` are mandatory.

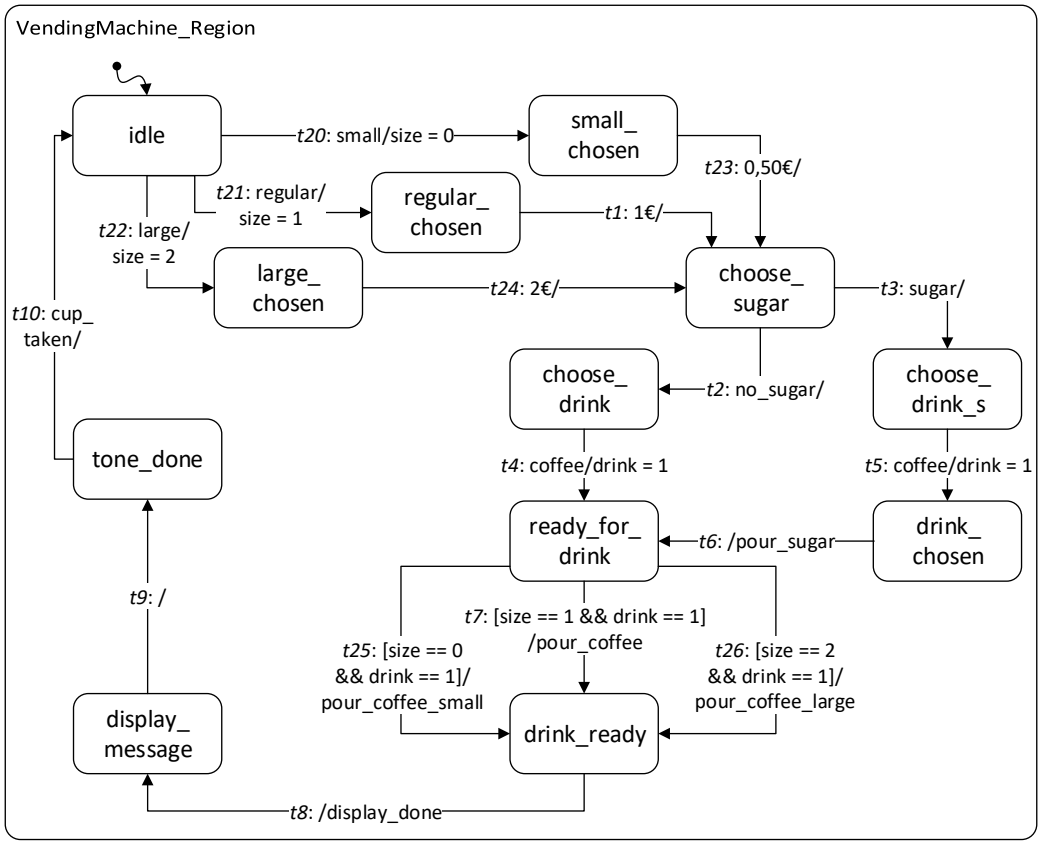

Figure 3.2: Product Model of the Vending Machine with Different Sizes

For the new beverage sizes, there also have to be different prices. The regular size is still priced at 1 €, whereas the small size will cost 0.50 € and the large size 2 €. We assume that there has to be given the exact amount of money for the beverage.

The product model which is required to realize a machine offering coffee in different sizes is shown in Fig. 3.2. For this state machine, the states `small_chosen`, `regular_chosen`, and `large_chosen` as well as the according transitions from and to the states `idle` and `choose_sugar` were added compared to the core state machine depicted in Fig. 2.2. Furthermore, the transitions `t25` and `t26` were added between the states `ready_for_drink` and `drink_ready`, while the initial transition `t7` was modified so that the size gets evaluated by the guard as well. For a machine offering the beverages tea and cappuccino, there have to be added similar transitions between the states `ready_for_drink` and `drink_ready`.

Modeling

In order to derive the machines described in this chapter, six new deltas are added to the delta model. This is specified by the higher-order delta `Size` shown in Fig. 3.3.

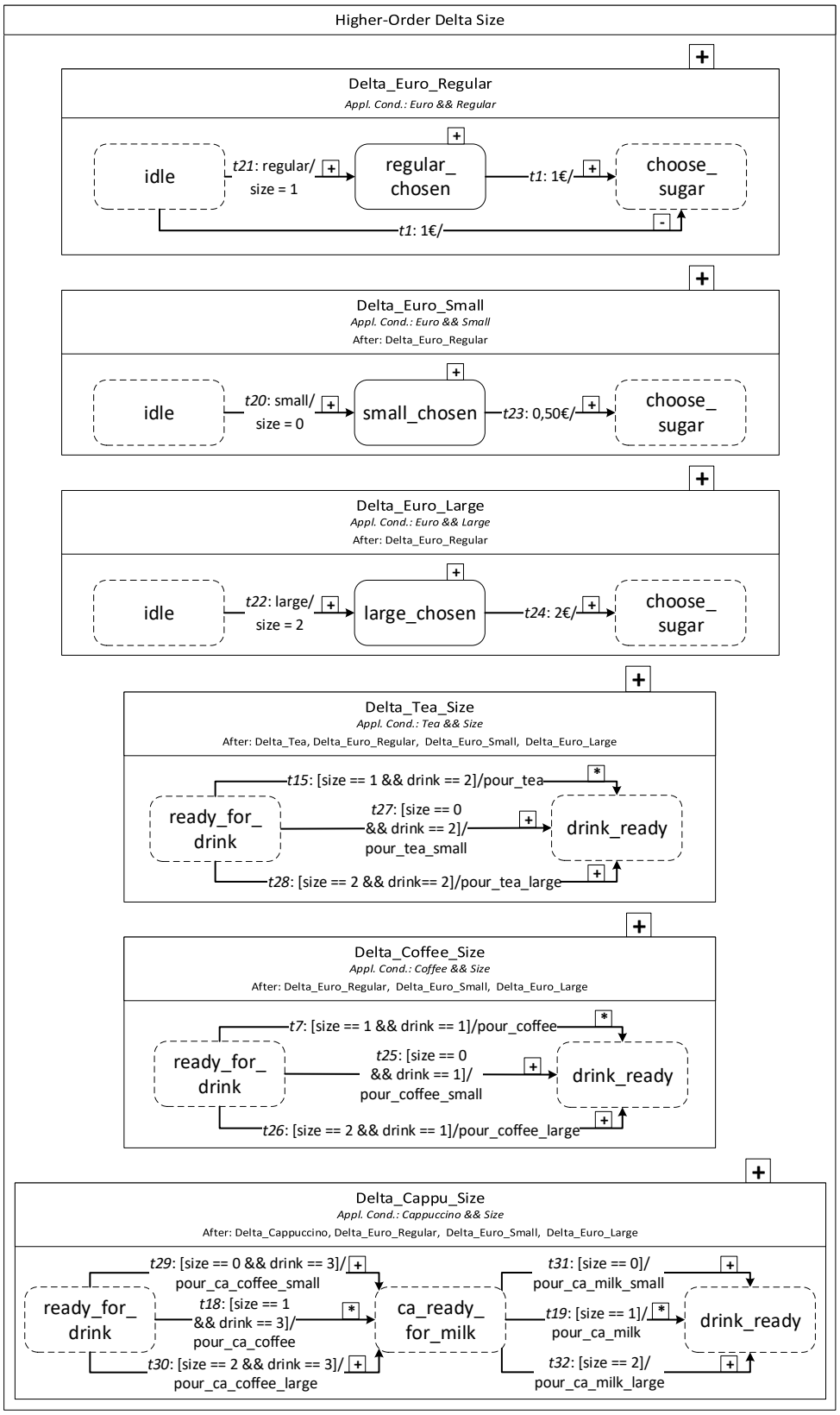

Figure 3.3: Higher-Order Delta for Adding Deltas for Different Beverage Sizes

To include the different prices of the beverages, the deltas `Delta_Euro_Small`, `Delta_Euro_Regular`, and `Delta_Euro_Large` are added to the delta model described in Chap. 2.1 by higher-order delta `Size`. `Delta_Euro_Small` adds the state `small_chosen` and the transitions `t20` and `t23` for the small size. For the large size, the state `large_chosen` and the transitions `t22` and `t24` are added similarly by `Delta_Euro_Large`. `Delta_Euro_Regular` adds the state `regular_chosen` and the transition `t21`. It further removes the existing transition `t1` between the states `idle` and `choose_sugar` and adds it again between the new state `regular_chosen` and `choose_sugar`.

The different sizes for the three beverages are realized by the added deltas `Delta_Tea_Size`, `Delta_Coffee_Size` and `Delta_Cappu_Size`. Based on the addition of the transitions `t27` and `t28`, `Delta_Tea_Size` adds a small and a large tea. Analogously, a small and a large coffee are added by `Delta_Coffee_Size` as well as a small and a large cappuccino by `Delta_Cappu_Size`. In addition, the already existing transitions `t15`, `t7`, `t18`, and `t19` are modified so that the size conditions are taken into account.

Due to this scenario, 14 new feature configurations with respective variant-specific delta sets and, thus, product variants are added to the vending machine SPL resulting in a total number of 42 variants.

3.1.2 Variable Beverage Sizes

This scenario describes the possibility to combine the different beverage sizes in a variable way.

Description

Initial Situation: Additionally to the choice of the different beverages, the alternative between two currencies, and the optional tone emission, there is the option to choose different beverage sizes. If a product offers the choice of beverage sizes, all three sizes *small*, *regular*, and *large* have to be offered.
Scenario: The vending machine distribution observes that there are regionally different preferences regarding the beverage size. Southern Europe orders more small beverages, while the north favors large beverages. The sales numbers for the regular size equal roughly in both regions. Therefore, the choice of beverage sizes shall be designed more flexible. When the choice of beverage sizes gets selected for a variant, there is now the possibility to decide if the small, the large or both sizes are to be offered. However, the regular size has to be offered in any case.

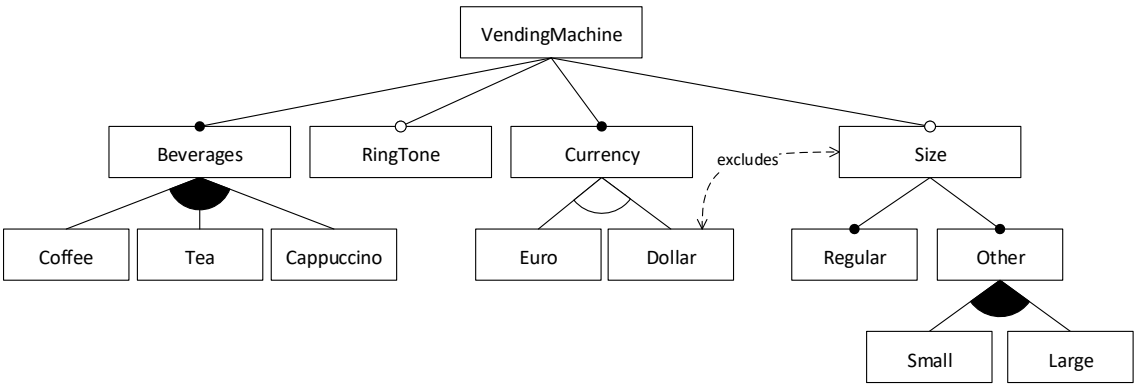

Figure 3.4: Feature Model of the Vending Machine SPL with a Variable Choice of Sizes

Realization: To design the child-features `Small` and `Large` of the feature `Size` as variable, they are transformed from mandatory- to or-features. In doing so, it is ensured, that at least one of the two features has to be selected whenever the feature `Size` gets selected. The feature `Regular` does not get changed and, thus, stays mandatory. Hence, it is guaranteed that there is always the choice of at least two beverage sizes in a product. The new feature model is shown in Fig. 3.4.

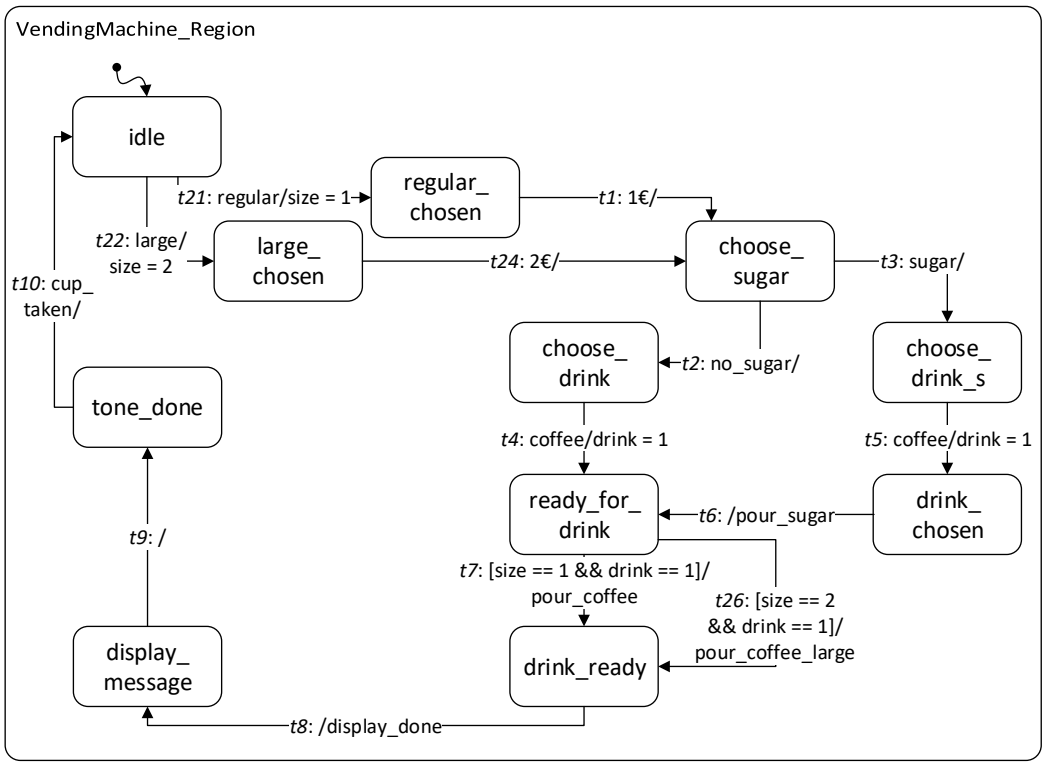

Figure 3.5: Product Model of the Vending Machine with Two Beverage Sizes

The product state machine depicted in Fig. 3.5 represents a machine for the feature `Size` with the selected features `Regular` and `Large`. In order to build such a product state machine, the different beverage sizes and their prices have to be added separately from one another in contrast to the scenario described in Sect. 3.1.1.

Modeling

In this evolution scenario, three deltas are modified and six deltas are added to the delta model. The higher-order delta `Variable_Size` capturing these changes is shown in Fig. 3.6.

Since the different sizes will be available separately from each other, the respective deltas adding the sizes must be adapted. The deltas `Delta_Euro_Small`, `Delta_Euro_Regular`, and `Delta_Euro_Large` responsible for the different costs are already divided according to the size, so they do not have to be changed. Thus, solely the deltas `Delta_Tea_Size`, `Delta_Coffee_Size`, and `Delta_Cappu_Size` adding the beverage dispensing in different sizes have to be modified. For this purpose, `Delta_Tea_Size` gets modified with the result that it only modifies `t15` from now on. To compensate the loss of adding the small and large tea, two new deltas `Delta_Tea_Small` and `Delta_Tea_`

Large are created. These deltas add the transitions t27 and t28. The deltas Delta_Coffee_Size and Delta_Cappu_Size for coffee and cappuccino are modified similarly to the procedure described for tea. Accordingly, the four additional new deltas Delta_Coffee_Small, Delta_Cappu_Small, Delta_Coffee_Large, and Delta_Cappu_Large are added to the delta model. Those deltas add the transitions t25, t29, and 31 as well as t26, 30, and t32 to the core model.

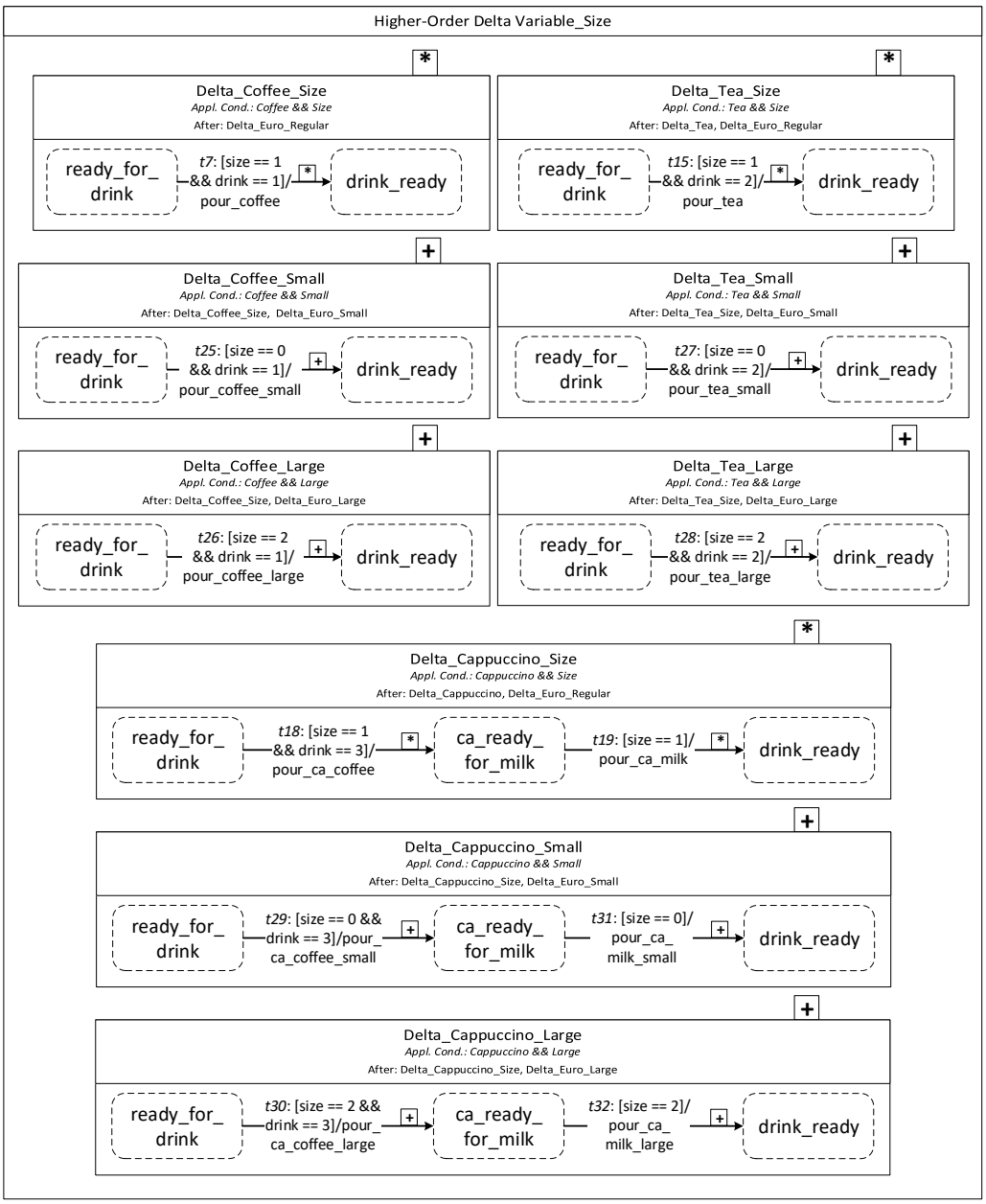

Figure 3.6: Higher-Order Delta for the Adding of Variable Beverage Sizes

This scenario adds 28 new feature configurations with respective variant-specific delta sets to the vending machine SPL such that the total number of product variants increases to 70. The modifications of the deltas result in 14 modified variants, whereas 28 variants remain unchanged.

3.1.3 Removal of One Beverage Size

This scenario describes the removal of one of the beverage sizes.

Description

Initial Situation: There are still three different beverages at choice, one of two possible currencies has to be chosen and if requested there is a tone emission. Furthermore, there exists the possibility to offer different beverage sizes. In case the different beverage sizes are offered, the regular size always has to be covered and additionally one or both of the sizes *Small* and *Large* have to be selected.
Scenario: Consumer surveys revealed, that only very low numbers of small beverages are sold. Since mostly beverages in the sizes *Regular* and *Large* are sold, the vending machine operators order less and less vending machines with the offer of small beverages. Thus, the vending machine manufacturer decides for the offering of small beverage sizes that it is not profitable enough and withdraws it from the product line. Hence, in the future there still exists the option to offer different beverage sizes, but there are only *Regular* and *Large* beverages available. Those two beverages have to be selected on an obligatory basis for the feature `Size`.
Realization: Accordingly to the new requirements of the product line, the feature `Small` is removed from the feature model shown in Fig. 3.7. Furthermore, the features `Large` and `Regular` are now mandatory. This way, it is ensured that there is a distinction in case the choice of different sizes is selected.

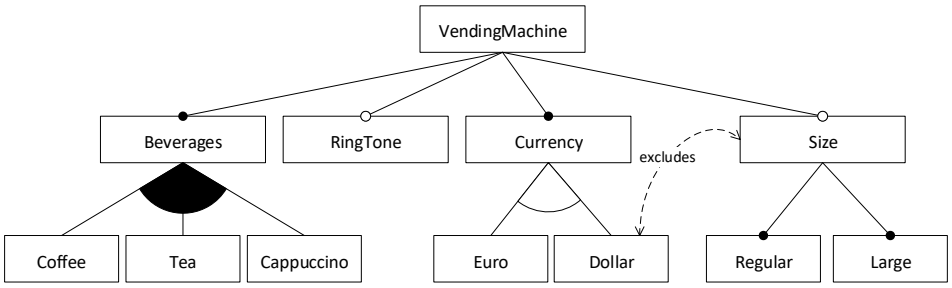

Figure 3.7: Feature Model of the Vending Machine SPL without Beverage Size `Small`

A product state machine with the offering of the beverage sizes *Regular* and *Large* is already depicted in Fig. 3.5 in Sect. 3.1.2. The generation of such a product model is already possible based on existing deltas.

Modeling

In order to generate product state machines matching the new requirements, no new deltas are required as there are already enough deltas for the addition of the beverage sizes. However, the deltas adding the small size still exist in the delta model. Those deltas are obsolete. Therefore, the deltas `Delta_Tea_Small`, `Delta_Coffee_Small`, `Delta_Cappu_Small`, and `Delta_Euro_Small` are removed from the delta model via the higher-order delta `Remove_Small`. `Remove_Small` is shown in Fig. 3.8.

This scenario solely reduces the number of valid feature configurations by 28, with the result that 42 variants remain unchanged.

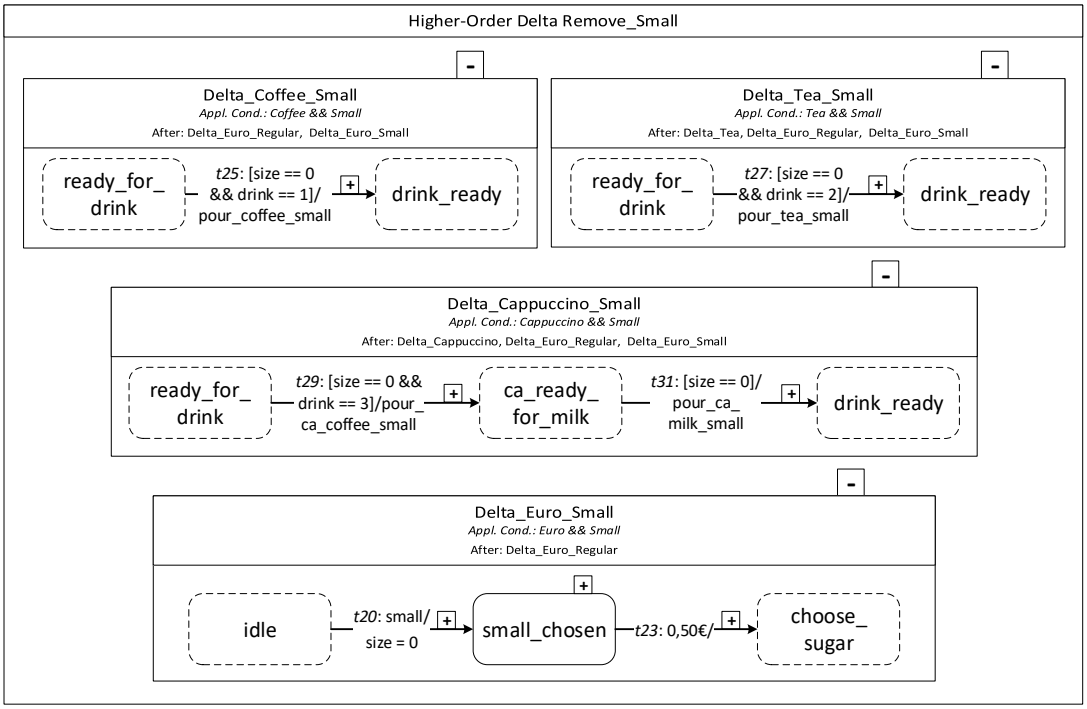

Figure 3.8: Higher-Order Delta for the Removal of Beverage Size *Small*

3.1.4 Milk Counter

In this scenario, the addition of a milk counter is described. This scenario is inspired by the vending machine introduced by Kamischke et al. [8].

Description

Initial Situation: To get a valid vending machine, at least one of the beverages coffee, tea and cappuccino can be selected, a currency has to be chosen and there is the possibility to offer two beverage sizes as well as to optionally emit a tone.

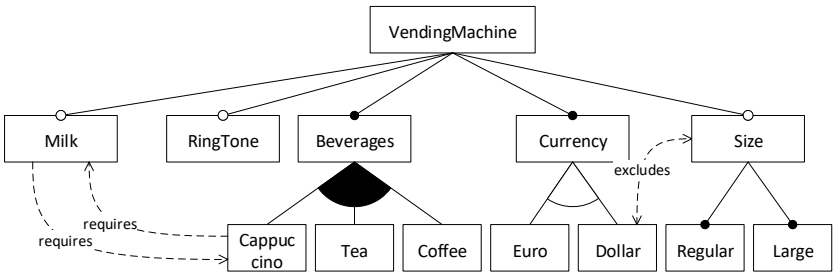

Figure 3.9: Feature Model of the Vending Machine SPL with the New Feature `Milk`

Scenario: For vending machines offering cappuccino, complaints of the customers about incorrect served beverages become more frequent. Thus, cappuccinos with insufficient or even entirely without milk were served frequently. In all cases, the cause was an empty milk tank. Up to now, there is

no possibility to check the liquid level of the milk without laboriously disassembling the machine completely. Hence, a milk counter is to be installed in each vending machine offering cappuccino.
Realization: The new feature `Milk` is added to the feature model of the product line depicted in Fig. 3.9. Since each vending machine offering cappuccino is delivered with a milk counter from now on, there is a `requires`-relation from `Cappuccino` to `Milk`. Thus, it is guaranteed that the feature `Milk` can not be mixed with the features `Coffee` and `Tea` which do not contain milk without the presence of `Cappuccino`.

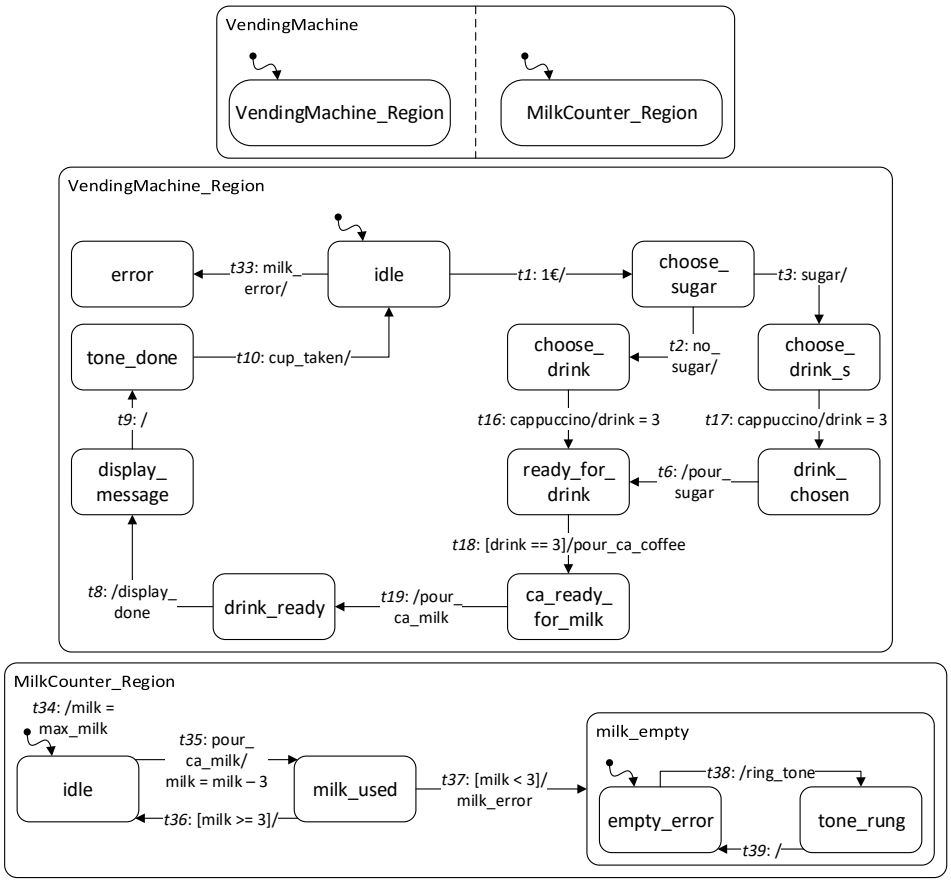

Figure 3.10: Product Model of the Vending Machine with a Milk Counter

Figure 3.10 shows a product state machine offering the beverage cappuccino in just one size along with the milk counter. The milk counter is realized in a new substate machine. Initially, after refilling the machine, the milk counter is set to the maximum liquid level. Each time milk is poured for a regular cappuccino, the liquid level is reduced by the poured amount, namely three units of milk. Afterwards, the system checks if there is enough milk left for another cappuccino. For a machine offering different beverage sizes, the poured amount of milk for a large cappuccino accounts for four units. The remaining liquid level is always compared with the biggest amount of milk needed for the offered beverages. In case there is not enough milk left, a tone is emitted until the machine is serviced. Additionally, the whole system transitions into an error state so that no beverages can be sold anymore.

Modeling

The deltas for the realization of a milk counter are added by the higher-order delta `Milk_Counter` depicted in Fig. 3.11. `Delta_Milk_Error` adds state `error` and transition `t33` to the `VendingMachine_Region`. By using `Delta_Milk_Counter`, the substate machine `MilkCounter_Region` is added to state machine `VendingMachine`. This substate machine consists of the states `milk_idle`, `milk_used`, and `milk_empty` as well as the transitions `t35` for decreasing the value of the milk counter, `t36` for a sufficient liquid level, and `t37` for an insufficient liquid level. State `milk_empty` is composed of the substates `empty_error` and `tone_rung` together with the transitions `t38` for emitting a tone and `t39`. Transition `t34` sets the liquid level value to the maximum value.

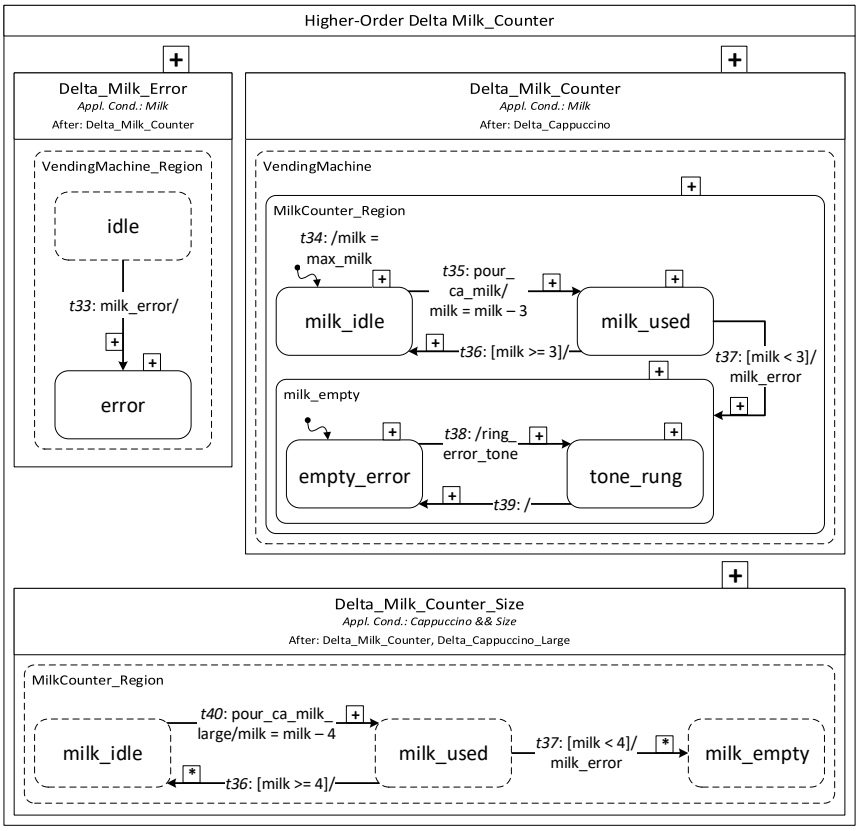

Figure 3.11: Higher-Order Delta for the Addition of a Milk Counter

`Delta_Milk_Counter_Size` is used whenever the features `Milk` and `Size` are offered together in one variant. In that case, the transition `t40` for decreasing the milk counter value by four units for a large cappuccino is added to the substate machine `MilkCounter_Region`. Additionally, the transitions `t36` and `t37` are modified, such that the error condition is triggered by the time there is not enough milk left to produce a large cappuccino.

By adding a milk counter to all vending machines offering cappuccino, all 24 variants containing feature `Cappuccino` in their feature configurations are modified as their variant-specific delta sets are also modified. Consequently, there still exists a total number of 42 product variants, where 18 variants remain unchanged.

3.1.5 Milk Display

In this scenario, the addition of a milk display is described. This scenario is also inspired by Kamischke et al. [8].

Description

Initial Situation: The product line of the vending machine offers the alternative between two different currencies, an optional tone emission when completing a beverage and the possibility to provide two beverage sizes. The beverages coffee, tea, and cappuccino are served and whenever cappuccino is sold a milk counter has to be used as well.
Scenario: Due to the introduction of the milk counter, there are significant losses in the beverage business. Whenever the liquid level of the milk is low, the machines offering cappuccino are out of service for several days before the maintenance service comes to clean and refill the machine. For that reason, a milk display displaying the liquid level of the milk is to be added to machines so that the maintenance service can be called before the milk is completely consumed.
Realization: There is no new feature added for realizing the milk display, but the existing feature `Milk` is extended for that purpose. Thus, the feature model of this scenario matches the model depicted in Fig. 3.9 from Sect. 3.1.4.

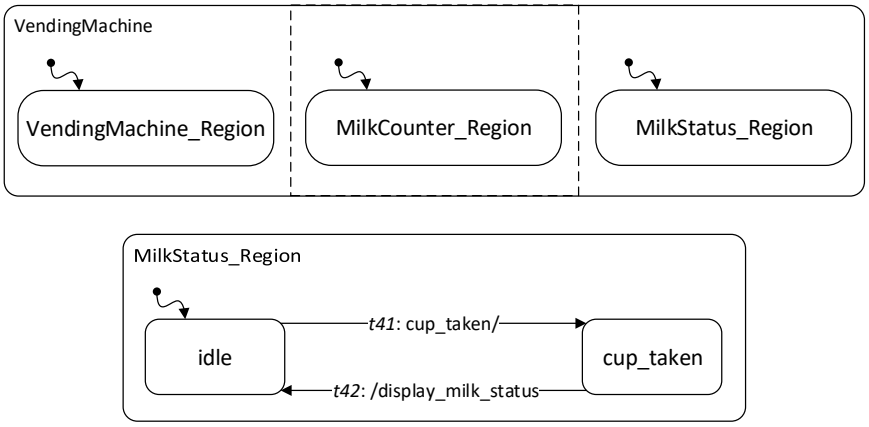

Figure 3.12: Product Model of the Vending Machine with a Milk Display

To realize the milk display, the product state machines of vending machines offering cappuccino have to be extended by the new substate machine `MilkStatus_Region`. This substate machine is shown in Fig. 3.12 and induces the displaying of the current liquid level each time a cup of an arbitrary beverage is taken. The substate machines `VendingMachine_Region` and `MilkCounter_Region` are not modified compared to the state machine in Fig. 3.10 and, thus, are not shown additionally for this scenario.

Modeling

Figure 3.13 depicts the higher-order delta `Milk_Status` which adds solely the delta `Delta_Milk_Status` to the delta model. This delta adds the substate machine `MilkStatus_Region` to `VendingMachine`. The substate machine consists of the states `ms_idle` and `ms_cup_taken` as well as the transitions `t41` for recognizing that the cup is taken and `t42` for displaying the liquid level.

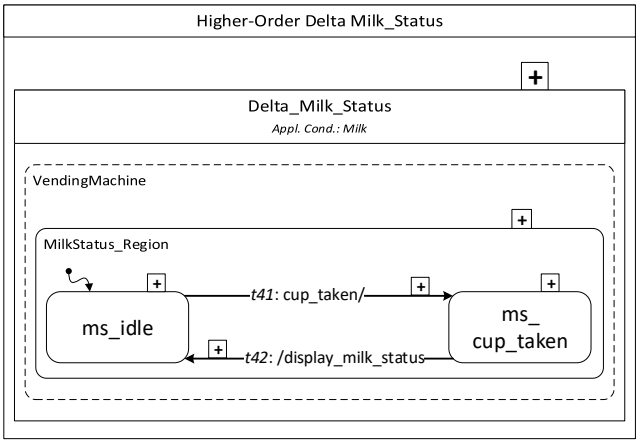

Figure 3.13: Higher-Order Delta for the Addition of a Milk Display

In this scenario, neither new variants are introduced nor obsolete variants are removed. However, the function of feature `Milk` is extended by the milk display. This causes all 24 product variants containing `Milk` to be modified. Thus, the total number still amounts to 42 product variants, where 18 variants remain unchanged.

3.1.6 Milk Obligatory for Coffee and Optional for Tea

This scenario describes the addition of an optional choice of milk for tea and the mandatory choice of milk for coffee.

Description

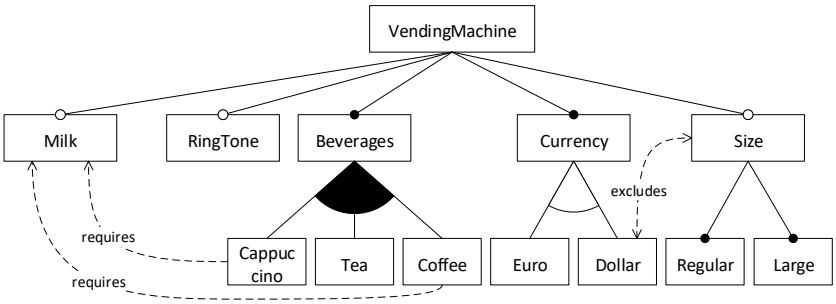

Figure 3.14: Feature Model of the Vending Machine SPL with an Obligatory Milk Selection for Coffee and an Optional Selection for Tea

Initial Situation: A valid vending machine needs one of two currencies, can emit a tone if requested and provides either one or two beverage sizes. It can offer an arbitrary choice of the beverages coffee, tea, and cappuccino, and if cappuccino is offered, a milk counter and a milk display are provided.

Scenario: Regarding vending machines offering coffee, complaints increasingly arise from customers criticizing that merely sugar can be chosen for the coffee but not milk. Furthermore, in case

of vending machines offering cappuccino as well as coffee, the cappuccino is not an adequate compensation since customers only want a dash of milk for their coffee and not coffee consisting almost half of foamed milk. In order to make allowances for these accumulating customer wishes, milk will be standardly offered for coffee. At the same time customers from Britain and East Frisia ask for milk for their tea. Thus, milk shall be optionally offered for tea to cover the regionally different preferences of tea. The milk counter and display have to be supplied for coffee and tea with milk functionality.

Realization: The offer of milk for coffee and tea is already realized in the feature model by feature `Milk`. As shown in Fig. 3.14, there are only small changes to the feature model. The `requires`-relation from `Milk` to `Cappuccino` is dropped, since milk is not only offered for cappuccino anymore. To ensure that coffee is offered with milk from now on, there is now a `requires`-relation from feature `Coffee` to feature `Milk`. For products offering tea together with coffee or cappuccino, there exists automatically the possibility to choose milk for tea.

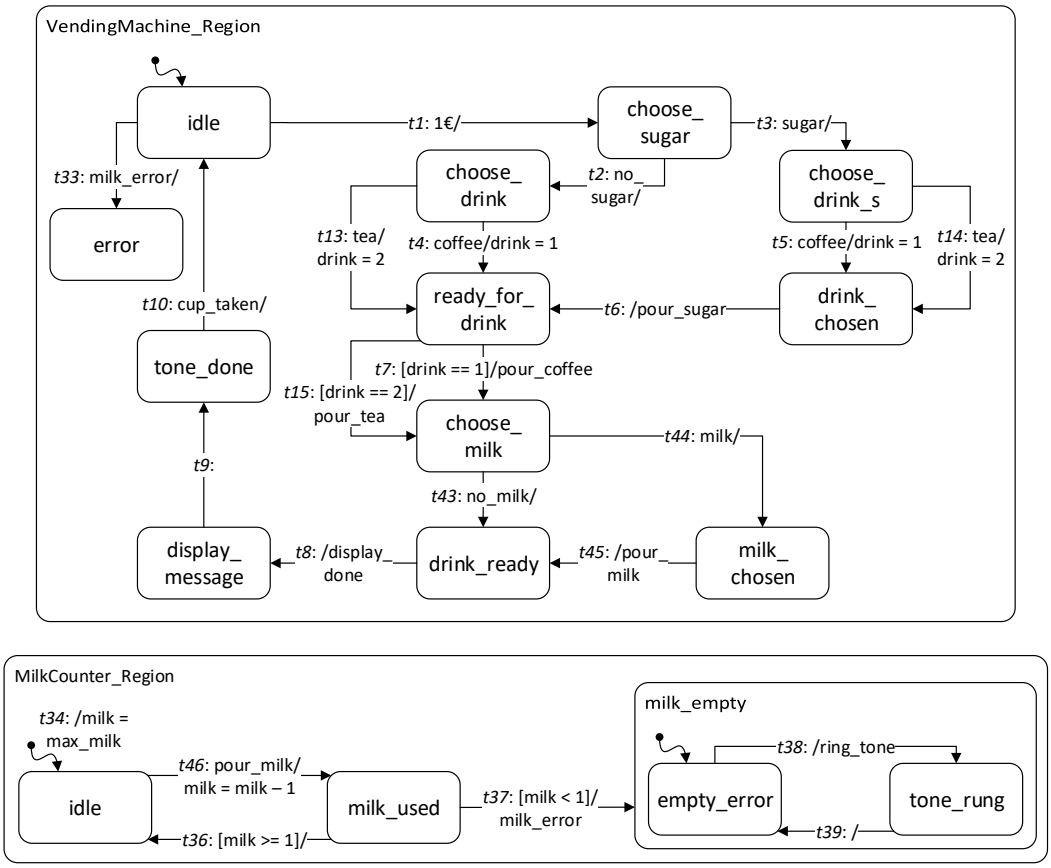

Figure 3.15: Product Model of the Vending Machine with a Choice of Milk for Coffee and Tea

Figure 3.15 depicts a product state machine offering milk for coffee and tea. For both beverages, after pouring the beverage, the customer is offered the possibility to choose milk for the beverage or not. In order to get such a product model, there has to be a delta in the delta model to add the milk choice. Additionally, the value of the milk counter is decreased by one unit after each pouring of milk, whereas the decreasing by three units is dropped because of the missing cappuccino. Also,

some obsolete functions can be removed since coffee can not be offered without milk anymore.

Modeling

Higher-order delta `Milk` shown in Fig. 3.16 and Fig. 3.17 removes the deltas `Delta_Coffee_Regular` and `Delta_Coffee_Large`, modifies the deltas `Delta_Milk_Counter` and `Delta_Milk_Counter_Size`, and adds nine new deltas.

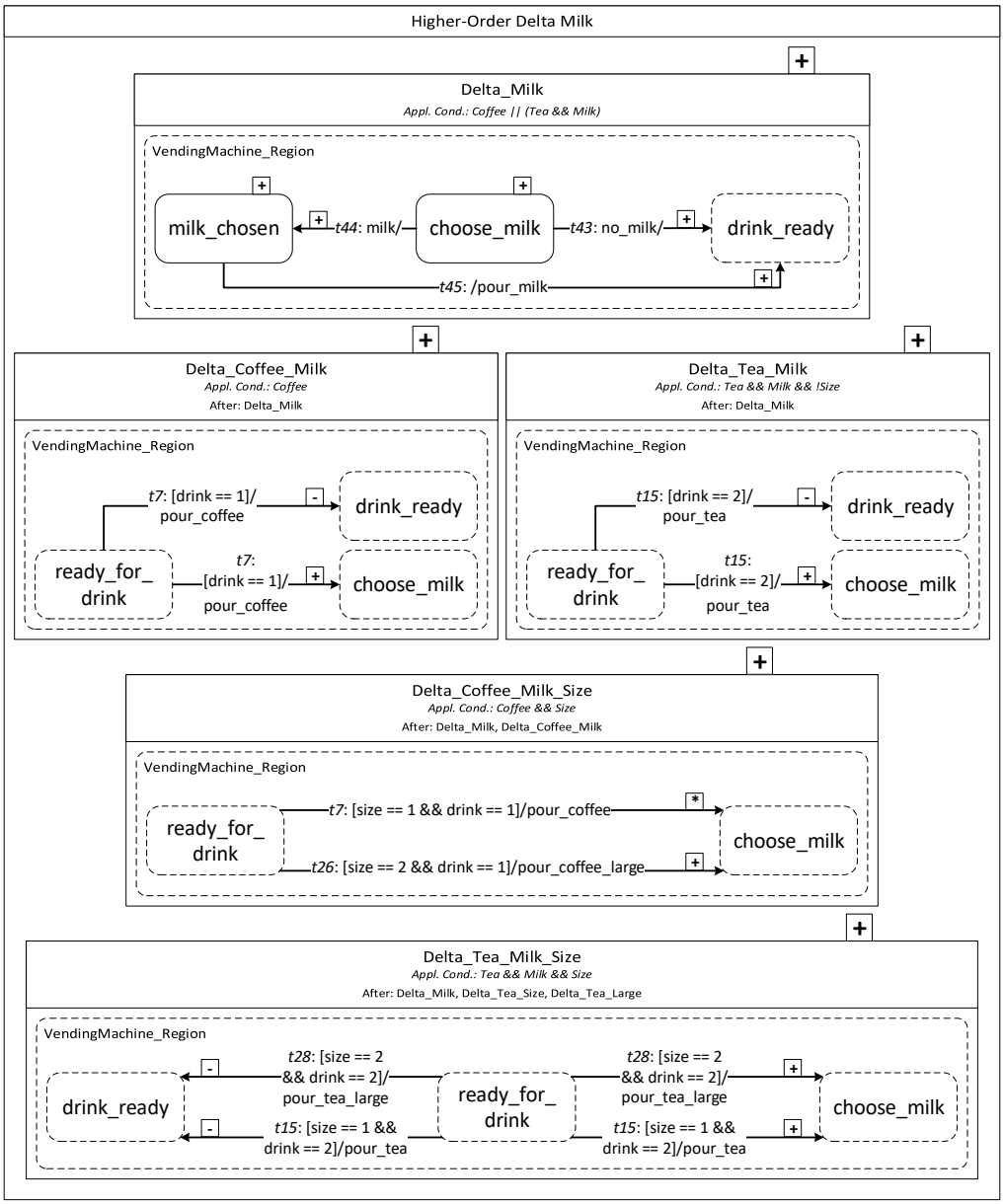

Figure 3.16: First Part of the Higher-Order Delta for Milk for Coffee and Tea

The addition of the functionality of the milk choice for coffee and tea is made by the new delta `Delta_ Milk`. `Delta_Coffee_Milk` replaces transition `t7` between `ready_for_drink` and `drink_ready` with `t7` between `ready_for_drink` and `choose_milk`. `Delta_Coffee_Milk` can only be applied after `Delta_Milk`. The same applies for `Delta_Tea_Milk` with transition `t15`. `Delta_Tea-`

_Milk can also only be applied after Delta_Milk and, additionally, when feature Size is not selected.

Delta_Coffee_Size and Delta_Coffee_Large are removed from the delta model. The replacing and adding of transitions t7 and t26 is now done by Delta_Coffee_Milk_Size. This delta is only applied after Delta_Milk and Delta_Coffee_Milk. Delta_Tea_Milk_Size is selected for application when feature Size is selected. It relocates transitions t15 and t28. Prior to applying Delta_Tea_Milk_Size, the deltas Delta_Milk, Delta_Tea_Size, and Delta_Tea_Large have to be applied.

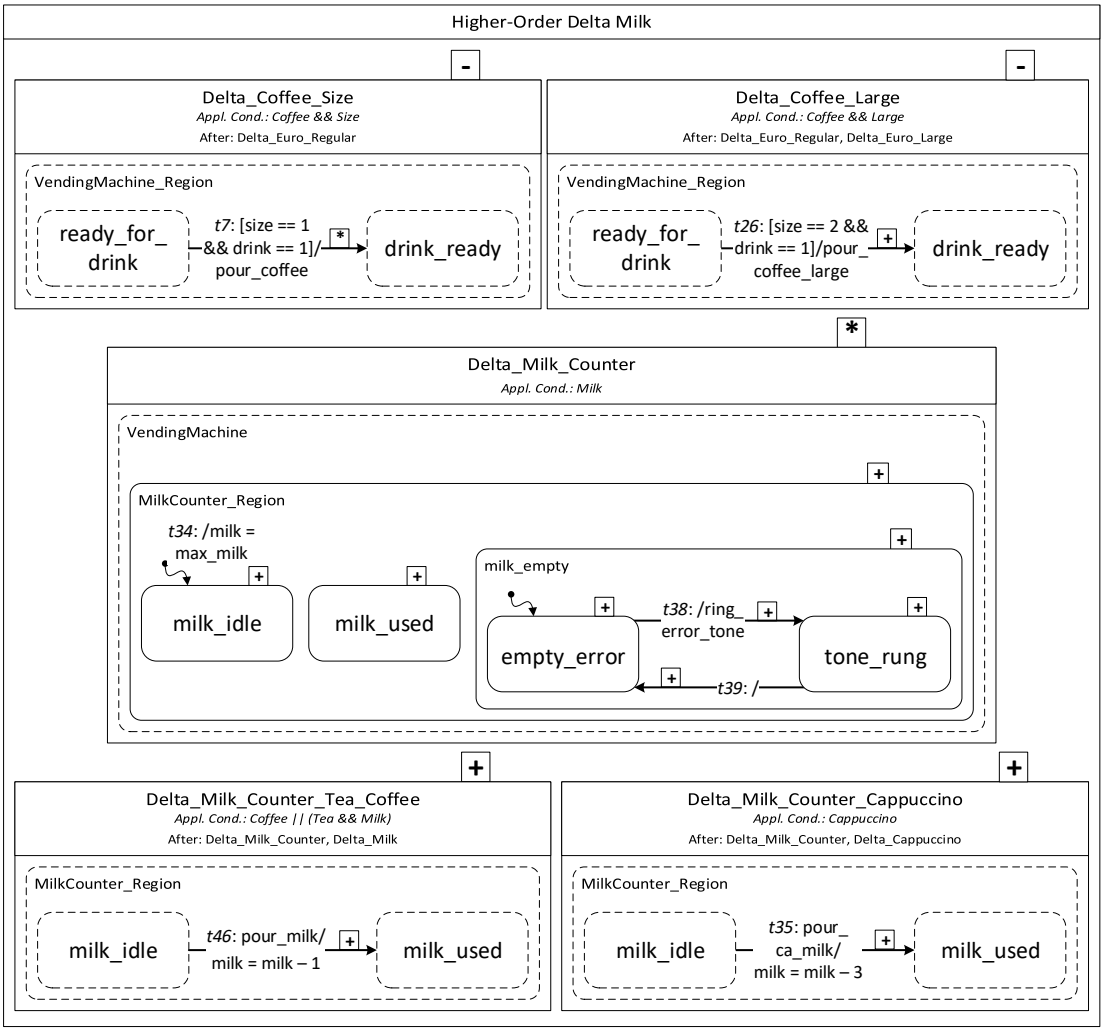

Figure 3.17: Second Part of the Higher-Order Delta for Milk for Coffee and Tea

To adapt the milk counter for handling coffee and tea, Delta_Milk_Counter is modified. Transitions t35, t36, and t37 are specialised in cappuccino and, thus, are not added by Delta_Milk_Counter anymore. Transition t35 is now added by Delta_Milk_Counter_Cappuccino. This delta can only be applied after Delta_Milk_Counter. Transitions t36 and t37 are added by Delta_Milk_Counter_Comparison_Cappuccino which can also only be applied after Delta_Milk_Counter.

Whenever coffee or tea with milk shall be offered by a vending machine, transition t46 decreasing the milk counter by only one unit is added by Delta_Milk_Counter_Tea_Coffee. For vending

machines not offering cappuccino, `Delta_Milk_Counter_Comparison_NoCappuccino` is used to add transitions `t36` and `t37`. In contrast to the transitions `t36` and `t37` used for cappuccino, these transitions do not display an error message until the remaining milk level falls below one milk unit instead of three.

Finally, `Delta_Milk_Counter_Size` is modified to adapt the application order to the new and modified deltas. Additionally to `Delta_Milk_Counter`, `Delta_Milk_Counter_Comparison_Cappuccino` has to be applied prior to `Delta_Milk_Counter_Size`.

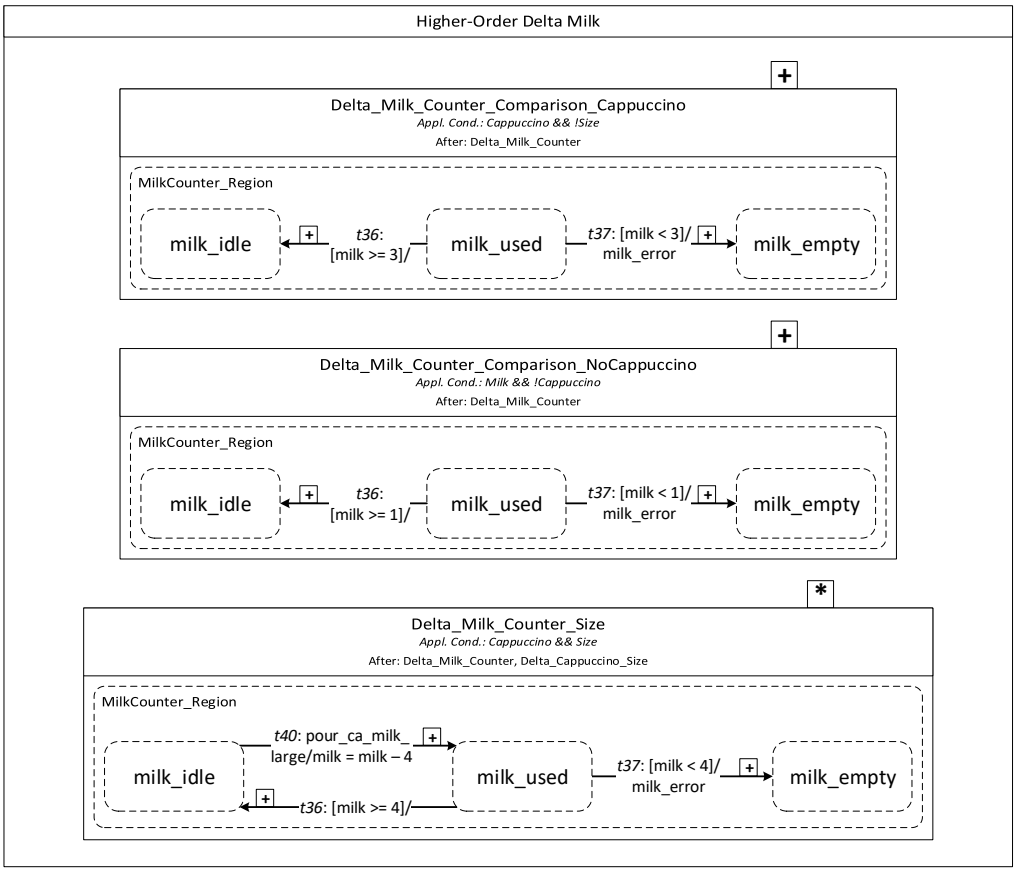

Figure 3.18: Third Part of the Higher-Order Delta for Milk for Coffee and Tea

Due to this scenario, 18 new feature configurations with respective variant-specific delta sets are added, while 12 variants are deleted. Furthermore, 24 existing product variants are modified. This is primarily because of the newly added deltas. Consequently, there are in total 48 product variants specifying the vending machine SPL, where six variants remain unchanged.

3.1.7 Different Milk Types

This evolution scenario describes the extension of the milk functionality by a choice of different milk types.

Description

Initial Situation: The product line of the vending machine offers two different currencies, the possibility to emit a tone after finishing a beverage, two different beverage sizes and the beverages coffee, tea and cappuccino. Vending machines offering cappuccino or coffee always need milk, whereas milk is optional for vending machines only offering tea. Whenever milk is contained in a vending machine, a milk counter and a display are included.

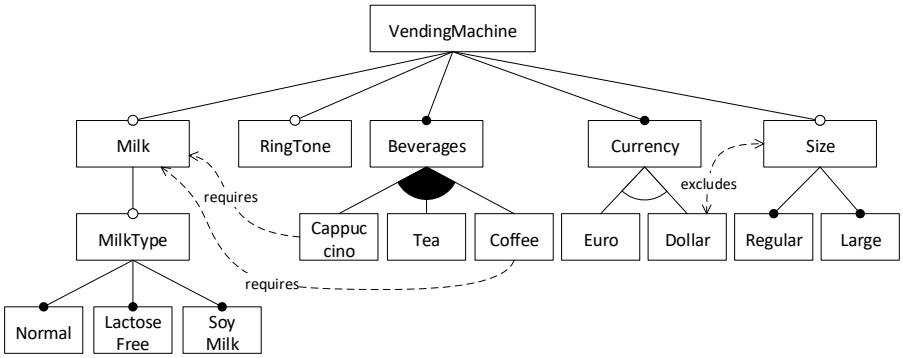

Figure 3.19: Feature Model of the Vending Machine SPL with Different Milk Types

Scenario: The introduction of the milk offer for coffee and tea was a visible success. In consequence of veganism and spreading diseases of civilization like lactose intolerance, the vending machine producers decide to broaden the milk offer for the machines. For lactose intolerant people, there will be offered lactose free milk and for vegans there will be soy milk. Normal milk is still offered as well.

Realization: To realize the choice of different milk types, the new optional feature `MilkType` is added to the feature diagram depicted in Fig. 3.19. `MilkType` has three mandatory child features: `Normal`, `LactoseFree`, and `SoyMilk`. Thus, if feature `MilkType` gets selected, all three milk types are offered. The feature automatically affects all beverages selected for the vending machine.

Figure 3.20 shows a product state machine featuring coffee and cappuccino as well as the three milk types. After choosing milk for coffee, the customer now has to choose which milk type he desires. When choosing cappuccino, after pouring the coffee it is automatically asked which milk is wanted. For the milk counter, the particular liquid levels of the different types are decreased by one or three milk units, depending on whether coffee or cappuccino was served. When falling below the highest liquid level needed for a beverage (in this case three milk units) of one of the milk types, the machine passes into the error state. To realize this process, the former milk selection process as well as the milk counter have to be adapted and extended.

Modeling

Higher-order delta `Milk_Type` adds overall 16 new deltas to the delta model and modifies one existing delta in order to adapt the milk choice to the new milk types. `Milk_Type` is depicted in Fig. 3.21, Fig. 3.22, Fig. 3.23 and Fig. 3.24.

`Delta_MilkType` adds states `choose_milk_type` and `milk_type_chosen` as well as transitions `t47`, `t48` and 49 for choosing the milk type to the delta model. `Delta_Pour_MilkType_Tea_Coffee`

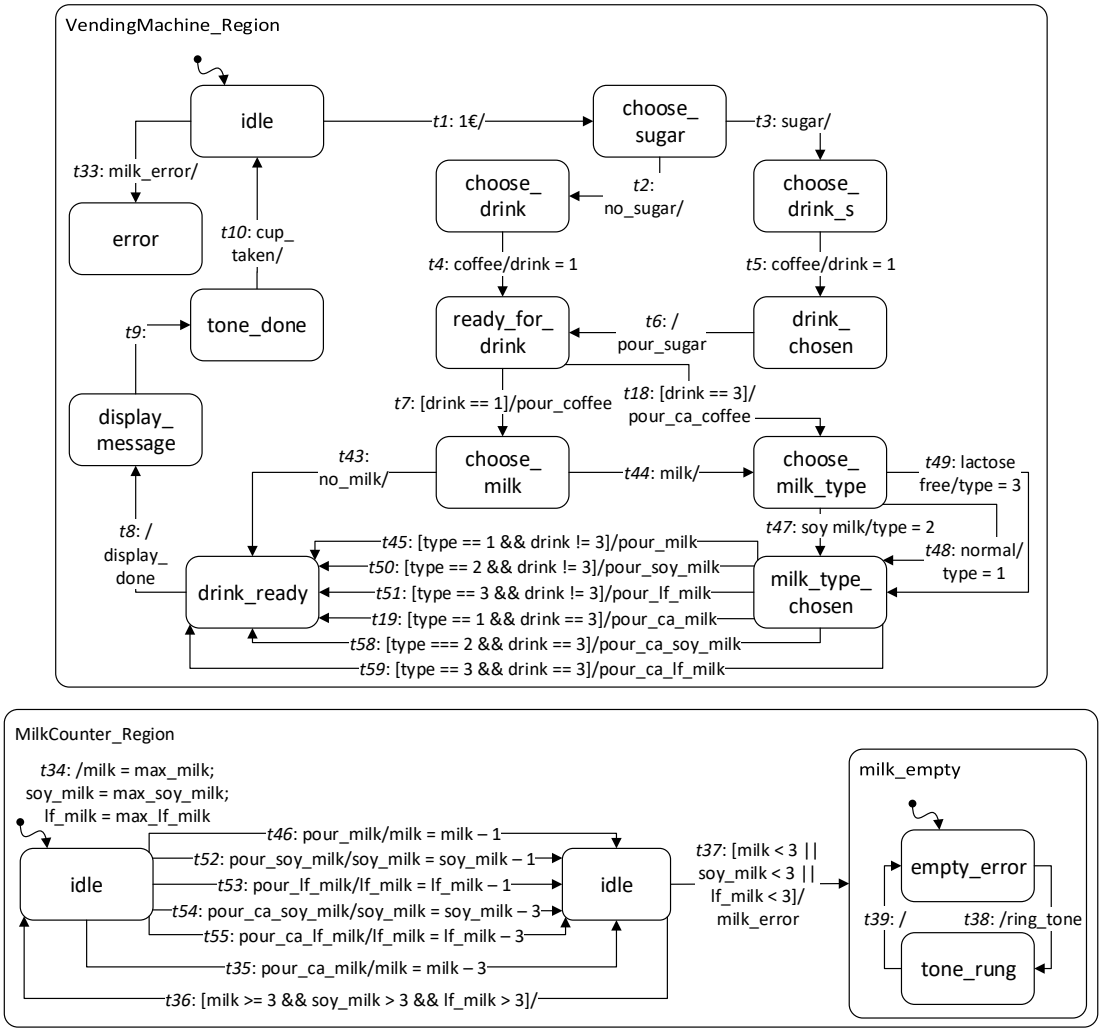

Figure 3.20: Product Model of the Vending Machine with Different Milk Types

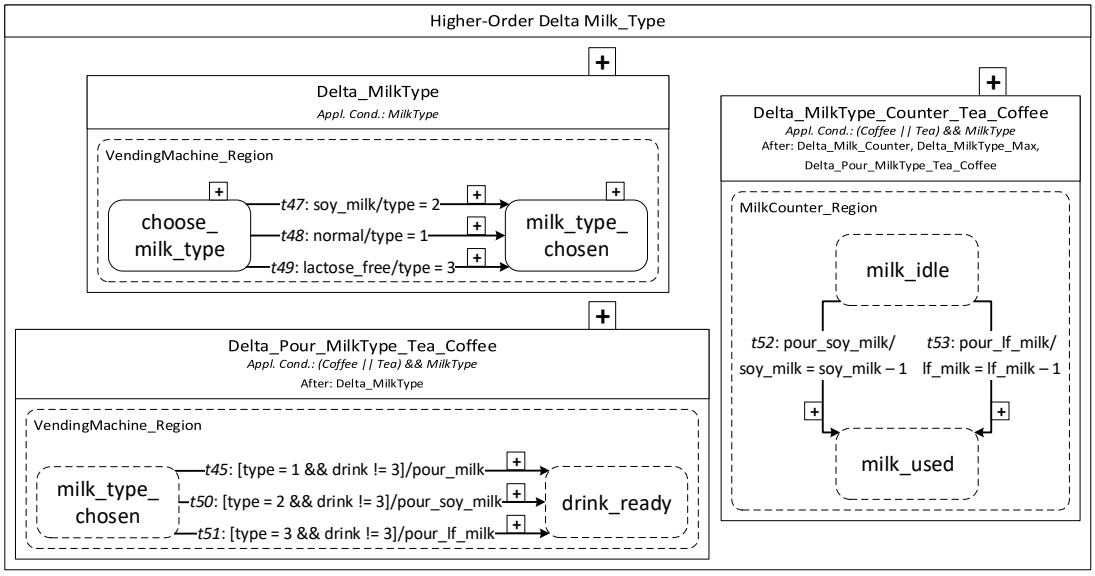

Figure 3.21: First Part of the Higher-Order Delta for Different Milk Types

adds transitions `t45`, `t50` and `t51` for pouring the chosen milk type for tea or coffee. This delta can only be added after `Delta_MilkType`. `Delta_MilkType_Tea_Coffee` adds `t44` between `choose_milk_type` and `choose_milk` and removes state `milk_chosen` as well as the transitions `t44` and `t45`. For this delta, `Delta_Milk` and `Delta_MilkType` have to be applied previously. `Delta_MilkType_Counter_Tea_Coffee` adds transitions `t52` and `t53` for decreasing the counter of lactose-free and soy milk for coffee and tea to the `MilkCounter_Region`. This delta can only be applied after deltas `Delta_Milk_Counter`, `Delta_MilkType_Max`, and `Delta_Pour_MilkType_Tea_Coffee`.

`Delta_MilkType_Counter_Comparison_NoCappuccino`, `Delta_MilkType_Counter_Comparison_Cappuccino`, and `Delta_MilkType_Counter_Comparison_Cappuccino_Size` each modify the transitions `t36` and `t37` in order to compare the liquid levels of lactose-free and soy milk to the particular acceptable low level. `Delta_MilkType_Counter_Comparison_NoCappuccino` can only be applied after `Delta_Milk_Counter_Comparison_NoCappuccino`, `Delta_MilkType_Counter_Comparison_Cappuccino` after `Delta_Milk_Counter_Comparison_Cappuccino`, and `Delta_MilkType_Counter_Comparison_Cappuccino_Size` after `Delta_Milk_Counter_Size`. Additionally, `Delta_MilkType_Max` has to be applied prior to all three deltas.

`Delta_Pour_MilkType_Cappuccino` adds transitions `t58` and `t59` for pouring lactose-free and soy milk for cappuccino and can only be applied after `Delta_MilkType`. `Delta_Pour_MilkType_Cappuccino_Size` adds transitions `t60` and `t61` and modifies transitions `t58` and `t59`. Thus, it can only be applied after `Delta_MilkType` and `Delta_Pour_MilkType_Cappuccino`.

`Delta_MilkType_Counter_Cappuccino` adds transitions `t54` and `t55` for decreasing the value of the milk counter of lactose-free milk and soy milk for regular sized cappuccinos to the model. The same applies for `Delta_MilkType_Counter_Cappuccino_Size` with `t56` and `t57` in case of large cappuccinos. `Delta_Milk_Counter`, `Delta_MilkType_Max`, and `Delta_Pour_MilkType_Cappuccino` have to be applied prior to `Delta_MilkType_Counter_Cappuccino` as well as `Delta_MilkType_Counter_Cappuccino_Size`.

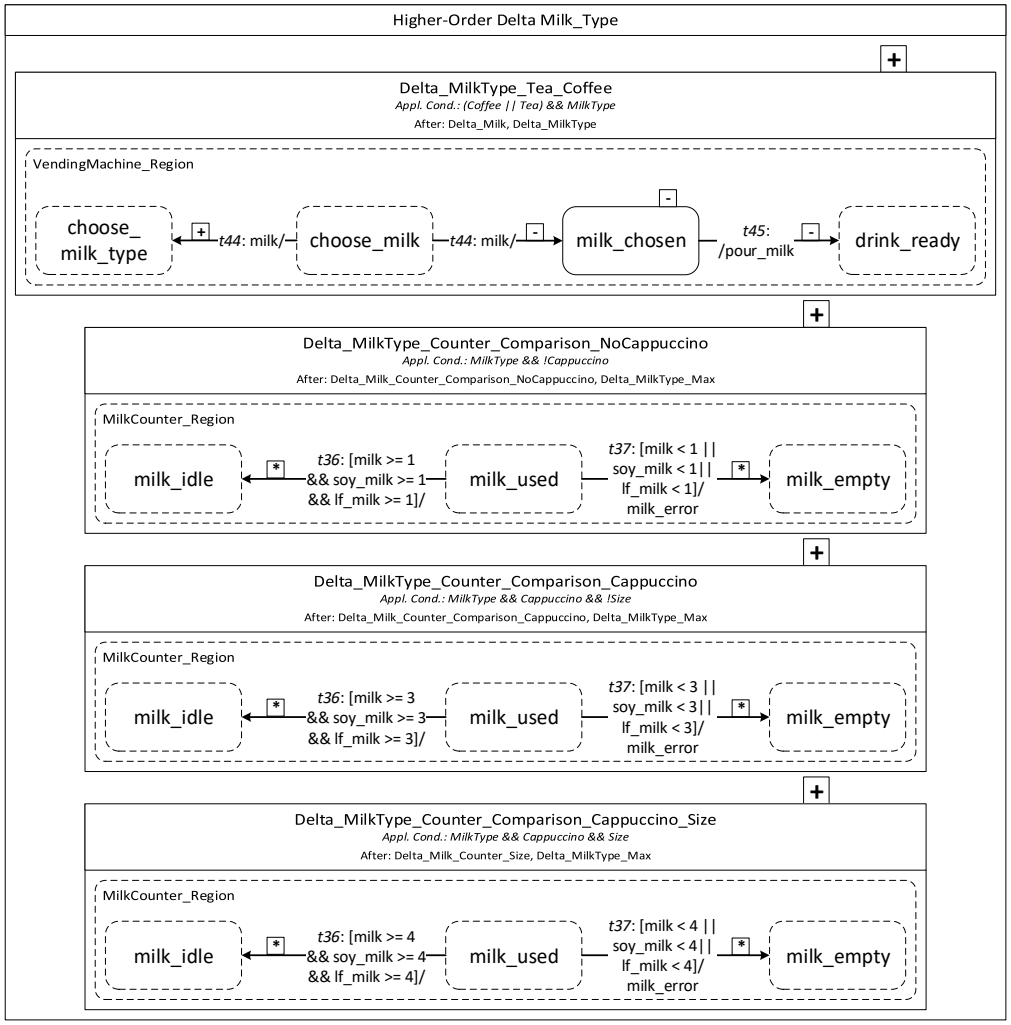

Figure 3.22: Second Part of the Higher-Order Delta for Different Milk Types

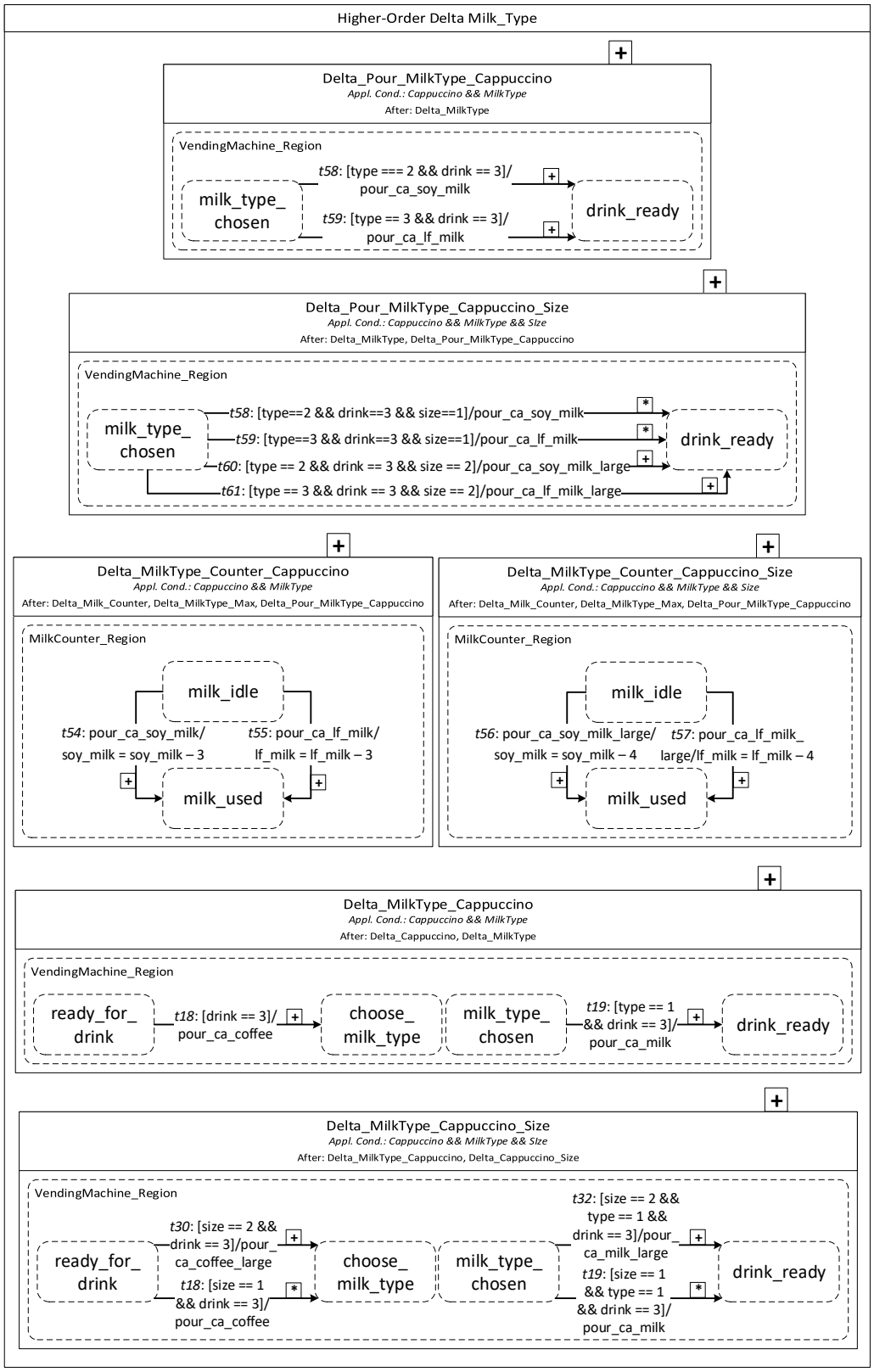

Figure 3.23: Third Part of the Higher-Order Delta for Different Milk Types

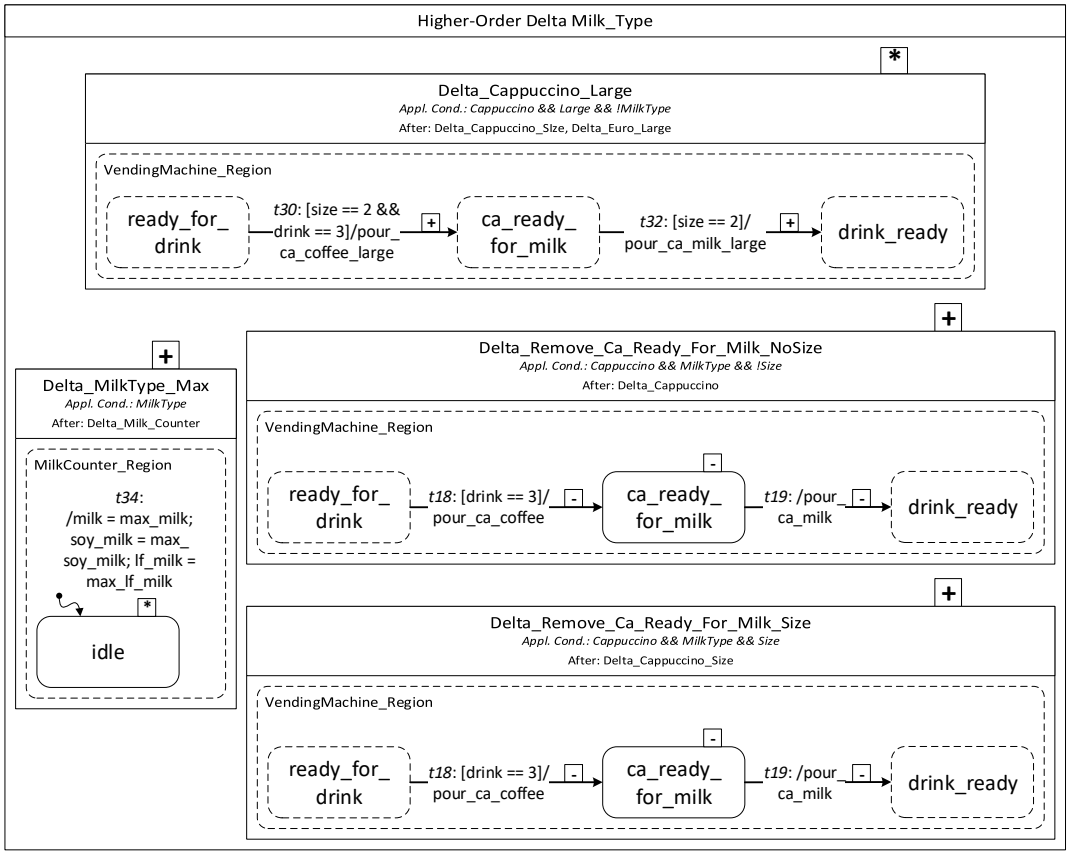

Figure 3.24: Fourth Part of the Higher-Order Delta for Different Milk Types

Delta_MilkType_Cappuccino adds transition t18 between state ready_for_drink and choose_milk_type as well as t19 between milk_type_chosen and drink_ready. Delta_MilkType_Cappuccino can only be applied after the deltas Delta_Cappuccino and Delta_MilkType. Delta_MilkType_Cappuccino_Large adds transitions t30 and t32, and modifies t18 and t19 in order to adapt their transition conditions to the beverage size. Delta_MilkType_Cappuccino and Delta_Cappuccino_Size have to be applied prior to Delta_MilkType_Cappuccino_Large. In order to not add t30 and t32 in two different places, the application condition of Delta_Cappuccino_Large is modified so that Delta_Cappuccino_Large is not selected for application if feature MilkType is selected.

Delta_MilkType_Max modifies t34 with the result that in case of a reboot of the machine lactose-free and soy milk are also restored to their maximum liquid level value. Delta_MilkType_Max can only be applied after Delta_Milk_Counter. Deltas Delta_Remove_Ca_Ready_For_Milk_NoSize and Delta_Remove_Ca_Ready_For_Milk_Size both remove state ca_ready_for_milk and transitions t18 and t19. Both deltas differ only in the fact, that Delta_Remove_Ca_Ready_For_Milk_Size is used when feature Size is selected, whereas Delta_Remove_Ca_Ready_For_Milk_NoSize is used when Size is not selected. Additionally, Delta_Cappuccino_Size has to be applied prior to Delta_Remove_Ca_Ready_For_Milk_Size, whereas Delta_Remove_Ca_Ready_For_Milk_NoSize can only be applied after Delta_Cappuccino.

By extending the vending machine SPL with the offer of different milk types, 42 new feature configurations with respective variant-specific delta sets are added. Hence, the total number of valid product variants is 90, where the 48 already existing variants remain unchanged.

3.2 Wiper System

This section describes the evolution scenarios for the wiper system SPL. In total, there are five scenarios. On the one hand, the system is extended by the detection of medium rain and the addition of different intensity levels for the permanent wiping. On the other hand, a cleaning function is added as well as a check of the liquid level of the screenwash before using the cleaning function and a check of the window temperature prior to any movement of the wiper.

3.2.1 Detection of Medium Rain

This scenario describes the addition of the intensity *Medium Rain* to the recognizable rain intensities.

Description

Initial Situation: The wiper system consists of a sensor and a wiper, that both are offered in high or low quality. Sensor and wiper in either quality can be arbitrarily combined with one another. The high quality sensor recognizes the rain intensities *little* and *heavy*, while the low quality sensor only recognizes rain in general. Additionally, for every combination of sensor and wiper, there is the option to choose a permanent wiping function.

Scenario: Customers are upset about the combination of the high quality sensor and wiper. They say that for little rain the combination works perfectly well, but once the rain gets a little heavier the system takes too long to switch to fast wiping. During this stage, the customers can hardly see anything through the rain, a fact already leading to dangerous situations. As soon as the rain reaches the intensity triggering the fast wiping, the latter is disproportionally fast and annoys more than it

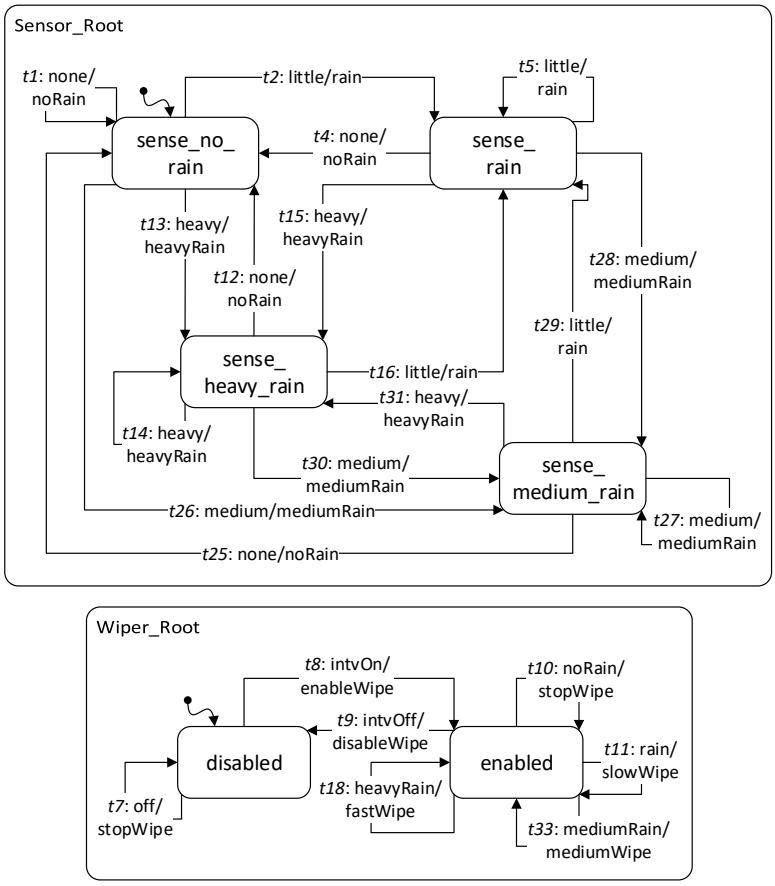

Figure 3.25: Product Model of the Wiper System with the Detection of Medium Rain

helps. In order to meet the requirements of the customers, the manufacturer of the wiper system decides to install a medium intensity with the high quality sensor/wiper combination as standard.
Realization: Since this scenario describes an internal change of the behavior of the sensor and the wiper, there is no change of the feature model of the wiper system product line. Thus, the feature model still matches Fig. 2.5 in Sect. 2.2.

The product state machine shown in Fig. 3.25 represents a combination of a high quality sensor and a high quality wiper. Now, the sensor also recognizes medium rain and contains the needed transitions to switch between the recognition of none, little, or heavy rain and medium rain. The high quality wiper reacts to the event of medium rain with a medium wiping.

Modeling

The higher-order delta `Medium_Rain` adds four new deltas to the delta model that are needed to realize the recognition of medium rain. `Medium_Rain` is shown in Fig. 3.26. `Delta_Medium_LowSensor` is defined for the low quality sensor and adds transitions `t23` and `t24` which translate the recognition of medium rain simply to the event `rain`.

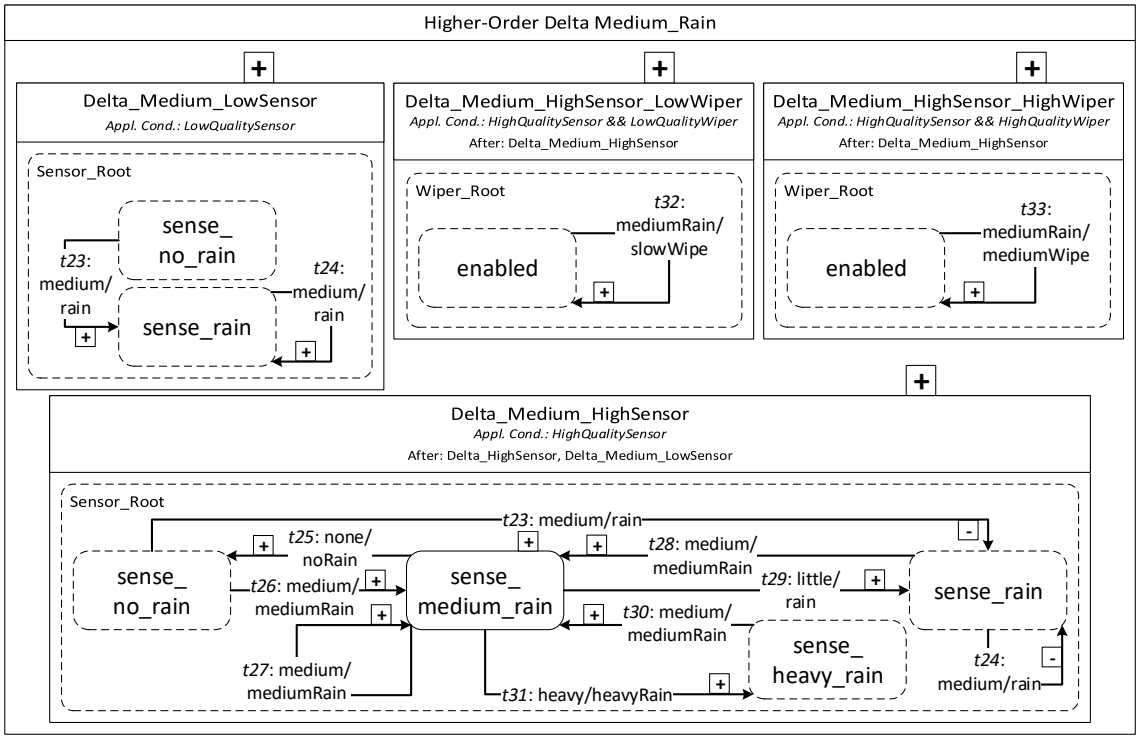

Figure 3.26: Higher-order Delta for Adding the Detection of Medium Rain

`Delta_Medium_HighSensor` adds the state `sense_medium_rain`. Furthermore, the transitions `t25` and `t26` for switching between none and medium rain, `t28` and `t29` for switching between little and medium rain, `t30` and `t31` for switching between heavy and medium as well as `t27` for keeping up medium rain are added. In addition, the transitions `t23` and `t24` are removed.

To adjust the low quality wiper to the signals of the high quality sensor, `Delta_Medium_HighSensor_LowWiper` adds transition `t32` for reacting to medium rain with slow wiping. For the high

quality wiper, `Delta_Medium_HighSensor_HighWiper` adds transition `t33` for triggering medium wiping in case of medium rain.

The set of possible feature configurations does not get changed by this evolution scenario. However, all eight existing product variants are modified due to the introduction of the new deltas w.r.t. the medium rain intensity. Thus, there results a total number of eight product variants for the wiper system SPL.

3.2.2 Intensities for Permanent Wiping

In this scenario, the permanent wiping function is extended by three intensity levels.

Description

Initial Situation: A wiping system can arbitrarily be composed by a high or low quality wiper and a high or low quality sensor. For each combination, a permanent wiping function can be selected.
Scenario: Customers increasingly complain that the permanent wiping is too fast for little rain, but too slow for heavy rain. Thus, the manufacturers decide to offer an option of choosing intensity levels for the permanent wiping function. Three new levels are introduced, between which the customer can switch manually.
Realization: The feature model for this scenario is depicted in Fig. 3.27. It captures the addition of feature `Intensity` as optional child-feature of feature `Permanent`. Thus, in case the permanent wiping function gets selected, a customer now can also choose if it is supplied with one intensity level or three.

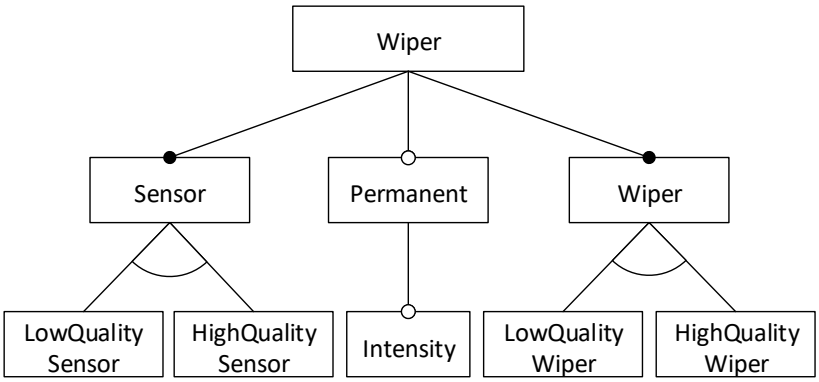

Figure 3.27: Feature Model of the Wiper System SPL with Intensities for Permanent Wiping

A product state machine containing the new intensities for permanent wiping is shown in Fig. 3.28. In the model, a new substate machine dealing with the management and setting of the three new intensities is added to the state machine. In the substate machine of the wiper, the permanent wiping is automatically executed in the previously set intensity after activation. Additionally, the intensity levels can be changed arbitrarily during permanent wiping. There are no changes to the sensor. Thus, its substate machine is not depicted again for the current product state machine.

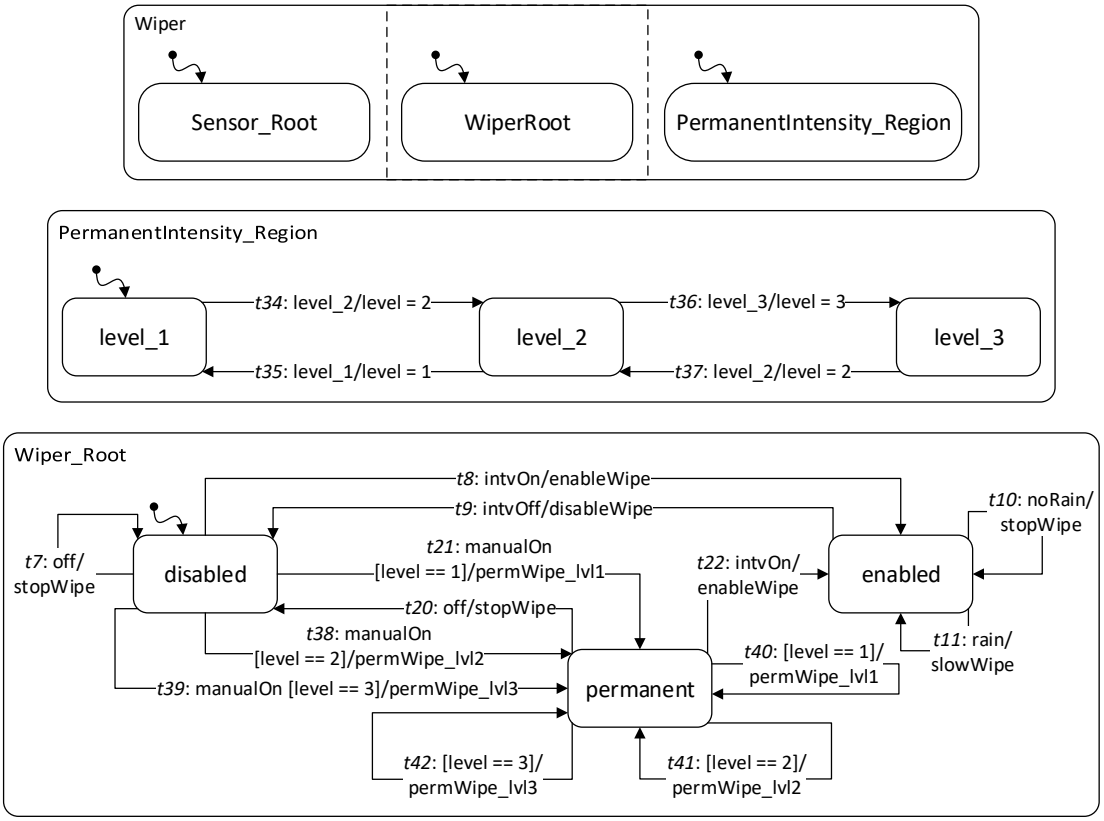

Figure 3.28: Product Model of the Wiper System with Intensities for Permanent Wiping

Modeling

The higher-order delta `Permanent_Intensity` required for this evolution step is shown in Fig. 3.29. `Permanent_Intensity` adds the two new deltas `Delta_Intensity` and `Delta_Permanent_Intensity` to the delta model. `Delta_Intensity` adds the substate machine `PermanentIntensity_Region` to state machine `Wiper`. `PermanentIntensity_Region` consists of the three states `level_1`, `level_2`, and `level_3` as well as the transitions `t34`, `t35`, `t36`, and `t37`, responsible for switching between the intensity levels.

`Delta_Permanent_Intensity` modifies transition `t21`. Thus, `t21` now triggers permanent wiping at level 1 for set level 1. Additionally, transitions `t38` and `t39` are added for a similar handling of levels 2 and 3. Furthermore, transitions `t40`, `t41`, and `t42` are added in order to react to a switching command during the permanent wiping. `Delta_Permanent_Intensity` can only be applied after `Delta_Permanent_Wiping`.

Because of the addition of the intensities for permanent wiping, four new feature configurations with respective variant-specific delta sets are added. In total, there exist twelve product variants, where eight variants remain unchanged.

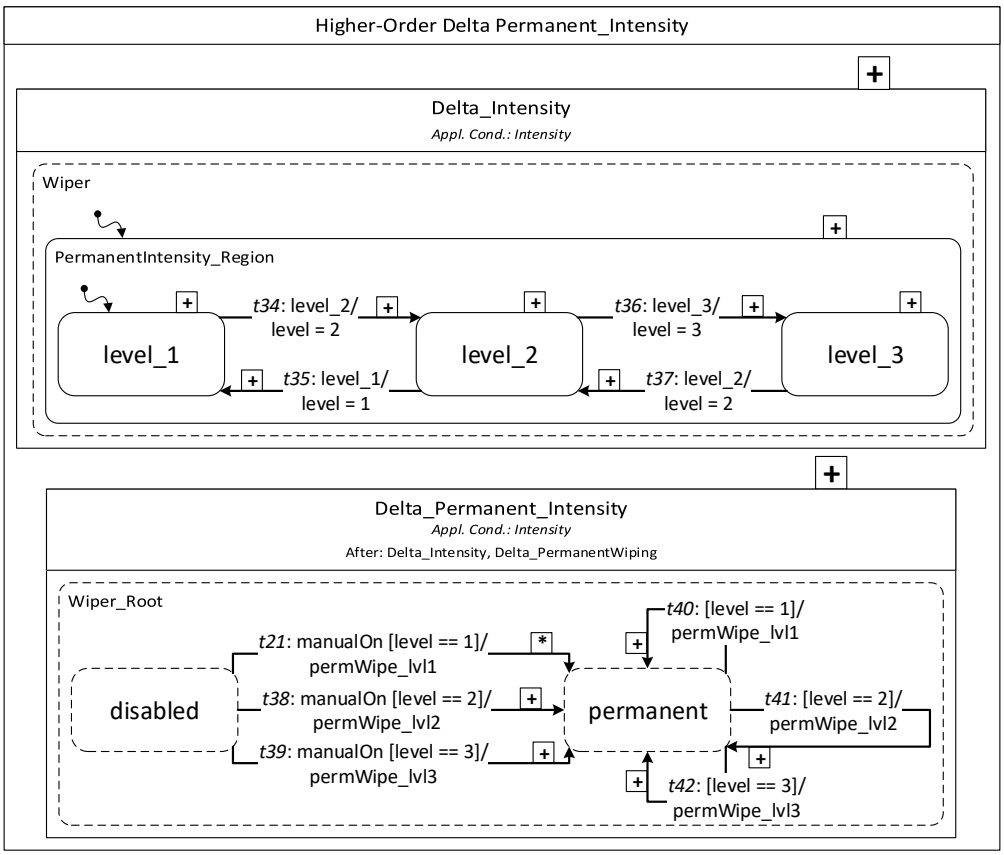

Figure 3.29: Higher-Order Delta for Adding intensities for Permanent Wiping

3.2.3 Window Cleaning Function

This scenario adds an optional cleaning function to the product line of the wiper system.

Description

Initial Situation: The wiper system SPL consists of a sensor and a wiper which exist in high or low quality and can be combined arbitrarily. Furthermore, it can contain a permanent wiping function, that can come with three intensities.
Scenario: In the past, the customers are tired of regularly stopping at the service station in order to remove dirt from the windshield. Thus, many customers decide with increased regularity to buy the wiper system of competitors. To counter the loss of customers, the wiper system manufacturer decides to also offer a cleaning function for his wiper system.
Realization: The cleaning function is represented in the feature diagram by the new optional feature `Clean`. The feature diagram is shown in Fig. 3.30.

The function of this feature is to spray screenwash and wipe three times, regardless which state (automatic, permanent or off) the system was in before, and afterwards return to this state. The product state machine depicted in Fig. 3.31 represents the behavior of a low quality wiper with permanent wiping function for the new feature. The sensor is not changed for this scenario and, thus, is not shown additionally.

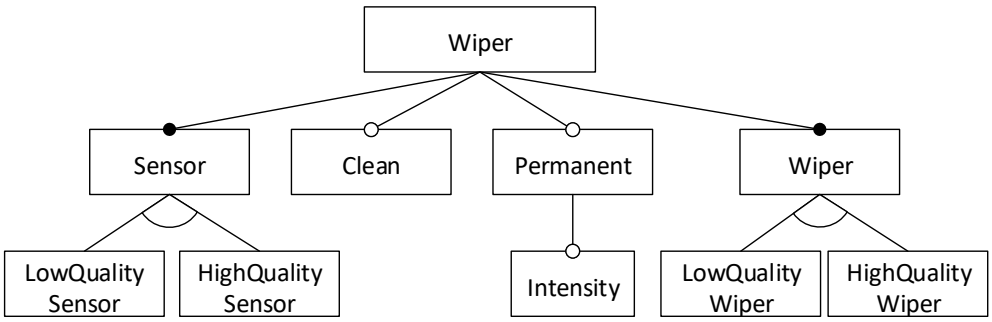

Figure 3.30: Feature Model of the Wiper System SPL with Window Cleaning Function

Modeling

In order to realize the cleaning function, the higher-order delta `Clean` shown in Fig. 3.32 adds three new deltas to the delta model. `Delta_Clean` adds the states `clean` and `used_screenwash` as well as the transitions `t43`, `t44`, `t45`, and `t46` for activating the cleaning function when the system is in the states `disabled` or `enabled` and returning to these states after the cleaning. Furthermore, the transitions `t47` and `t48` for spraying screenwash and wiping three times are added.

`Delta_Clean_Permanent` is used when additionally to feature `Clean` the feature `Permanent` is selected. In this case, the transitions `t49` and `t50` for activating the cleaning function and reactivating the permanent wiping after the cleaning are added. Prior to `Delta_Clean_Permanent`, the deltas `Delta_Clean` and `Delta_PermanentWiping` have to be applied.

If feature `Intensity` is also selected, `Delta_Permanent_Intensity` is used to modify transition `t50` such that it now activates level 1 of the permanent wiping, when the corresponding intensity is set. Likewise the transitions `t51` and `t52` are added to cover the same behavior for levels 2 and 3. `Delta_Permanent_Intensity` can only be applied after `Delta_Clean_Permanent`.

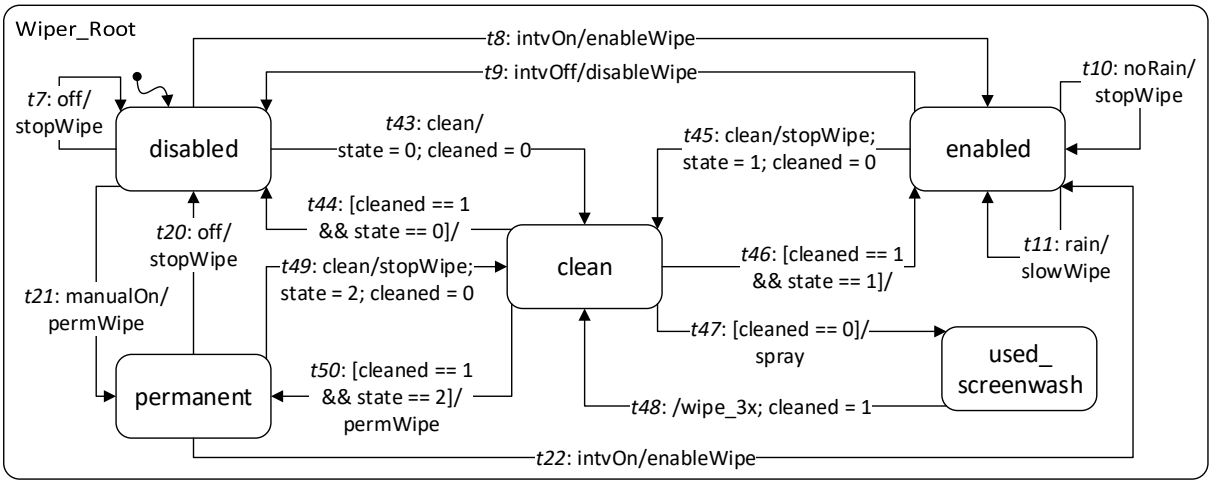

Figure 3.31: Product Model of the Wiper System with Window Cleaning Function

This scenario adds twelve feature configurations with respective variant-specific delta sets to the SPL of the wiper system. As a result the total number accounts for 24 product variants, where twelve variants remain unchanged.

3.2.4 Screenwash Liquid Level Check

In this scenario, an automatic checking of the screenwash liquid level is added.

Description

Initial Situation: The wiper system is composed of an arbitrary combination of a low or high quality sensor and wiper and can comprise a permanent wiping function which can optionally be adjustable in three intensities. Additionally, a cleaning function can be selected if desired.
Scenario: More and more often the profile rubber of the wipers is damaged as a result of using the cleaning function even though there is not enough screenwash left. Thus, there is an unwanted wiping on the dry windscreen. To counter this effect, a function automatically checking the screenwash liquid level before cleaning with the cleaning function will always be supplied from now on. However, the screenwash can only be topped up by the service.
Realization: Since the new function shall be supplied as a standard with each wiper system featuring the cleaning function, the function is realized directly in feature `Clean`. Thus, there is no change to the feature model depicted in Fig. 3.30.

Figure 3.33 shows the product model of a low level wiper system with the permanent wiping function and the revised cleaning function. In order to realize the checking of the screenwash liquid level, a new substate machine is added. It counts down by one screenwash unit for each spraying of the cleaning function and displays a warning in case of an empty liquid level. Additionally, for each attempt to activate the cleaning function the screenwash liquid level gets checked. In case there is not enough screenwash left, the transition to the state `clean` is blocked. For this scenario, there is also no change to the sensor, so that its state machine is not shown again.

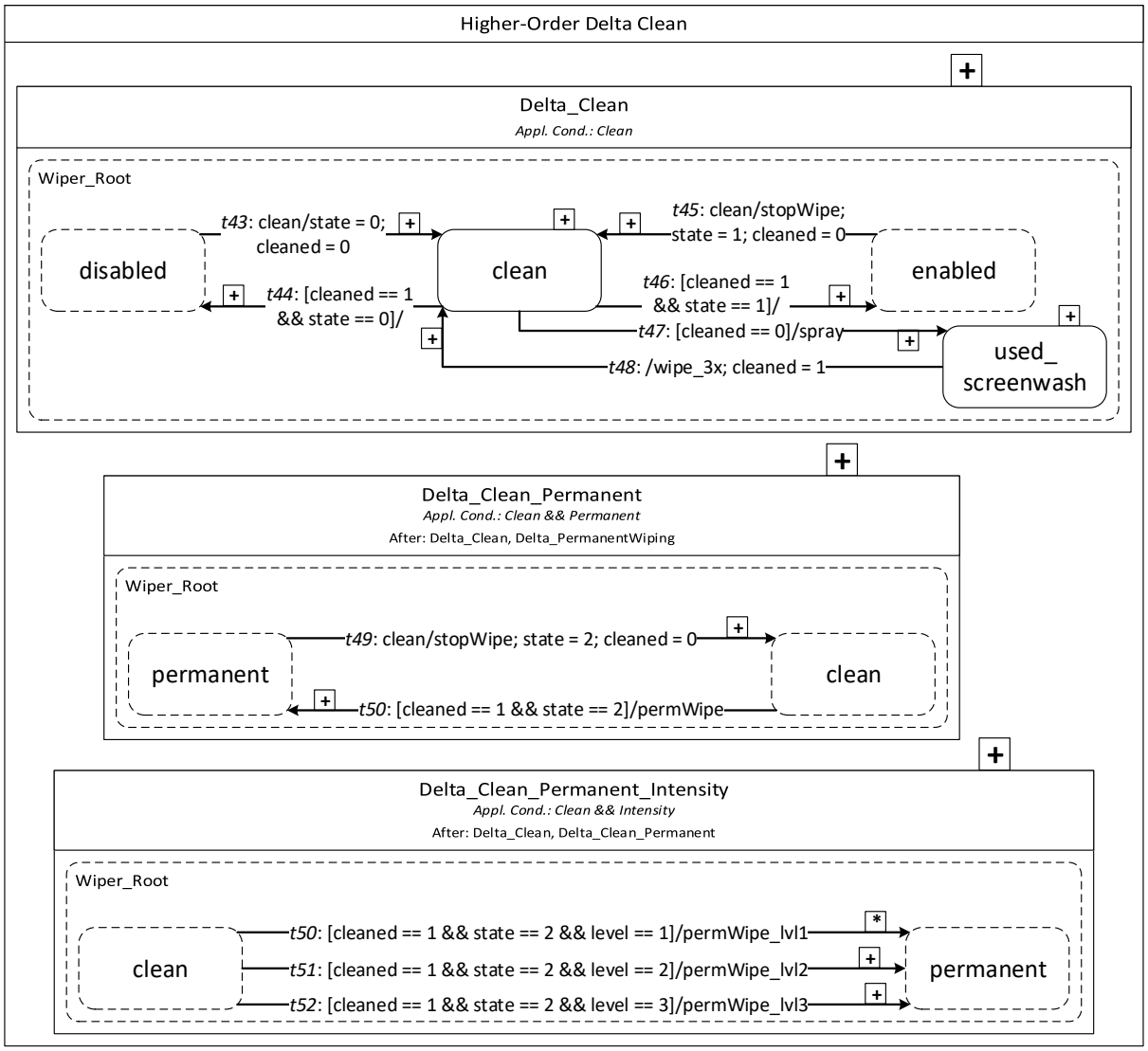

Figure 3.32: Higher-Order Delta for Adding the Window Cleaning Function

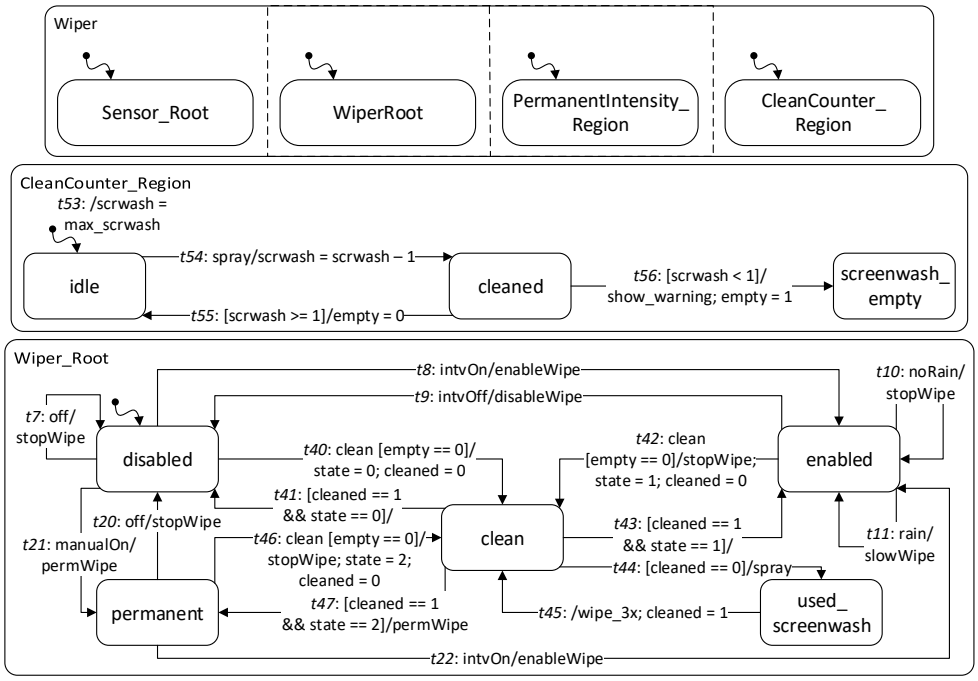

Figure 3.33: Product Model of the Wiper System with Screenwash Liquid Level Check

Modeling

The higher-order delta `Clean_Counter` depicted in Fig. 3.34 adds a new delta and modifies two existing deltas. The new delta `Delta_Clean_Counter` adds the new substate machine `CleanCounter_Region` to the wiper system. This substate machine is composed of the states `idle`, `cleaned`, and `screenwash_empty` as well as the transitions `t53` for initializing the liquid level, `t54` for decreasing the counter when spraying, `t55` for returning to the idle state in case of a sufficient liquid level, and `t56` for displaying the warning in case of a too low liquid level.

`Delta_Clean` is modified so that the labels of the transitions `t43` and `t45` now contain the condition, that the screenwash liquid level must not be empty. Furthermore, `Delta_Clean_Permanent` is modified so that the same condition is added to transition `t49`.

The set of the feature configurations is not changed based on this scenario, but the feature `Clean` is extended internally. Thus, the twelve product state machines containing the cleaning function are modified accordingly. In total, there are still 24 product variants as a consequence, where twelve variants remain unchanged.

3.2.5 Frost Check

This scenario adds an automatic checking of the window temperature prior to any movement of the wiper.

Description

Initial Situation: A wiper system consists of a sensor and a wiper, each in either high or low quality. Furthermore, it can have a permanent wiping function, optionally with different intensities, and a cleaning function with a screenwash liquid level check.

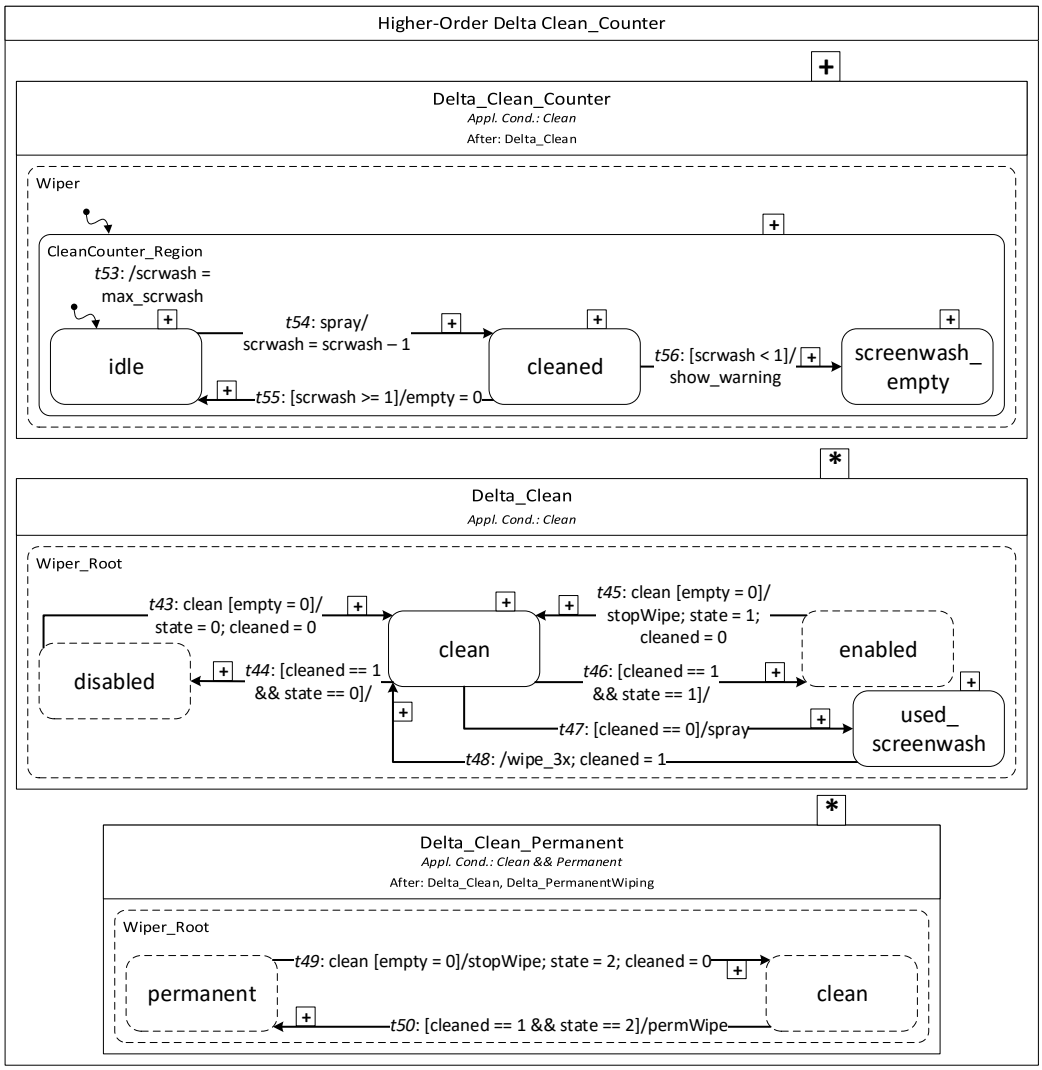

Figure 3.34: Higher-Order Delta for Adding the Screenwash Liquid Level Check

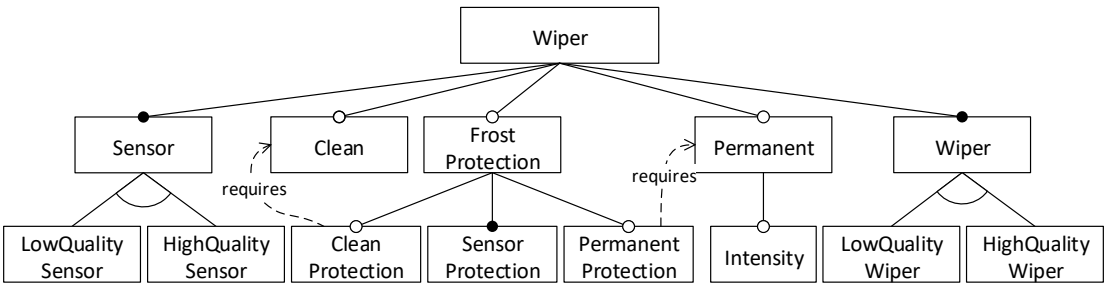

Figure 3.35: Feature Model of the wiper system SPL with Frost Check

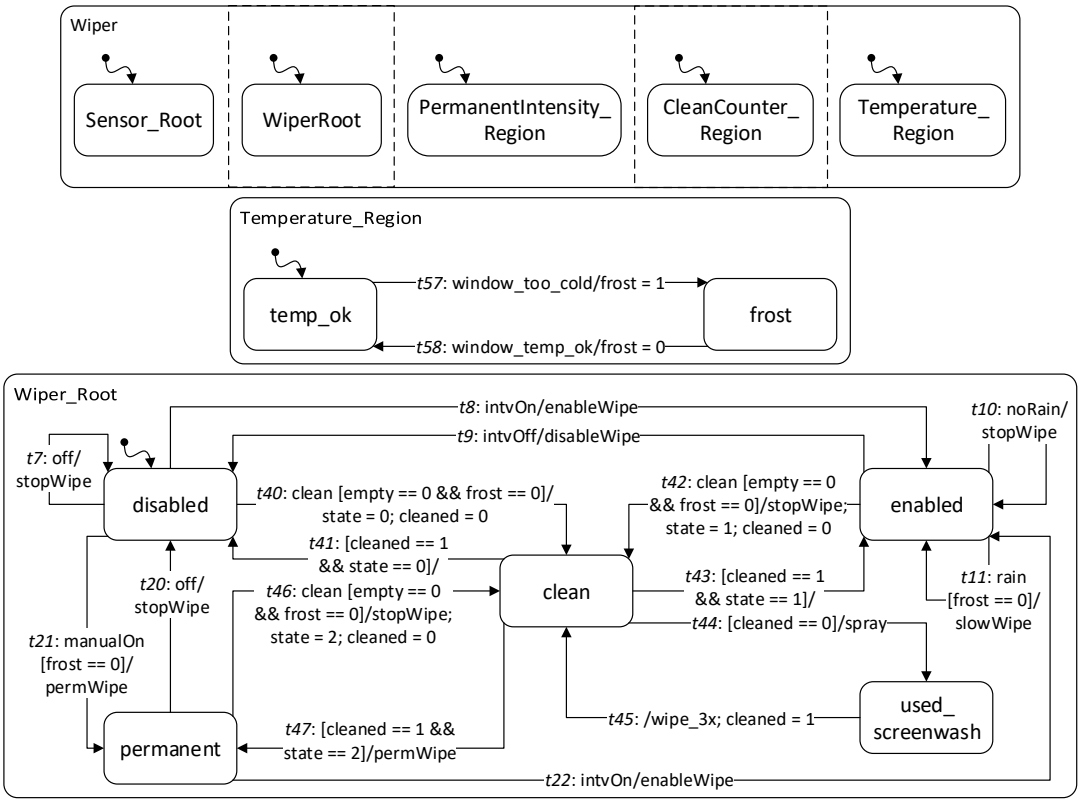

Figure 3.36: Product Model of the Wiper System with Frost Check

Scenario: Winter is coming and temperatures are reaching degrees below zero already in some regions. In these regions, damages to the motors of the wipers occur more and more often, since the wipers are still frozen to the windows because of the temperatures, when the automatic wiping starts or the permanent wiping, or alternatively the cleaning function, are activated. To prevent those damages, in the future there shall be a possibility to choose a frost check.
Realization: The feature diagramm is extended by the new feature `FrostProtection` which is shown in Fig. 3.35. `FrostProtection` possesses the three child features `CleanProtection`, `SensorProtection`, and `PermanentProtection`. `SensorProtection` is a mandatory feature and has to be selected when `FrostProtection` is selected. `CleanProtection` and `PermanentProtection` are optional features. If `CleanProtection` is selected, feature `Clean` has to be selected as well, indicated by the `requires`-relation. The same holds for the features `PermanentProtection` and `Permanent`.

The product state machine for a low quality wiper with permanent wiping function, window cleaning function and the new frost check is depicted in Fig. 3.36. For the frost check, a new substate machine is added. If the temperature is too low, this substate machine passes into a state indicating that it is freezing and, thus, the wiper may be frozen to the window. The wiper uses this informationen and checks prior to each wipe-triggering command (automatic wiping, permanent wiping or cleaning function) if there is frost and, in case there is, prevents the movement of the wiper.

Modeling

For the frost check, eleven new deltas have to be added to the delta model by higher-order delta `Frost_Protection`. This delta is shown in Fig. 3.37 and Fig. 3.38.

`Delta_Temperature` adds the new substate machine `Temperature_Region` together with the states `temp_ok` and `frost` as well as the transitions `t57` for a too low window temperature and `t58` for a not freezing window temperature. `Delta_FP` modifies for each sensor-wiper-combination except `LowQualitySensor` with `HighQualityWiper` the transition `t11`, so that it only starts wiping in case of no frost. `Delta_HighSensor_LowWiper_FP`, `Delta_HighSensor_HighWiper_FP`, `Delta_LowSensor_HighWiper_FP`, `Delta_Medium_HighSensor_LowWiper_FP`, and `Delta_Medium_HighSensor_HighWiper_FP` make the same modification for the transitions `t17`, `t18`, `t19`, `t32`, and `t33`. Those deltas can only be applied after the respective deltas adding the transitions to be modified.

For the selected feature `CleanProtection`, `Delta_Clean_FP` adapts the conditions of the transitions `t43` and `t45` to the frost check. This delta can not be applied prior to `Delta_Clean`. In case feature `CleanProtection` and feature `Permanent` are both selected, `Delta_Clean_Permanent_FP` is applied. It modifies transition `t49` so that the condition of the transition also contains the frost check. Prior to using `Delta_Clean_Permanent_FP`, `Delta_Clean_Permanent` and `Delta_Temperature` has to be applied.

In the same manner for selected feature `PermanentProtection`, `Delta_PermanentWiping_FP` modifies transition `t21` if feature `Intensity` is not selected as well as `Delta_Permanent_Intensity_FP` modifies the transition if feature `Intensity` is selected. Additionally, `Delta_Permanent_Intensity_FP` modifies transitions `t38` and `t39` so that they also contain the condition. `Delta_PermanentWiping_FP` and `Delta_Permanent_Intensity_FP` can only be applied after `Delta_PermanentWiping`, or alternatively `Delta_Permanent_Intensity`, and `Delta_Temperature`.

The addition of the frost check adds 60 new feature configurations with respective variant-specific delta sets. Hence, the total number of valid product variants for the wiper system SPL amounts to

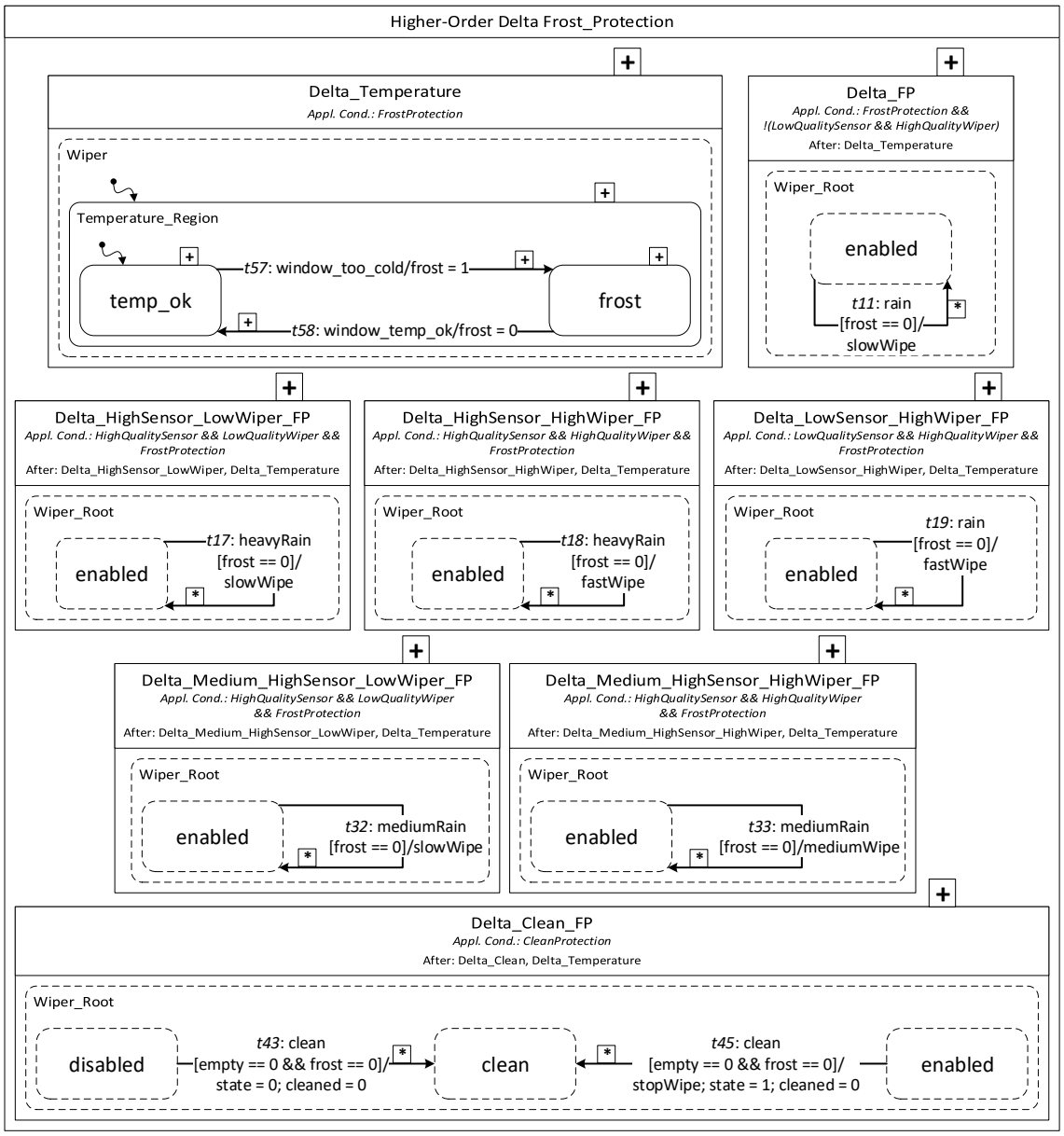

Figure 3.37: First Part of the Higher-Order Delta for Adding the Frost Check

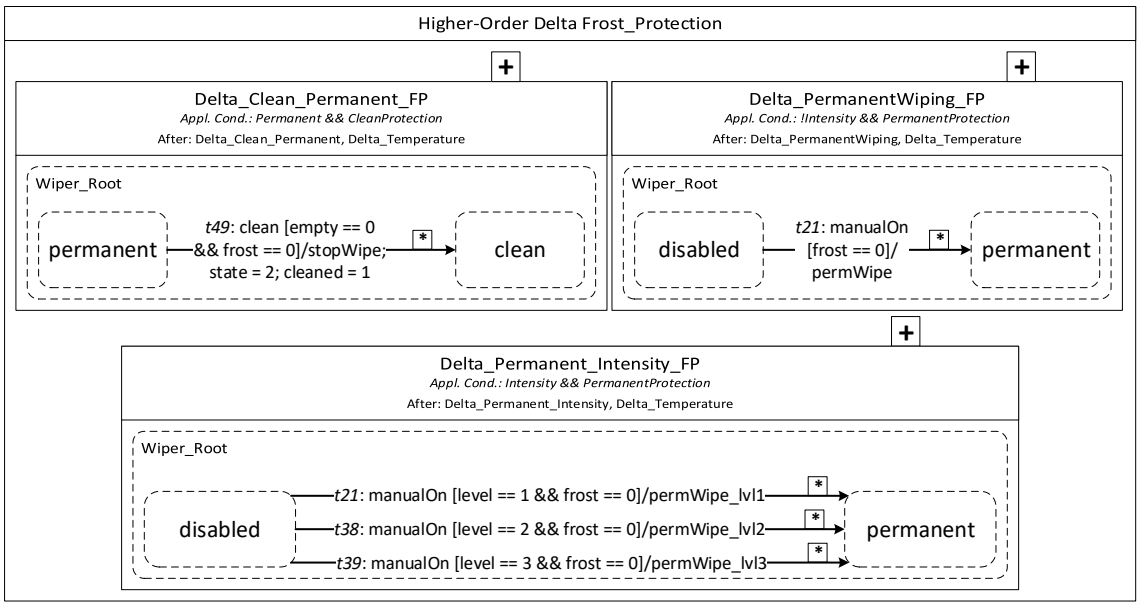

Figure 3.38: Second Part of the Higher-Order Delta for Adding the Frost Check

84, where 24 variants remain unchanged.

3.3 Mine Pump System

In this section, the evolution scenarios for the mine pump system SPL are described. Three scenarios are presented including the extension of the system by a methane extraction, the addition of an air check, and the addition of an air exchange mechanism.

3.3.1 Methane Extraction

In this scenario, an automatic methane extraction is added to the product line of the mine pump system.

Description

Initial Situation: A mine pump system consists of a water regulation for monitoring the water level. The water regulation automatically pumps out water in case of a high water level and can also optionally react to low or normal water level. Furthermore, the mine pump system can have a methane detection stopping the pump or preventing an automatic start of the pump once a critical threshold of methane is passed. Additionally, the mine pump system can optionally come with a command function for manually starting and stopping the pumping function.

Scenario: During the last months, many mine shafts using mine pump systems with methane detection flooded. This is due to the fact, that the pumps were stopped once methane was detected. Because of a missing possibility to quickly and effectively lead away the methane, the water rose too high before the pumps could be activated again. Therefore, the mine pump system manufacturers want to extend their system. They want to standardly provide a possibility to automatically extract the methane to prolong the stopping of the pump or to anticipate the restarting of the pump.

Realization: The methane extraction shall be contained obligatory in each mine pump system with

methane detection. Therefore, there is no new feature added to the feature model, but rather the existing feature `MethaneDetection` is extended in its function. The feature model for this scenario is not changed compared to the feature model shown in Fig. 2.8 in Sect. 2.3.

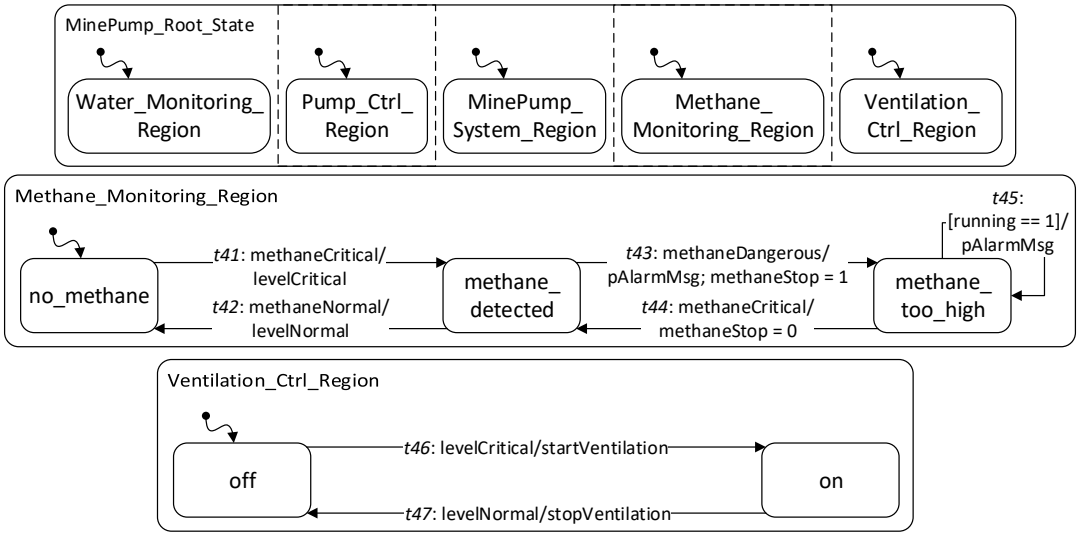

Figure 3.39: Product Model of the Mine Pump System with Methane Extraction

Figure 3.39 shows a possible product state machine with methane extraction. The product model has been extended by a substate machine starting or stopping the extraction unit according to the methane content in the air. The previous methane detection was changed such that it now works with three levels. If the methane content rises to a critical level, the extraction unit is activated, and pumping the water is still possible. Once the methane content reaches a dangerous level despite the extraction, the pump is stopped or alternatively its starting is prevented. When the methane content is dropping under the critical level, the pump can be used normally again, and when reaching a normal methane level, the extraction unit is turned off. There is no change to the other substate machines of the mine pump system and, thus, they are not shown again for the current product state machine.

Modeling

Since the methane extraction is contained in the mandatory feature `MethaneDetection`, the delta `Delta_Add_Methane_Monitoring_Region` is not needed anymore in its previous version. Thus, it is modified by the higher-order delta `Methane_Extraction` depicted in Fig. 3.40. Compared to the prior version of the delta, the new state `methane_too_high` is added and transitions `t23`, `t24`, and `t25` are replaced by the transitions `t41` to `t45`. Transition `t41` signals a critical methane level, `t43` triggers the alarm in case the methane level is dangerously high, `t45` triggers the alarm anew in case the pump is still running despite the previous alarm, `t44` allows the activation of the pump again, and `t42` signals a normal methane level.

Furthermore, the higher-order delta adds `Delta_Add_Ventilation_Ctrl_Region` to the delta model. This delta adds the substate machine `Ventilation_Control_Region` containing states `on` and `off` as well as transitions `t46` for starting the extraction in case of critical methane level and

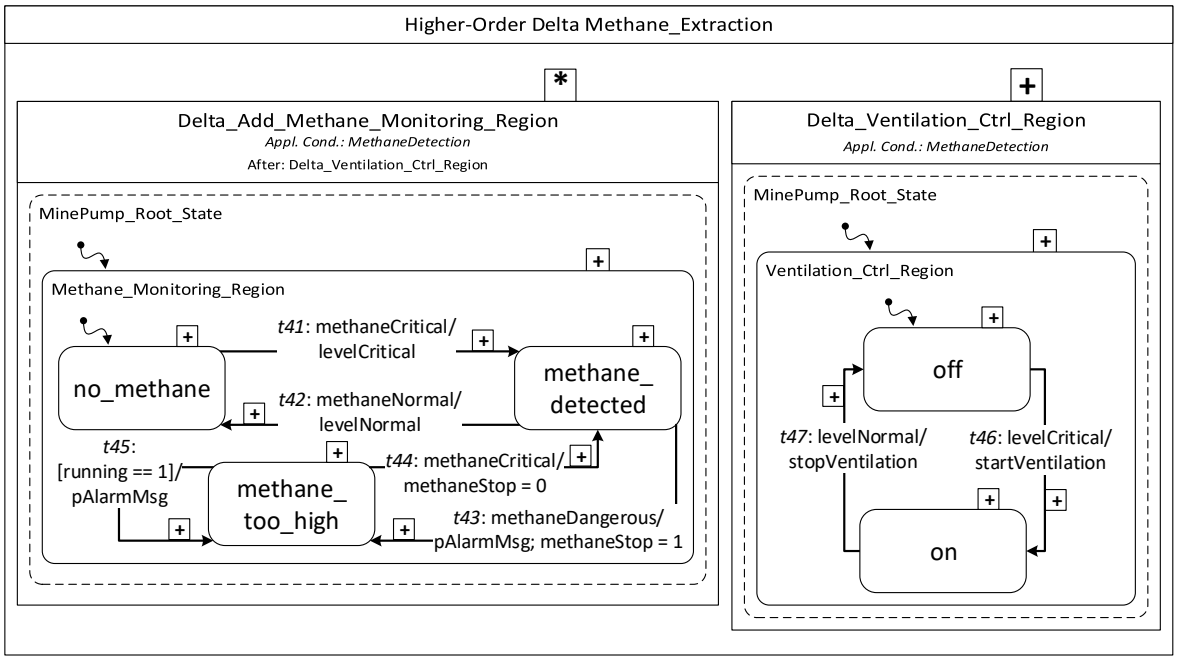

Figure 3.40: Higher-Order Delta for Adding the Methane Extraction

`t47` for stopping the extraction in case of normal methane level.

Due to the scenario, the set of feature configurations is not changed. But since the function of the feature `MethaneDetection` is extended, the eight product variants containing the methane detection are modified. Thus, in total, there are still 16 product variants for the mine pump system SPL, where eight variants remain unchanged.

3.3.2 Air Check

This scenario adds an air check monitoring the atmospheric conditions of the mine pump system SPL.

Description

Initial Situation: Each mine pump system contains a water regulation which monitors the water level and, in case of high water, automatically pumps out water until a certain time has elapsed or, if selected, a low water level is detected. If requested, the system can be equipped with a command function and a methane detection along with a methane extraction.

Scenario: Multiple miners were hospitalized with carbon monoxide poisoning or oxygen deficiencies after the mine pump systems had been running for several hours at a time. Due to the cramped shafts, the bad air ventilation and the exhaust gases emitted by the combustion engine of the pump, oxygen was displaced and carbon monoxide massed in the shafts. In order to prevent such dangers, the mine pump systems shall be equipped with an air check. The air check monitors the atmospheric conditions and sends a warning as well as stops the pump, if it detects a low oxygen level or high carbon monoxide level.

Realization: To realize this function, the optional feature `AirCheck` is added to the feature model

shown in Fig. 3.41.

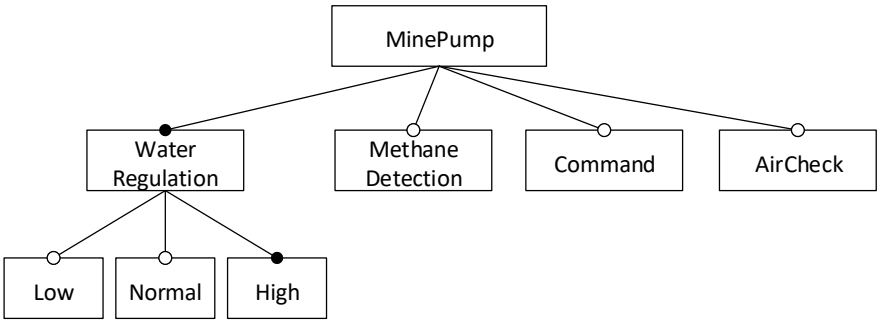

Figure 3.41: Feature Model of the Mine Pump System SPL with Air Check

Figure 3.42 depicts the product state machine for a feature configuration containing features `WaterRegulation` with `High` as well as `MethaneDetection` and `AirCheck`. A new substate machine for the monitoring of the atmospheric conditions was added to the model. This substate machine triggers an alarm and prevents the use of the water pump in case the air is polluted or does not contain enough oxygen. Additionally, `Methane_Level_Control_Region` was replaced by `Alarm_Control_ Region` with the result, that this substate machine is able to react to alarms triggered by methane as well as alarms triggered by air pollution. The `Water_Level_Control_Region` is also adapted. As a result, the pump can not be started despite a high water level if the methane level is too high or the air is polluted.

Modeling

In Fig. 3.43, Fig. 3.44, Fig. 3.45, and Fig. 3.46 the deltas being added, removed or modified by higher-order delta `Air_Check` are shown for this evolution step. `Delta_Add_Air_Monitoring_Region` adds `Air_Monitoring_Region` to the delta model. This substate machine consists of the states `air_clean` and `air_polluted` as well as transitions `t48` for triggering the alarm and blocking the pump in case of polluted air, `t49` for unblocking the pump in case of normal air, and `t50` for triggering the alarm again if the first attempt failed in blocking the pump. `Delta_Add_Methane_Handling_MinePump_System` is removed from the delta model. As a result, the substate machine `Methane_Level_Control_Region` is not part of any product state machine anymore.

Instead, the new delta `Delta_Alarm_Level_Control_Region` depicted in Fig. 3.44 adds `Alarm_Control_Region` containing states `alarm_ctrl_idle`, `alarm_detected_minepump_region` and `pump_stopped_by_alarm` which can be used for the selected feature `MethaneDetection` as well as the feature `AirCheck`. If `AirCheck` is selected, `Delta_Air_Alarm_Msg` adds transition `t51` for triggering the alarm in case of polluted air to substate machine `Alarm_Control_Region`, and if `MethaneDetection` is selected, `Delta_Methane_Alarm_Msg` adds transition `t27` for the methane alarm. Transitions `t28` for stopping the running pump and `t29` for skipping an action in case the pump is not running are added by `Delta_Alarm_Pump_Stop` if any of the two features is selected.

In case only `MethaneDetection` is selected, but not `AirCheck`, `Delta_Methane_Handling` adds transition `t30` for returning to the initial state if the pump is not running and also blocked because of a high methane level. In contrast, if feature `AirCheck` is selected, but not `MethaneDetection`,

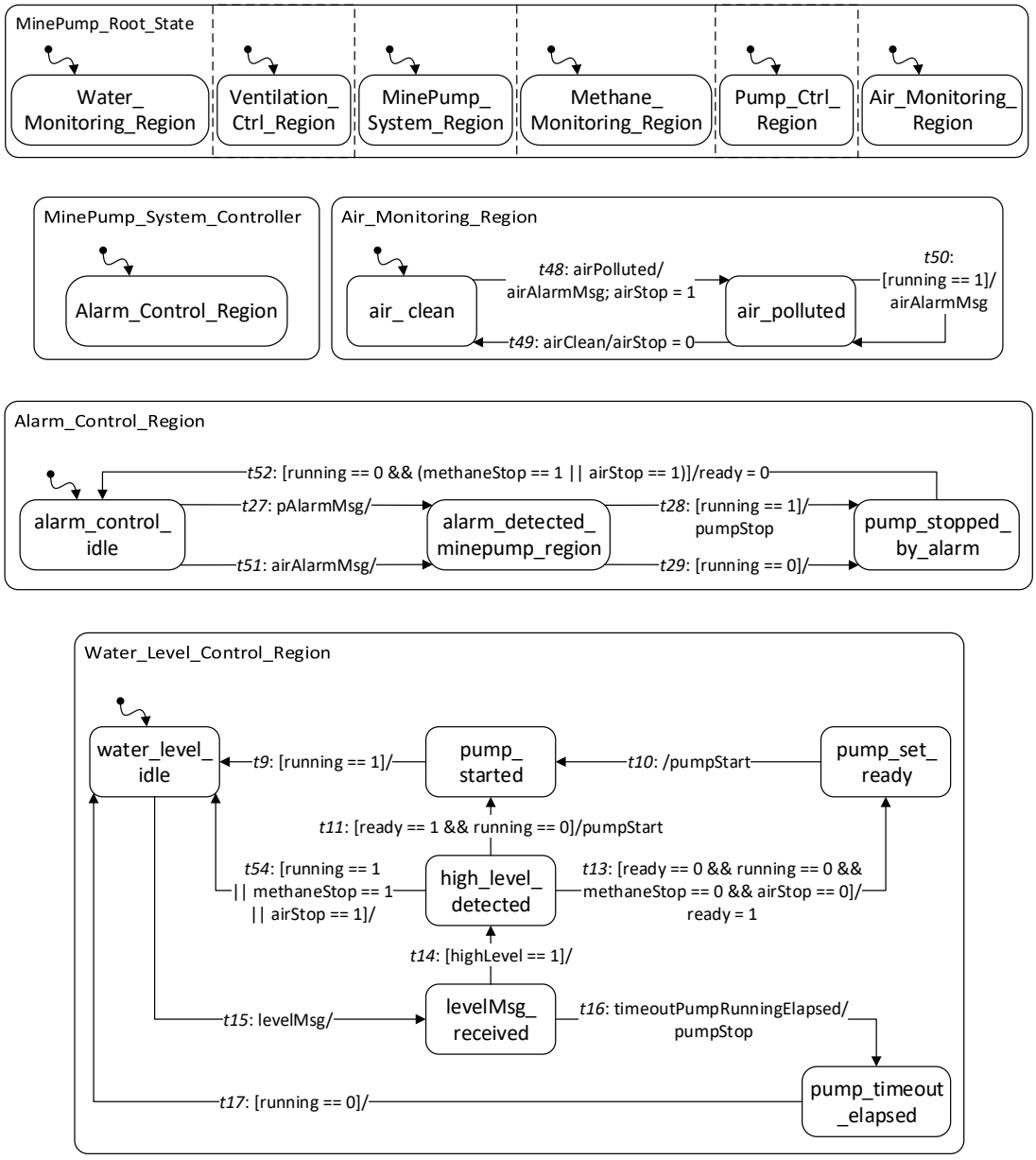

Figure 3.42: Product Model of the Mine Pump System with Air Check

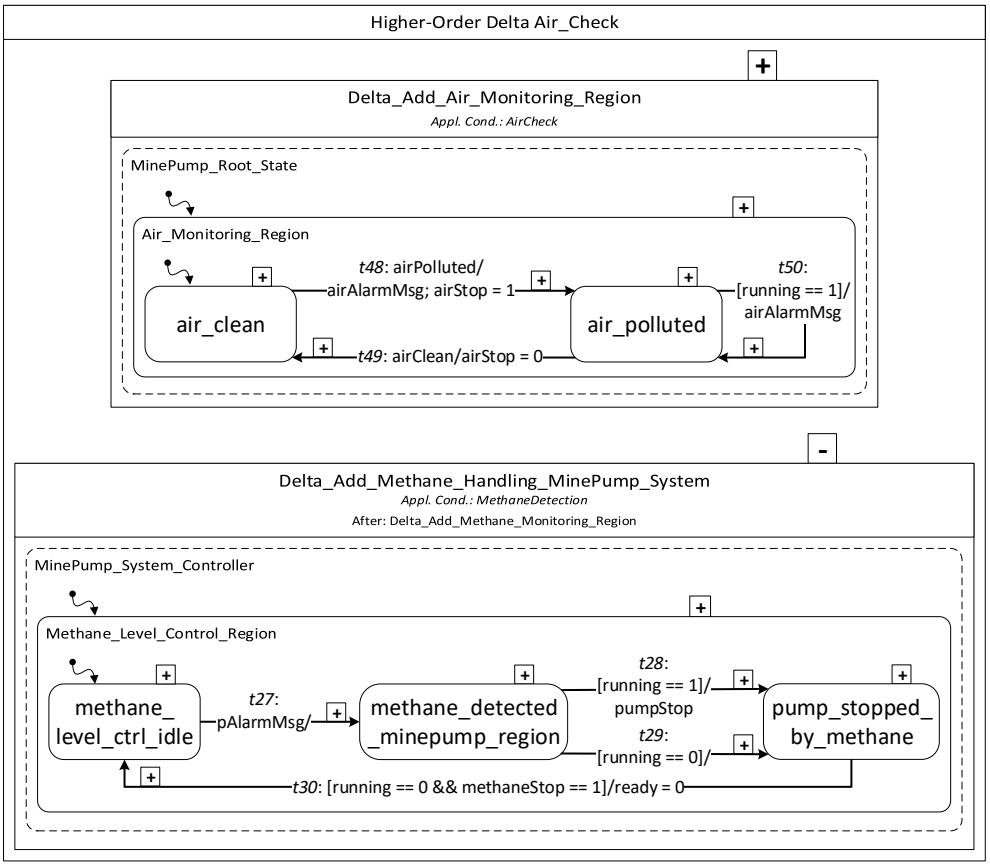

Figure 3.43: First Part of the Higher-Order Delta for Adding the Air Check

Delta_Air_Handling adds transition t53 for the same purpose, but in case the pump is blocked because of air pollution. In case both features are selected, Delta_Air_Methane_Handling adds transition t52 instead, which returns to the initial state if the pump was either blocked by methane or by air pollution. Delta_Air_Alarm_Msg, Delta_Methane_Alarm_Msg as well as Delta_Alarm_Pump_Stop can only be applied after delta Delta_Alarm_Level_Control_Region. Additionally, Delta_Air_Alarm_Msg can only be applied after Delta_Add_Air_Monitoring_Region as well as Delta_Methane_Alarm_Msg can only be applied after Delta_Add_Methane_Monitoring_Region. The deltas Delta_Methane_Alarm_Msg and Delta_Air_Alarm_Msg have to be applied prior to Delta_Methane_Handling, or alternatively Delta_Air_Handling. Furthermore, both Delta_Methane_Alarm_Msg and Delta_Air_Alarm_Msg have to be applied prior to Delta_Air_Methane_Handling.

The deltas Delta_Add_Methane_Handling_HighLevel_No_Cmd_MinePump_System, Delta_Add_Command_Handling_HighLevel_No_Methane_MinePump_System, and Delta_Add_Command_And_Methane_Handling_HighLevel_MinePump_System shown in Fig. 3.45 are modified by the higher-order delta with the result that their application conditions now additionally contain the requirement that feature AirCheck must not be selected.

Additionally, four more deltas shown in Fig. 3.46 are added. Delta_Add_Methane_And_Air_Handling_HighLevel_No_Cmd_MinePump_System modifies transition t12 so that, despite a high water level, t12 further returns to the initial state without action if the pump was stopped because of methane or polluted air. The delta also modifies transition t13 so that t13 additionally checks whether the pump is blocked because of methane or air pollution previous to starting the pump. Delta_Add_Command_And_Air_Handling_HighLevel_No_Methane_MinePump_System equally modifies transitions t12 and t13 with a condition checking for a stop because of air pollution or command, Delta_Add_Command_Methane_And_Air_Handling_HighLevel_MinePump_System modifies the transitions with a condition checking for a stop because of air pollution, methane, or command, and Delta_Add_Air_Handling_HighLevel_No_Command_No_Methane_MinePump_System modifies the transitions with a condition checking for a stop because only of air pollution. Delta_Add_Air_Monitoring_Region has to be applied prior to all four deltas. Additionally, Delta_Add_Methane_And_Air_Handling_HighLevel_No_Cmd_MinePump_System can only be applied after Delta_Add_Methane_Monitoring_Region, Delta_Add_Command_And_Air_Handling_HighLevel_No_Methane_MinePump_System only after Delta_Add_Command_Handling, and Delta_Add_Command_Methane_And_Air_Handling_HighLevel_MinePump_System can not be applied prior to the deltas Delta_Add_Methane_Monitoring_Region and Delta_Add_Command_Handling.

This scenario adds 16 new feature configurations with respective variant-specific delta sets to the software product line. In addition, eight variants are modified due to the modifications of deltas. In total, 32 product variants exist for the mine pump system SPL, where eight variants remain unchanged.

3.3.3 Air Exchange

This scenario adds a mechanism providing an automatic air exchange for the mine pump system.

Description

Initial Situation: The mine pump system has an automatic function for pumping out water in case of high water level and can, if requested, also react to low or normal water levels. Furthermore, it can contain a command function or a methane detection. In the last case, a methane extraction

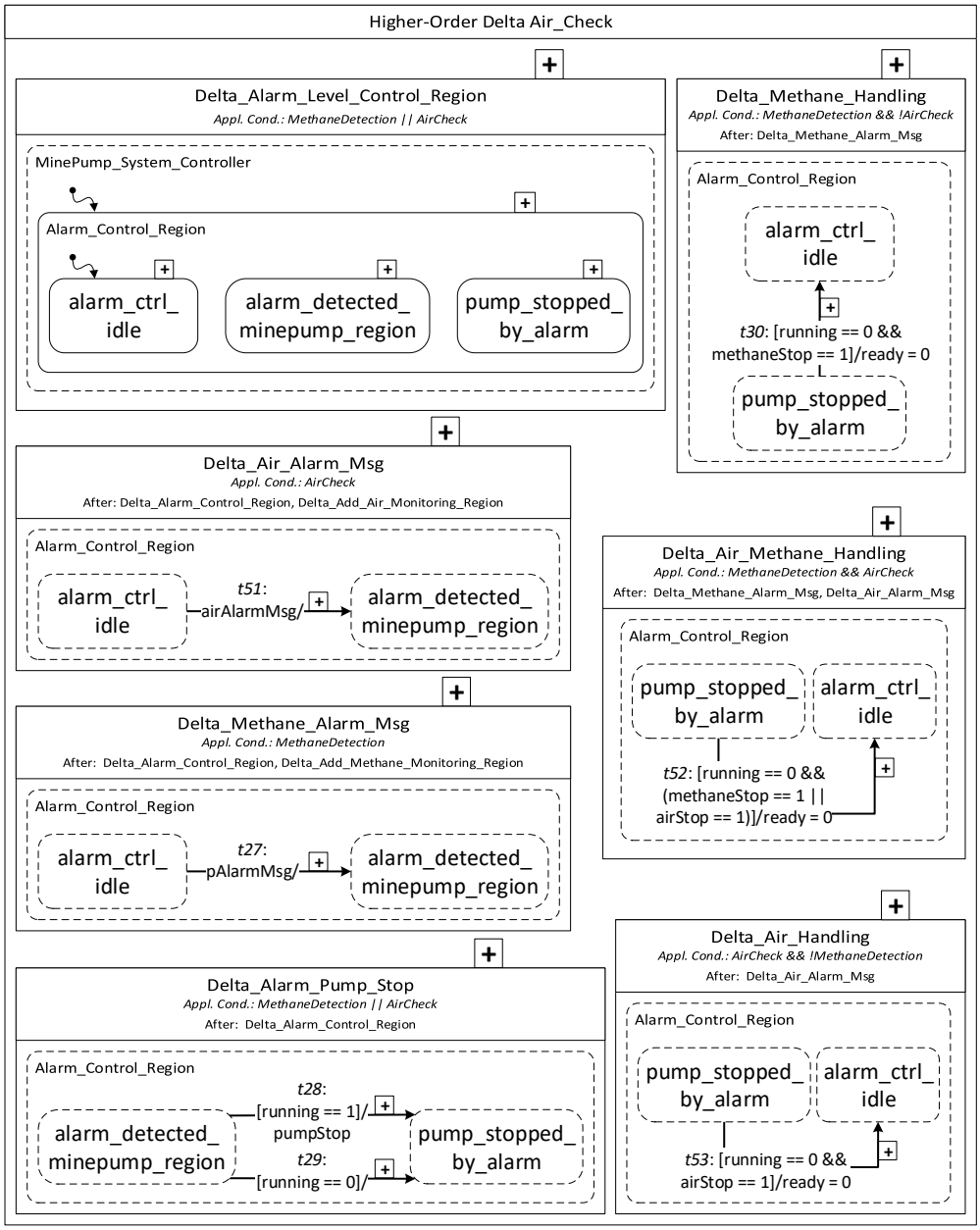

Figure 3.44: Second Part of the Higher-Order Delta for Adding the Air Check

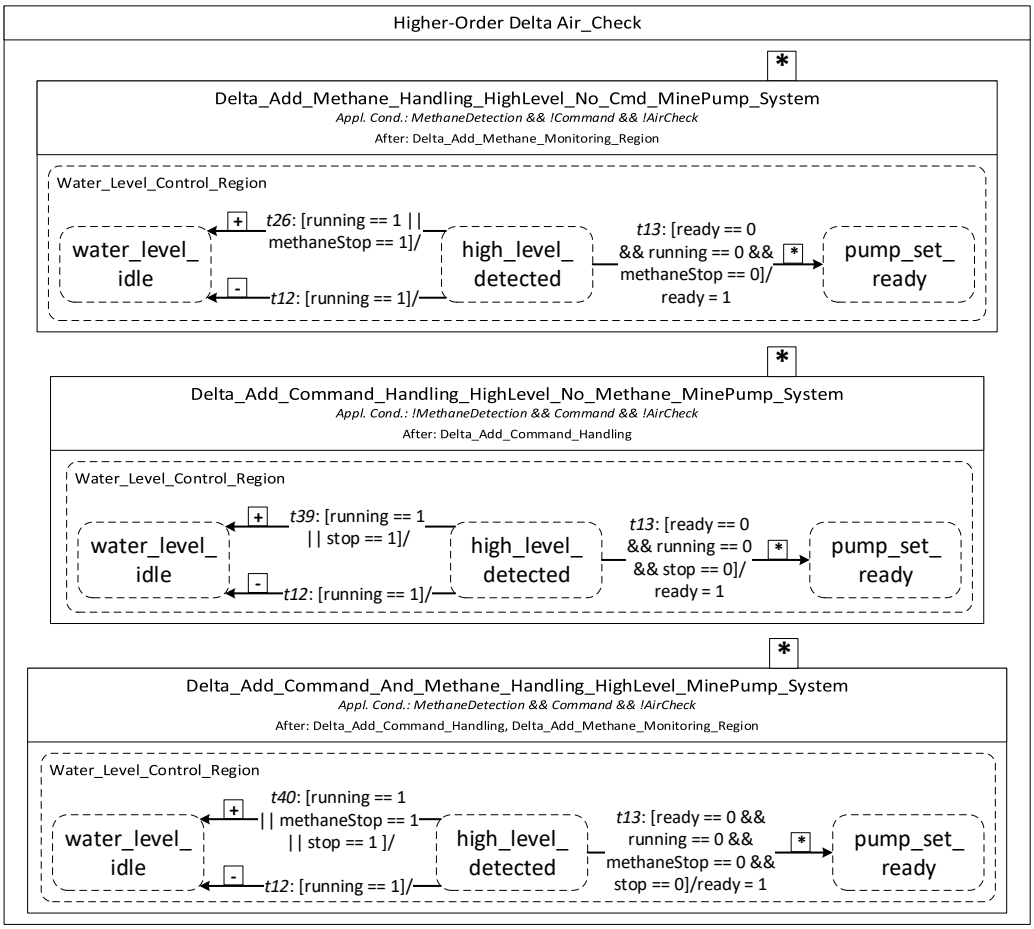

Figure 3.45: Third Part of the Higher-Order Delta for Adding the Air Check

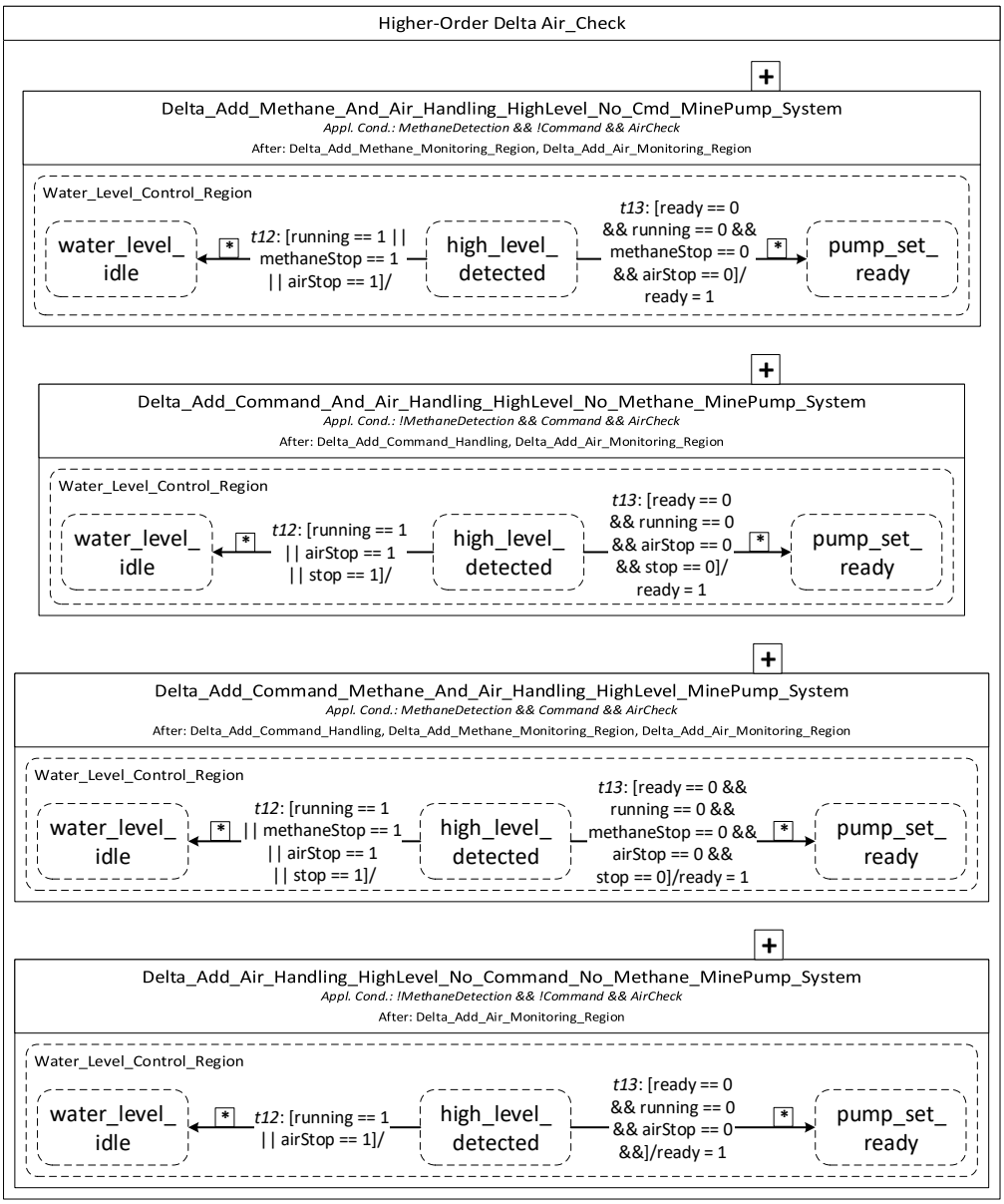

Figure 3.46: Fourth Part of the Higher-Order Delta for Adding the Air Check

is obligatory included. In addition, there is the possibility to choose an air check monitoring the atmospheric conditions and stopping the pump if the air quality is too bad.

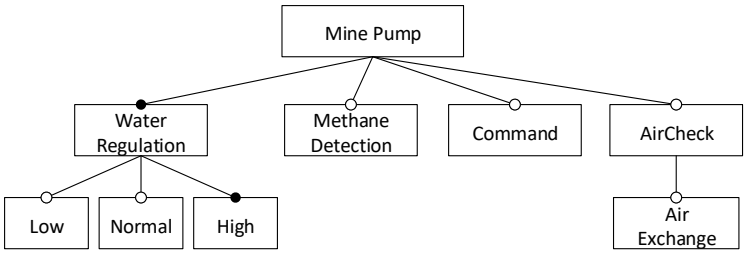

Figure 3.47: Feature Model of the Mine Pump System with AirExchange

Scenario: Caused by the new function `AirCheck`, shafts of badly ventilated mines were flooded frequently because the pump was blocked due to bad air quality. For this reason, the manufacturers now also offer an air exchange similar to the methane extraction for `AirCheck`. The air exchange shall be regulated by the same installation used for the methane extraction. The installation shall cover one or both of the functions according to the selected features of the product.
Realization: The feature model for this scenario is shown in Fig. 3.47, where the optional feature `AirExchange` was added as a child feature to `AirCheck`. Thus, in case `AirCheck` gets selected, there is now the possibility to decide, if `AirExchange` is desired for the system as well.

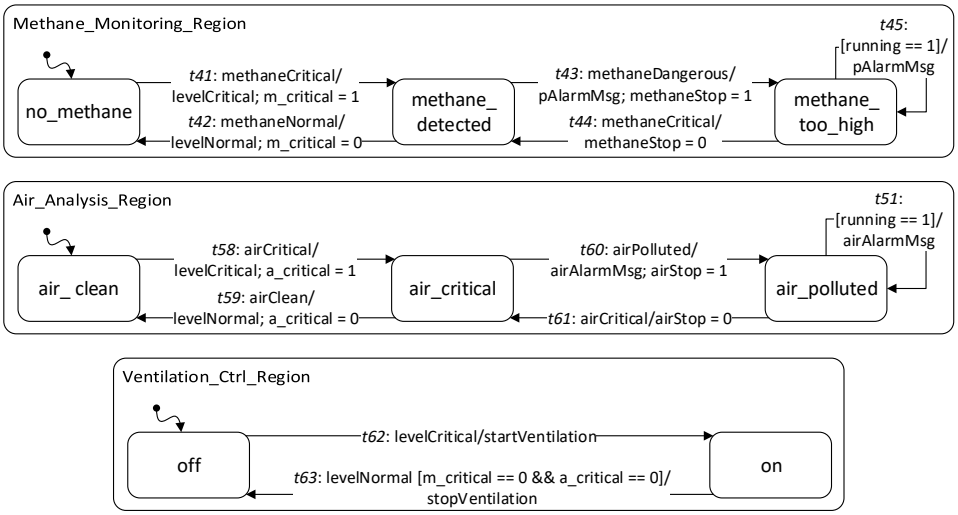

Figure 3.48: Product Model of the Mine Pump System with AirExchange

The parts of the product model changed for this feature are depicted in Fig. 3.48. The classification of the air quality is categorized in three levels, like already done for methane. In case of critical air quality, the ventilation system is started, but the use of the pump is still allowed, whereas in case of polluted air, an alarm is triggered and the pump is stopped or blocked to prevent starting. As soon as the air quality changes from polluted to critical again, the pump can be used once more while the air exchange is still active. Only when the air is clear, the ventilation system is turned

off again. Since the state machine represents a product featuring `MethaneExtraction` as well as `AirExchange`, the ventilation system is turned on if either the air quality or the methane level is critical. Accordingly, the system is only turned off if both values are below critical. The other parts of the product state machine remain unchanged and are, thus, not shown again at this point.

Modeling

For feature `AirExchange`, two new deltas have to be added and further two deltas have to be modified by the higher-order delta `Air_Exchange` shown in Fig. 3.49. In order to adapt the air check to the air exchange, the delta `Delta_Air_Exchange` adds the state `air_critical` to the substate machine `Air_Analysis_Region` and removes the transitions `t48` and `t49`. Additionally, the transitions `t58` indicating critical and `t59` indicating normal air quality, `t60` for triggering the alarm and blocking the pump in case of polluted air, as well as `t61` for unblocking the pump if the air quality is back to critical again, are added to the model.

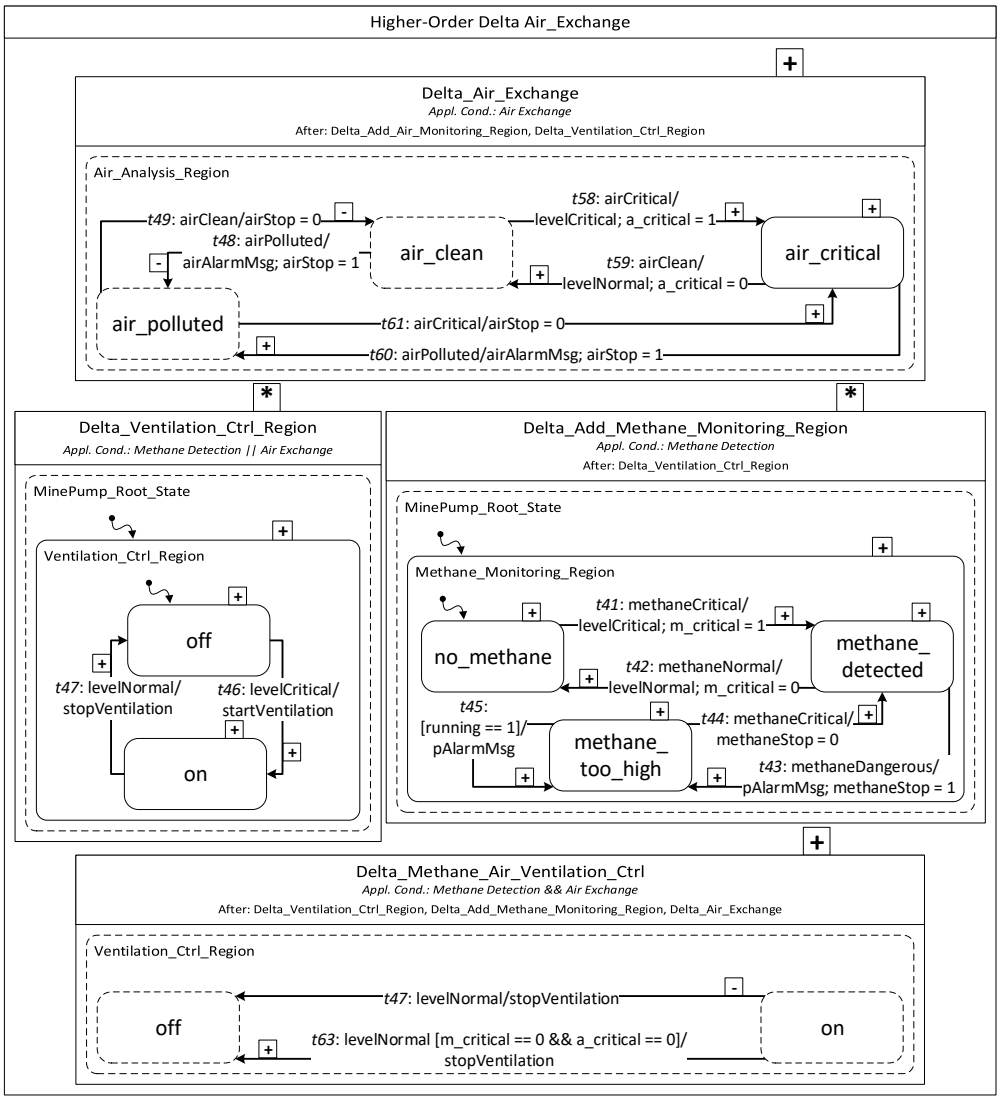

Figure 3.49: Higher-Order Delta for Adding the AirExchange

Delta_Ventilation_Ctrl_Region is modified with the result that it can now be used also for feature AirExchange. In case features MethaneDetection and AirExchange are both selected for the product, the new delta Delta_Methane_Air_Ventilation_Ctrl is applied to remove transition t47 and adds instead the transition t63 for turning off the ventilation only if both the methane content and the air quality are at a normal level. This delta can only be applied after deltas Delta_Ventilation_Ctrl_Region, Delta_Add_Methane_Monitoring_Region, and Delta_Air_Exchange. In order to realize the transition condition for t63, the delta Delta_Add_Methane_Monitoring_Region is also modified with the result that transitions t41 and t42 now additionally save the information if the methane level is critical in a corresponding variable.

Due to this scenario, 16 feature configurations with respective variant-specific delta sets are added. In addition, 16 are modified. The total number of product variants amounts to 48, where 16 variants remain unchanged.

3.4 Body Comfort System

In this section, the evolution scenarios for the body comfort system are described. In total, there are four scenarios including the addition of a wiper system, electrically adjustable seats based on mapped key IDs, heatable windows, and automatic headlights.

3.4.1 Wiper System

In this scenario, a wiper system is added to the body comfort system.

Description

Initial Situation: The body comfort system consists of a door system including an electrically adjustable exterior mirror and manually or automatically controllable power windows, and a human machine interface for interaction which can be equipped with a multitude of different LEDs. In addition, there is the possibility to choose various safety functions like a central locking system, a remote control key, and an alarm system.

Scenario: For offering more comfort to the customers, the default manual wipers shall be replaced by an automatic wiper system with sensors. In order to preempt the competitors, the manufacturers of the body comfort system buy up the company of the best wiper system manufacturer on the market (the one from Sect. 3.2) and integrate the whole software product line of the wiper system into the product line of the body comfort system. In addition, several status LEDs are added for visualization of some of the wiper system functions.

Theoretically, the different functions of the wiper system could also be added step-by-step in several scenarios to the body comfort system. But since this history was already completely described in Sect. 3.2, the whole wiper system is added as a full package in solely one scenario at this point.

Realization: The feature diagram of the body comfort system which has been extended by the whole feature model of the wiper system from Sect. 3.2.5 is shown in Fig. 3.50. The root feature Wiper has been attached as a mandatory child to the feature BodyComfortSystem. Hence, in each product of the body comfort system, there always has to be a wiper system with at least a combination of a low quality wiper and a low quality sensor from now on. Likewise, the or-features LEDWiper, LEDFrostProtection, and LEDClean were added to the feature model. With LEDWiper, there comes a LED glowing when the automatic wiper function is activated, LEDFrostProtection com-

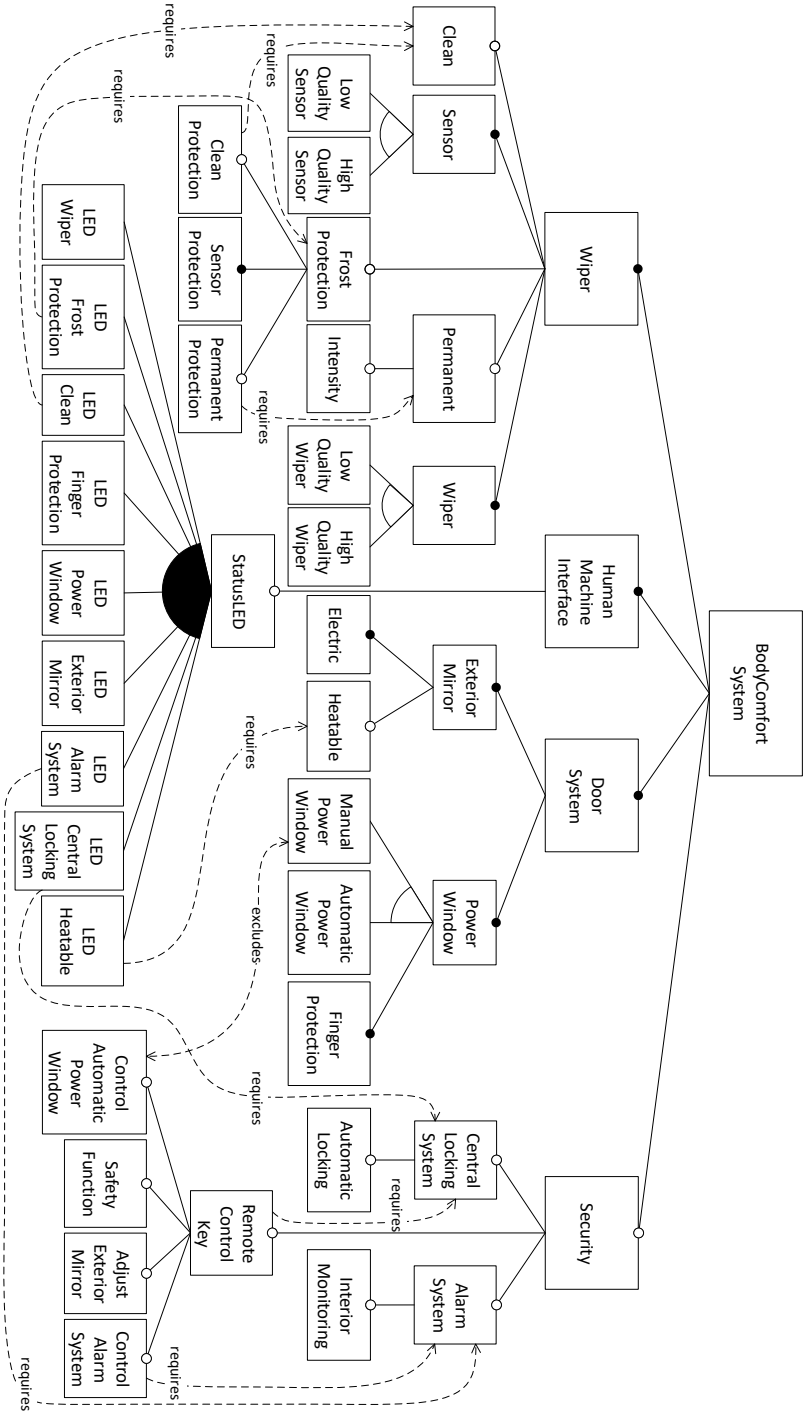

Figure 3.50: Feature Model of the Body Comfort System SPL with the Wiper System

prises a LED displaying if the temperature is too low to use the wiper, and LEDClean provide a LED signaling that the screenwash liquid is consumed. LEDFrostProtection and LEDClean require the features FrostProtection, respectively Clean, and are, thus, linked to these features via a requires-relation.

In order to realize the wiper system in the body comfort system, the new substate machine Wiper is added to the body comfort system. Wiper has the same structure as the core model of the wiper system in Sect. 3.2 and, thus, is not shown separately in the product state machine depicted in Fig. 3.51. Furthermore, three new substate machines handling the LEDs for the features Wiper, Clean, and FrostProtection are added to the substate machine LED. The LED coming with LEDFrostProtection turns on as soon as frost is detected. LED_Clean switches on the LED once a low screenwash liquid level has been reached. This LED is only switched off, when the screenwash liquid has been refilled completely by the service. The LED for Wiper is always on if the automatic wiping function is enabled regardless of whether it is raining or not.

Modeling

The deltas needed to realize the new features are (1) one new delta adding the complete core model shown in Fig. 2.6 and (2) all deltas being valid for the scenario in Sect. 3.2.5. All the deltas of the wiper system which do not have an application order have to be adapted, so that they only can be applied after the delta called Delta_Wiper_Core which adds the core of the wiper system to the state machine of the body comfort system.

In addition, three more new deltas are added by the higher-order delta Wiper which is depicted in Fig. 3.52. DAddStatusLEDFrost adds the substate machine LED_Frost including states frost_led_off and frost_led_on as well as transitions t100 for switching on the LED in case of frost and t101 for switching off the LED in case of no frost to the substate machine LED. This delta can only be applied after the deltas DAddStatusLED and Delta_Temperature. DAddStatusLEDClean adds the substate machine LED_Clean consisting of the states clean_led_off and clean_led_on as well as transitions 102 turning on the LED when too little screenwash liquid is left and t103 for switching it off again as soon as the screenwash liquid was refilled. The deltas DAddStatusLED and Delta_Clean_Counter have to be applied prior to this delta. DeltaAddStatusLEDWiper adds the substate machine LED_Wiper comprising the states wiper_led_off and wiper_led_on as well as the transitions t104 turning on the LED when the automatic wiping function is enabled, and t105 turning off the LED as soon as the automatic wiping is disabled again. DAddStatusLEDWiper can only be applied after the deltas DAddStatusLED and Delta_Wiper_Core.

Furthermore, the higher-order delta Wiper modifies the deltas Delta_Temperature and Delta_Clean_Counter in order to adapt both to the new LEDs. For this purpose, the actions frost and noFrost are added to the transitions t257, respectively t258, of Delta_Temperature. Transition t253 of Delta_Clean_Counter is extended by the action scrwash_full.

3.4.2 Electric Seat Adjustment Based on Key IDs

This scenario adds an automatic and electrical seat adjustment based on mapped key IDs to the body comfort system.

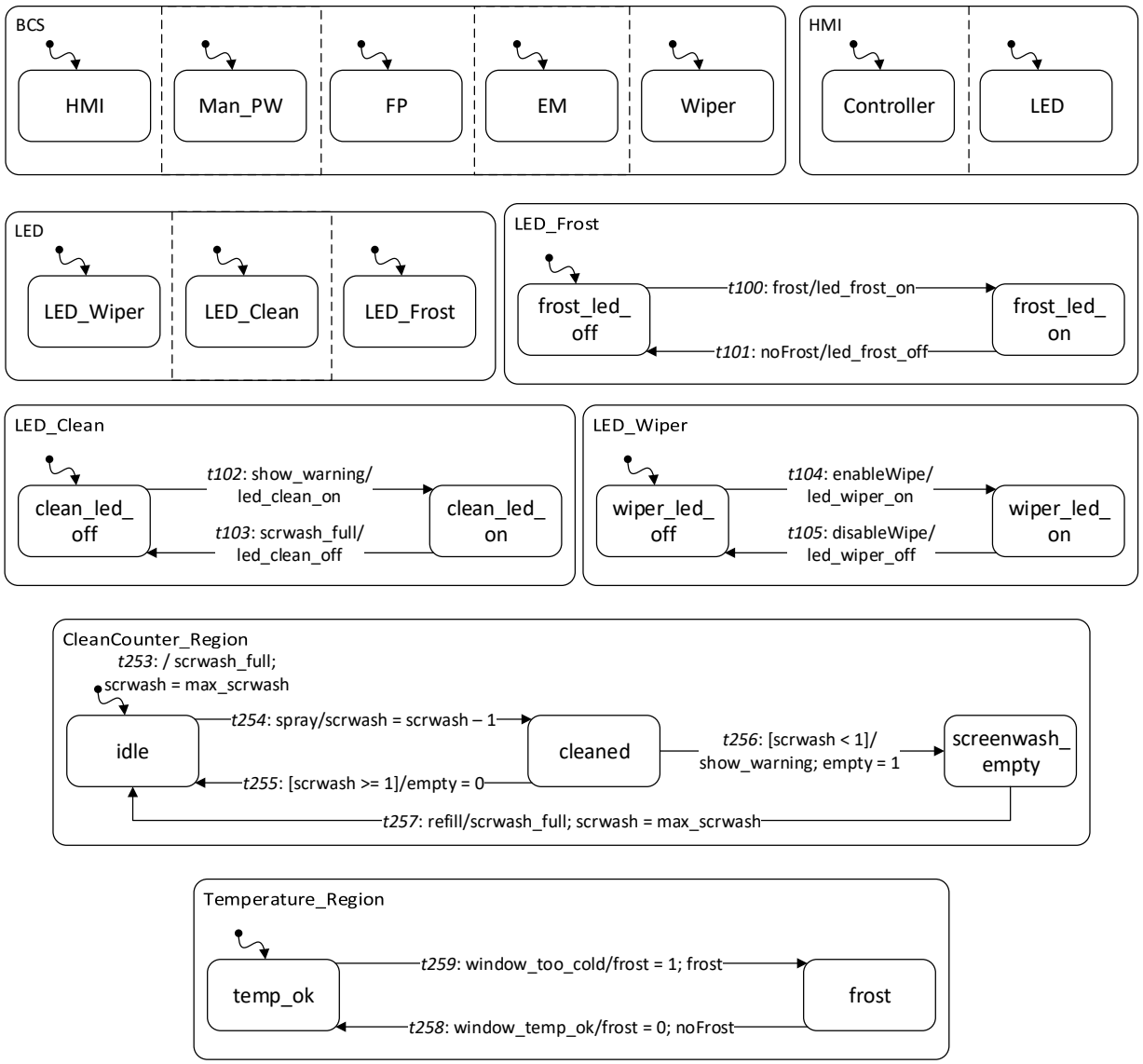

Figure 3.51: Product Model of the Body Comfort System with the Wiper System

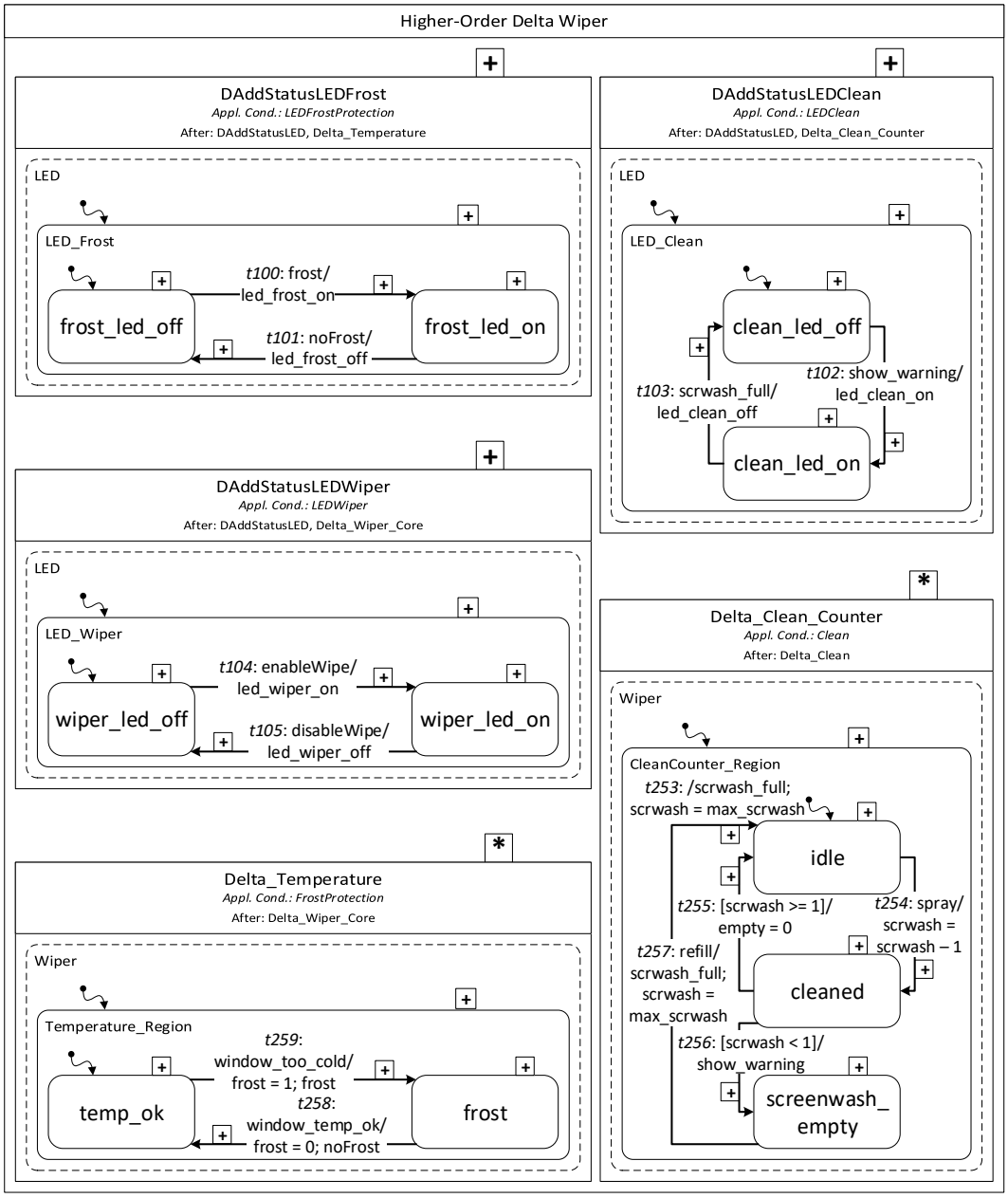

Figure 3.52: Higher-Order Delta for Adding the Wiper System

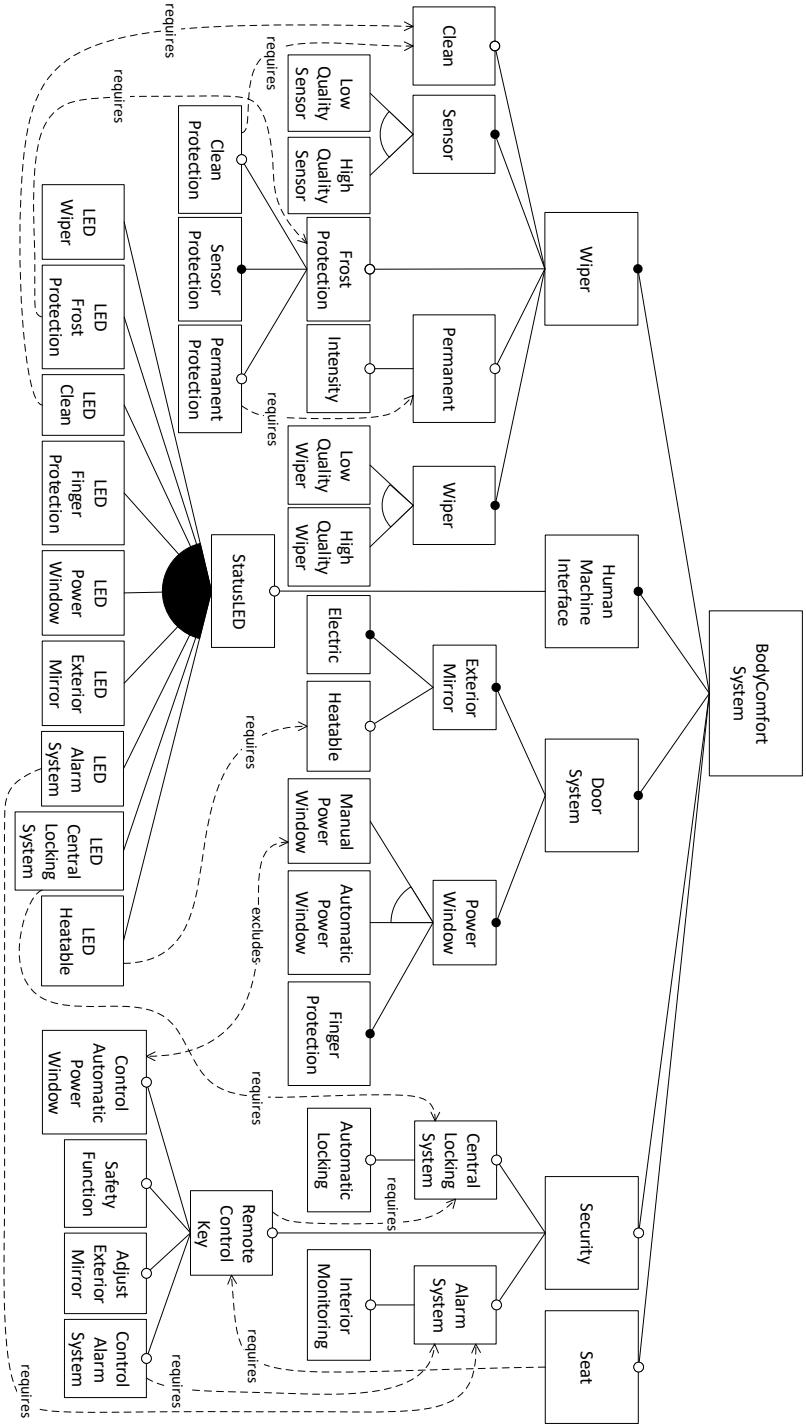

Figure 3.53: Feature Model of the Body Comfort System with Electrically Adjustable Seats

Description

Initial Situation: The body comfort system comprises electrically adjustable exterior mirrors, power windows and a wiper system, can have various LEDs for displaying information and can be equipped with a central locking system, a remote control key, and an alarm system.
Scenario: For many households sharing one vehicle, there arises the problem that every time another person drove before, the driver's seat always has to be repositioned manually. Especially, if two persons take turns at driving frequently, the repositioning can become annoying in the long run. Hence, the body comfort system is to be extended by a function recognizing the ID of the currently used key and saving the seat position based on this ID. Two keys will be delivered per vehicle.
Realization: For this scenario, the optional feature `Seat` is added to the feature model of the body comfort system shown in Fig. 3.53. The electric seat adjustment only works in combination with a remote control key. Hence, feature `Seat` is linked to the feature `RemoteControlKey` via a `requires`-relation.

Based on the ID of a key, the seat position which has been set using this key is saved when locking the vehicle. The next time the vehicle is unlocked with the same key, the saved seat position is read and the driver's seat is adjusted accordingly in case it has been positioned otherwise in the meantime. In order to realize this behavior, a substate machine consisting of two other substate machines is added to the body comfort system. The resulting product state machine containing only the parts relevant for this scenario is depicted in Fig. 3.54. `Set_Seat` loads the saved seat position according to the key ID, adjusts the seat if necessary and saves the position again later on. `Seat_Pos` handles the manual adjustment in forwards or backwards direction of the driver's seat and saves the set seat position. This seat position is assigned to the key ID when locking the car. Furthermore, the substate machine of the central locking system is adapted with the result that the saving of the seat position is triggered when the car gets locked and the adjusting of the seat is triggered according to the key when the car gets unlocked.

Modeling

Eleven new deltas are added by the higher-order delta `Seat` shown in Fig. 3.55 and Fig. 3.56. `DAddSeat` adds the empty substate machine `Seat` to the state machine `BCS`, while `DAddSeatSet` adds the substate machine `Set_Seat` to `Seat`. `Set_Seat` consists of the states `keys_idle`, `key1_in_use`, and `key2_in_use` as well as the transitions `t106` and `t109` checking if the saved seat position of key 1, or respectively key 2, matches the actual seat position. It further contains the transitions `t107` and `110` for saving the seat position for key 1, alternatively key 2, and transitions `t108` and `t111` for adjusting the respectively saved seat position in case the saved position does not match the actual position of the driver's seat.

`DAddSeatPos` adds the substate machine `Seat_Pos` to substate machine `Seat`. `Seat_Pos` contains the states `position`, `min_position`, and `max_position` as well as the transitions `t112` for moving the seat in backwards direction from the foremost position, `t113` for reaching the foremost position, `t114` for moving the seat in backwards direction from an arbitrary position not being the foremost one, `t115` for moving the seat forwards from an arbitrary position not being the rearmost position, `t116` for reaching the rearmost position, and `117` for moving the seat in forwards direction from the rearmost position. `DAddSeat` has to be applied prior to `DAddSeatSet` and `DAddSeatPos`.

In addition, the mechanism of the central locking system has to be adapted. For this purpose,

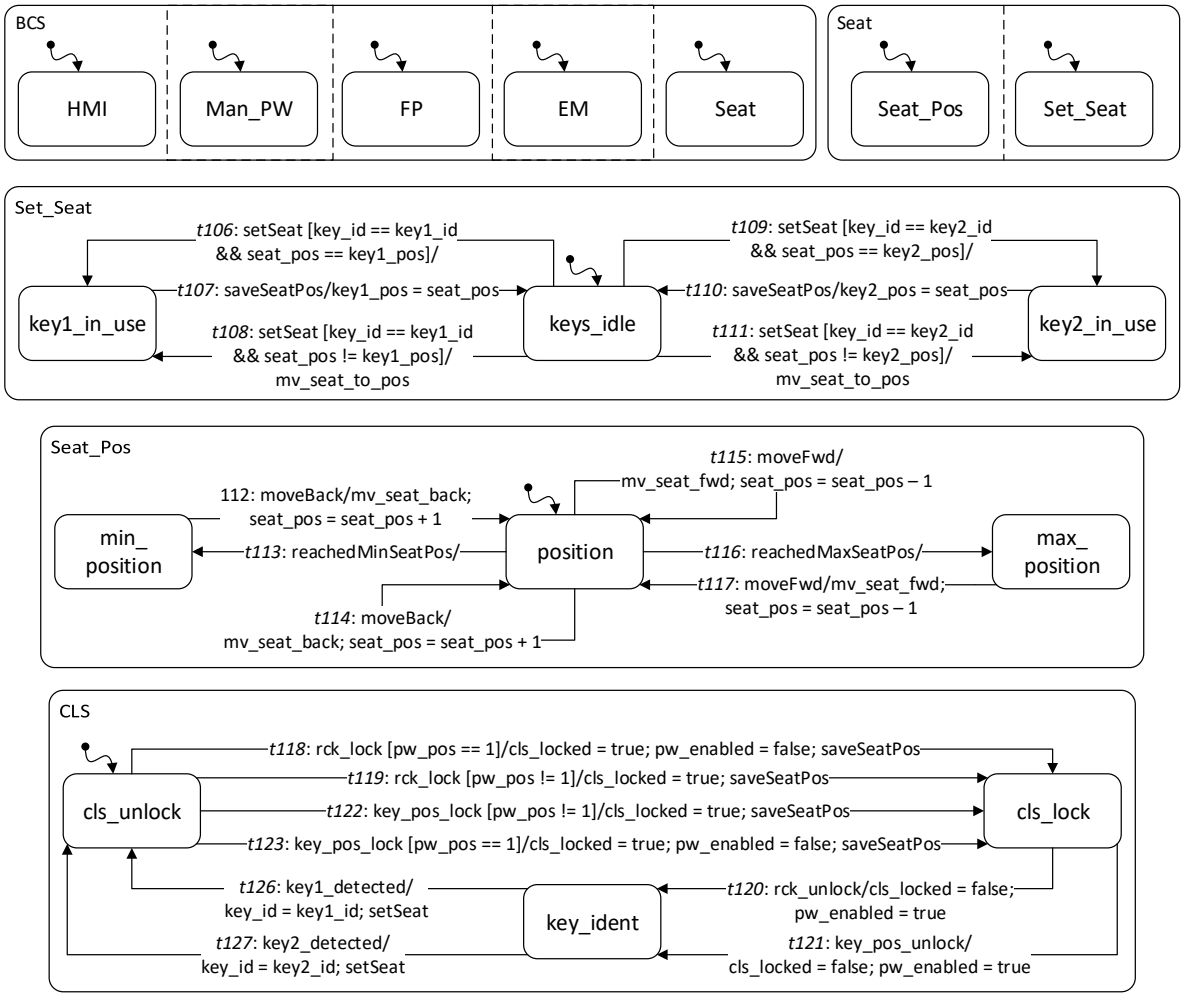

Figure 3.54: Product Model of the Body Comfort System with Electrically Adjustable Seats

DAddCLSBSMManPWSeat modifies the transitions t49 and t50, while DAddCLSBSMManPWRCKSeat modifies the transitions t45 and t46, DAddCLSBSMAutoPWSeatKey modifies the transition t51, and DAddCLSBSMAutoPWRCKSeatKey modifies t52 with the result that they all trigger the storing of the actual driver's seat position when locking the car. All those deltas can only be applied after the deltas which add the respective transitions getting modified.

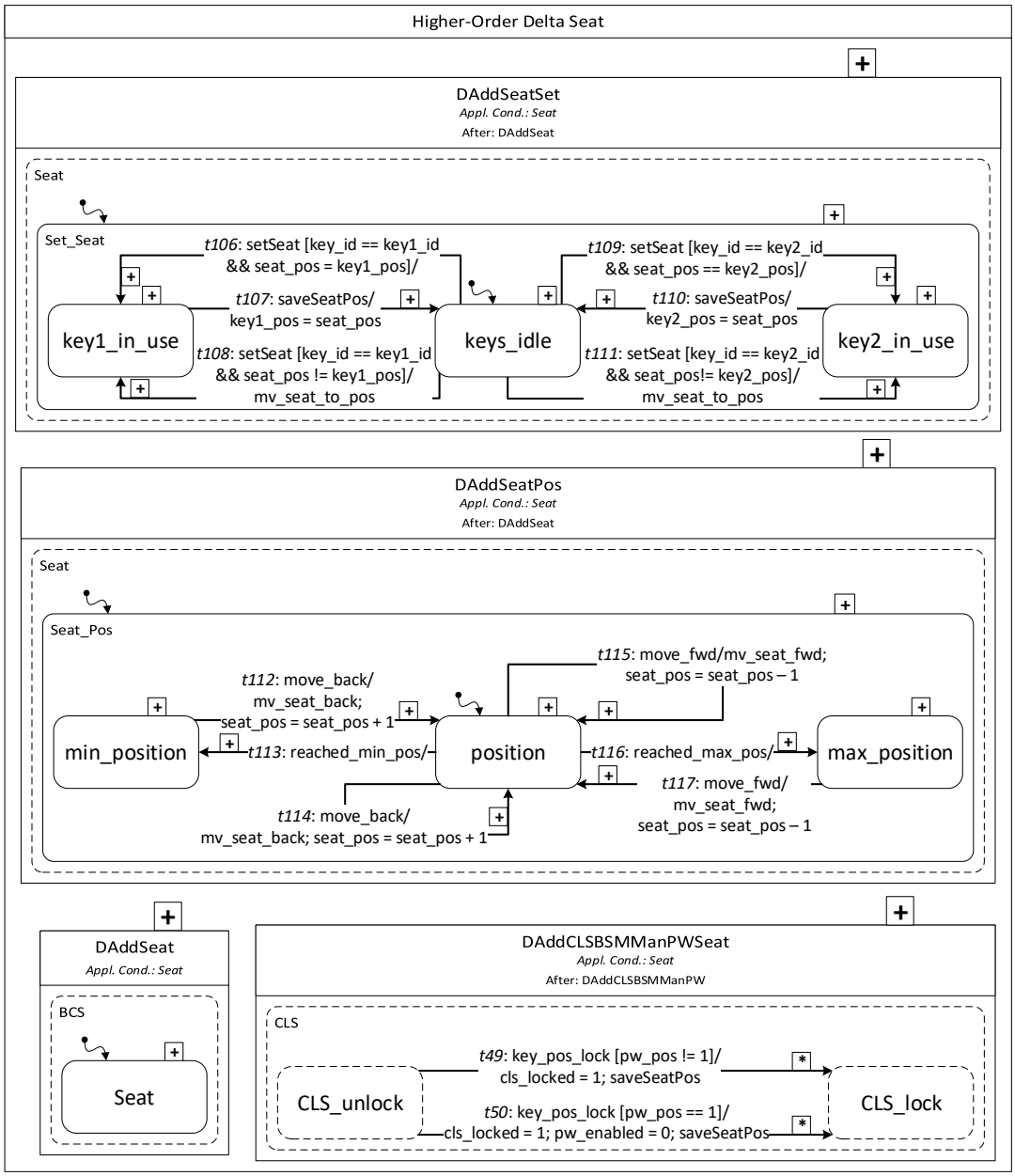

Figure 3.55: First Part of the Higher-Order Delta for Adding Electrically Adjustable Seats

In order to activate the automatic seat adjustment when unlocking the car, the state key_ident has to be added to the substate machine CLS by the delta DAddSeatKeyIdent. DAddCLS has to be applied prior to DAddSeatKeyIdent. Next, DAddCLSSBMSeatKey removes transition t48 leading

from state CLS_lock to CLS_unlock, and inserts it again leading from CLS_lock and key_ident. Same applies for delta DAddCLSSBMRCKSeat concerning transition t47. Delta DAddCLSSBMSeat can only be applied after deltas DAddCLSSBM and DAddSeatKeyIdent, while DAddCLSSBMRCKSeat can only be applied after DAddCLSSBMRCK and DAddSeatKeyIdent.

Furthermore, the transitions t118 and t119 are added between the states key_ident and CLS_unlock by delta DAddCLSSBMSeatSetKey. Those two transitions are responsible for detecting which key has been used and pass this information along for loading and saving the seat position. The deltas DAddSeatSet and DAddSeatKeyIdent have to be applied prior to DAddCLSSBMSeatSetKey.

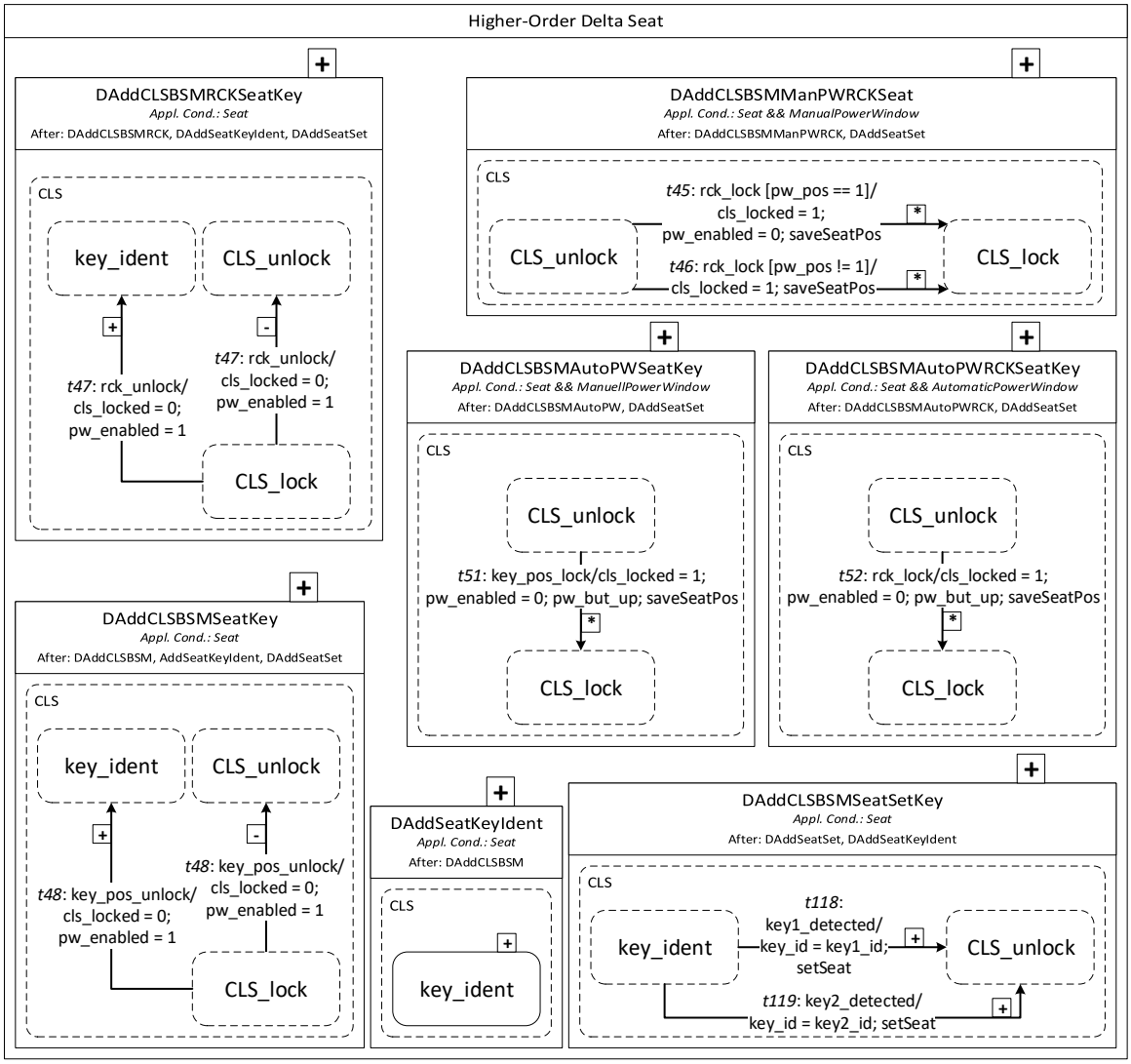

Figure 3.56: Second Part of the Higher-Order Delta for Adding Electrically Adjustable Seats

3.4.3 Heatable Windows

In this scenario, the body comfort system is extended by automatically heatable windows.

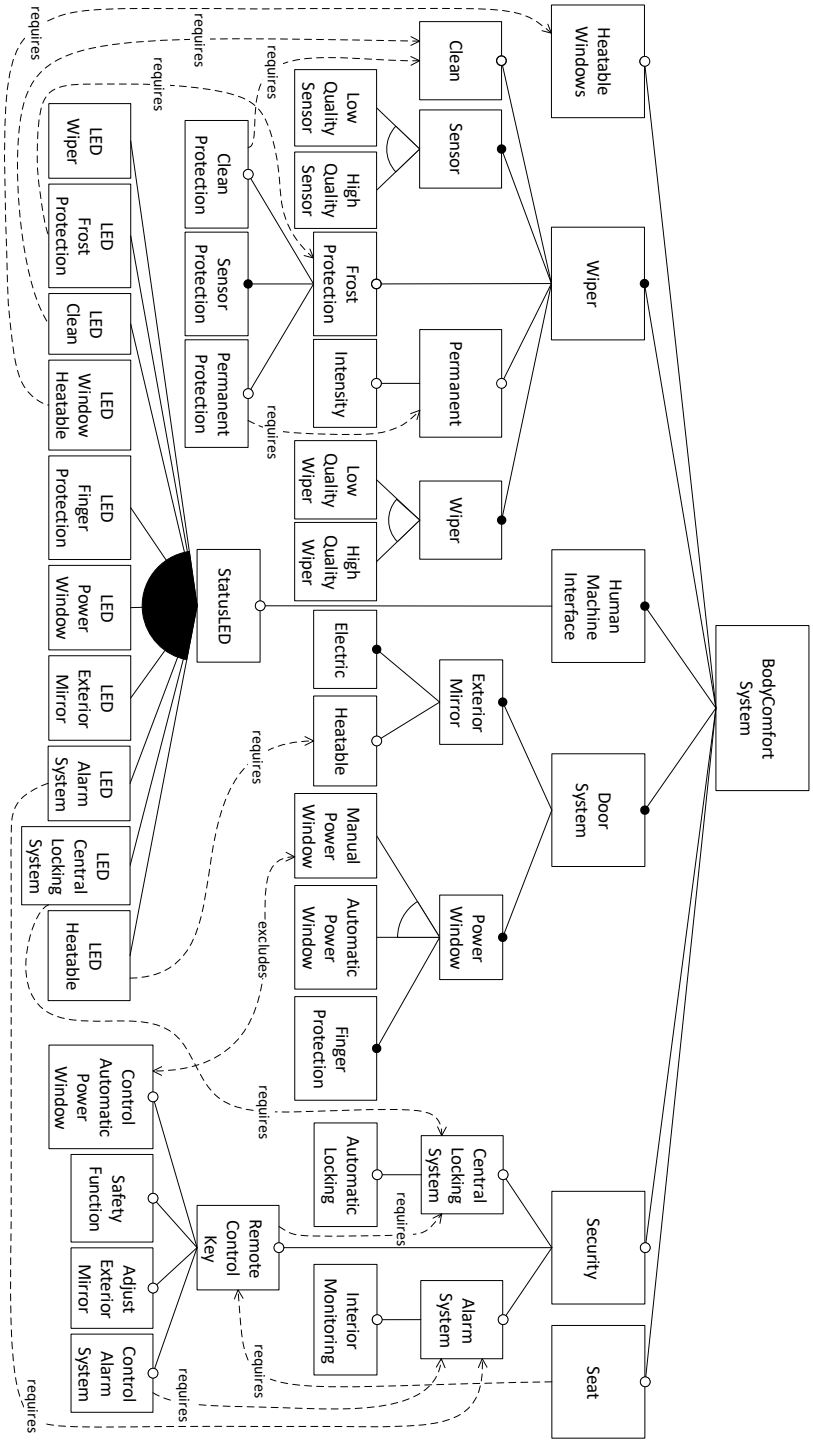

Figure 3.57: Feature Model of the Body Comfort System SPL with Heatable Windows

Description

Initial Situation: The body comfort system consists of a door system containing electrically adjustable exterior mirrors being heatable if requested as well as manually or automatically controllable power windows, a human machine interface which can be equipped with various LEDs and the wiper system introduced in Sect. 3.2. In addition, an automatically adjustable driver's seat based on key ID or safety functions like a central locking system, a remote control key, and an alarm system can be added.

Scenario: Many customers complain that it is extremely time-consuming to defrost the windows in case of extremely cold temperatures in winter. They further say, that it takes a very long time until the front window reaches a positive temperature. Thus, for products containing the frost check, it may take a while until the wiper is ready to be used again. For that reason, the body comfort system shall be extended by heatable windows.

Realization: For this function, the new optional feature `HeatableWindows` is added to the feature model shown in Fig. 3.57. For the heatable windows, a LED displaying whether the window heater is turned on can also be integrated if requested. For that purpose, the feature `LEDWindowHeatable` is added additionally. `LEDWindowHeatable` always requires the feature `HeatableWindows` and is, thus, connected with the latter via a `requires`-relation.

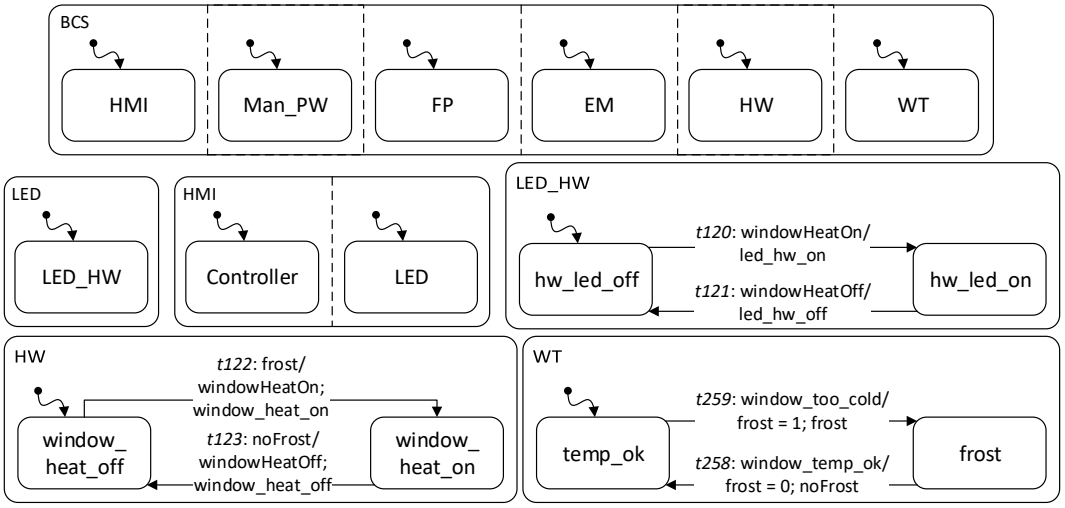

Figure 3.58: Product Model of the Body Comfort System with Heatable Windows

A function monitoring the window temperature and signaling if the latter is too cold, exists already in the context of the wiper system. This function is now transferred from the wiper system and directly integrated as an own substate machine `WT` (Window Temperature) in the state machine `BCS`. Thus, both the feature `HeatableWindows` and the feature `FrostProtection` can access the function. In addition, the substate machines `HW` (Heatable Windows) and `LED_HW` for turning on and off the window heater, or alternatively the LED of the window heater, are added.

Modeling

In order to realize this feature, two new deltas are added to the delta model and one existing delta is modified by the higher-order delta `Heatable_Windows` depicted in Fig. 3.59. Delta `DAddHW` adds the substate machine `HW` to state machine `BCS`. `HW` consists of the states `window_heat_off` and `window_heat_on` as well as transitions `t122` and `t123`. Transition `t122` turns on the window heater as soon as the window is too cold, while transition `t123` turns the window heater off once the window temperature is high enough again. This delta can only be applied after `Delta_Temperature`.

Delta `DAddStatusLEDHW` adds the substate machine `LED_HW` to the substate machine `LED`. `LED_HW` contains the states `hw_led_off` and `hw_led_on` as well as the transitions `t120` and `t121`. Transition `t120` switches on the LED displaying that the window heater is active when the window heater is turned on, whereas transition `t121` switches off the LED once the window heater is turned off again. The deltas `DAddStatusLED` and `DAddHW` have to be applied prior to delta `DAddStatusLEDHW`.

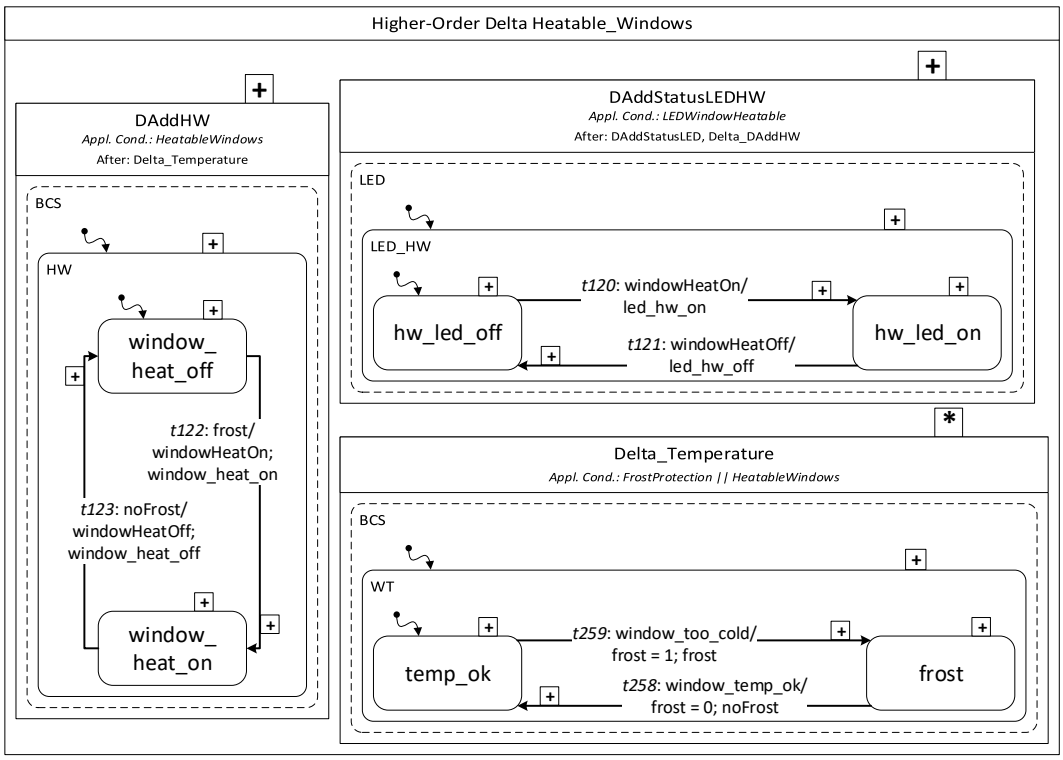

Figure 3.59: Higher-Order Delta for Adding Heatable Windows

Furthermore, `Delta_Temperature` is modified with the result that, on the one hand, the application condition is extended so that `Delta_Temperature` can be applied both for `FrostProtection` and `HeatableWindows`, and on the other hand, the states `temp_ok` and `frost` as well as the transitions `t259` and `t258` are now inserted into the substate machine `WT` which is directly added to the state machine `BCS`. Hence, this substate machine is not a subordinate part of the wiper system anymore, but exists independently.

3.4.4 Automatic Headlights

This scenario adds a function for automatic headlights to the body comfort system product line.

Description

Initial Situation: The body comfort system comprises a wiper system, a human machine interface, electrically adjustable exterior mirrors being heatable on request, and manually or automatically controllable power windows. Furthermore, it can contain automatically heatable windows, an electrically adjustable driver's seat based on key IDs, a central locking system, a remote control keys, an alarm system, and various LEDs for displaying informations.
Scenario: Because in many cases the drivers notice too late that meanwhile light conditions which call for switching on the light are reached, the body comfort system shall be extended by an automatic light function automatically adapting the headlights to the daylight and driving conditions.
Realization: The automatic light function is realized in the feature model depicted in Fig. 3.60 by adding the optional feature `AutomaticHeadlights`.

Figure 3.61 shows a product state machine with the new substate machine `Light` consisting of the five additional substate machines `ManLight`, `NatLight`, `AutoLight`, `HighBeam`, and `AutoHB`. `ManLight` allows for the manual choosing whether the headlights shall be turned off, turned on, or set to automatic. `NatLight` monitors the lighting conditions outside and concludes accordingly if the conditions are classified as dark or not dark. `AutoLight` determines which headlight type is appropriate depending on the provided classification as well as situation and switches it on. When starting the motor, the parking lights (PL) are switched on for a start, then, as soon as the vehicle starts driving, the daytime running lights (DRL) are switched on in case there is enough light, or the low beam (LB) is switched on in case it is dark. If the lighting conditions change during the ride, the system automatically switches to the according headlight type immediately. The parking lights are only used again once the motor is turned off, thus, preventing that the system changes to the parking lights for every little stop, for instance when waiting at traffic lights. Instead, the system still uses the daytime running lights or the low beam when the vehicle is stopping. In case the driver manually switches to automatic light only while already driving, the system immediately switches to the daytime running lights or the low beam according to the lighting conditions. Furthermore, the high beam (HB) can be manually turned off, on, or set to automatic by `HighBeam`. In case the high beam is set to automatic, it is switched on automatically as soon as it is dark outside and there is no approaching vehicle detected from the front. The high beam is automatically switched off once a vehicle is approaching from the front.

Modeling

In order to realize this function, eleven new deltas are needed and, thus, added by the higher-order delta `Automatic_Headlights` shown in Fig. 3.62 and Fig. 3.63.

`DAddLight` adds the empty substate machine `Light` to the state machine `BCS`. The substate machines `NatLight`, `AutoHB`, `HighBeam`, `ManLight`, and `AutoLight` are inserted into `Light` by the deltas `DAddLightNL`, `DAddLightAHB`, `DAddLightHB`, `DAddLightML`, and `DAddLightAL`. Those five deltas can only be applied after the delta `DAddLight`.

The delta `DAddLightNLNatLight` adds the states `not_dark` and `dark` as well as the transitions `t124` for detecting darkness and `t125` for detecting sufficient daylight to the substate machine

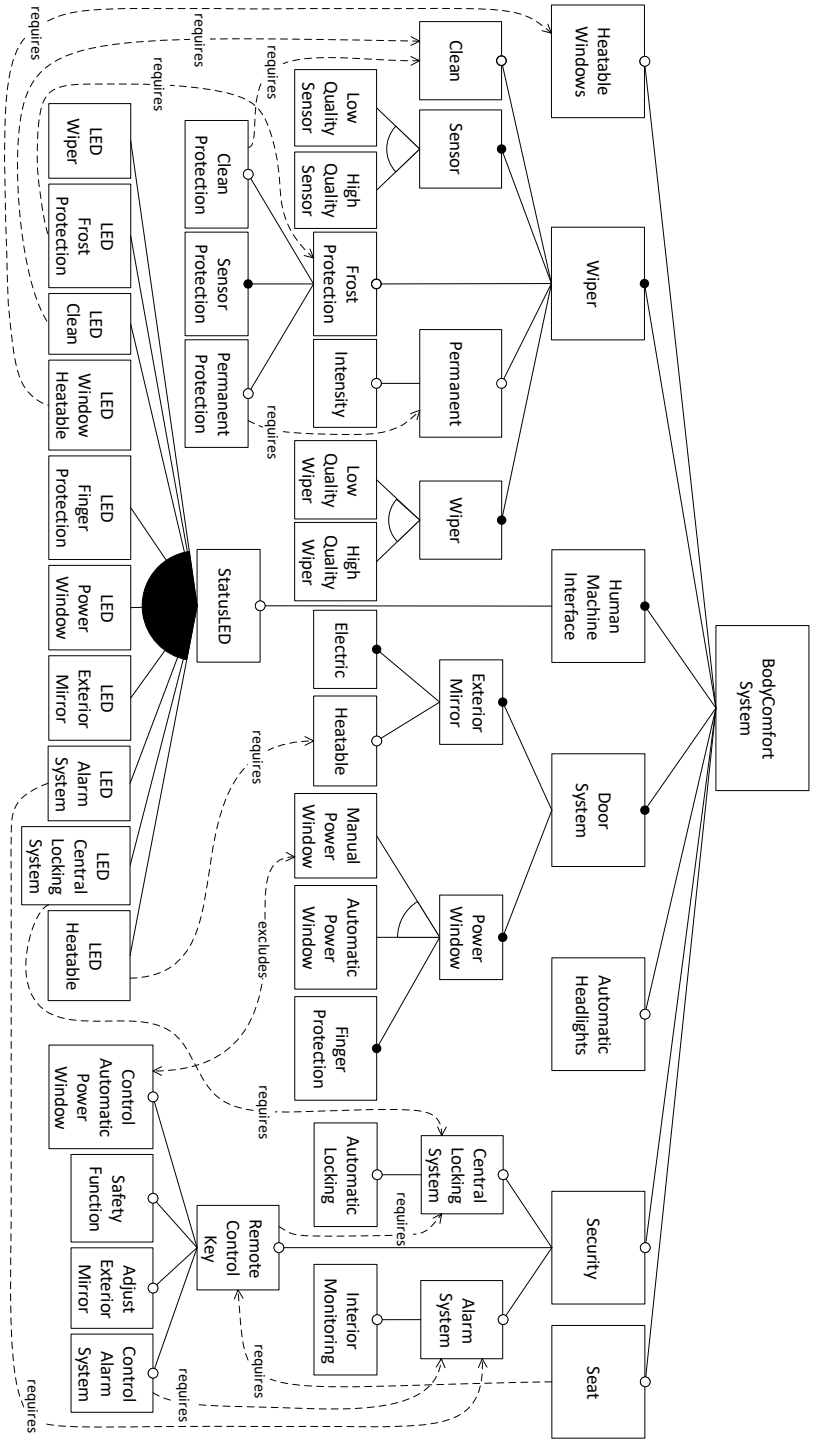

Figure 3.60: Feature Model of the Body Comfort System SPL with Automatic Headlights

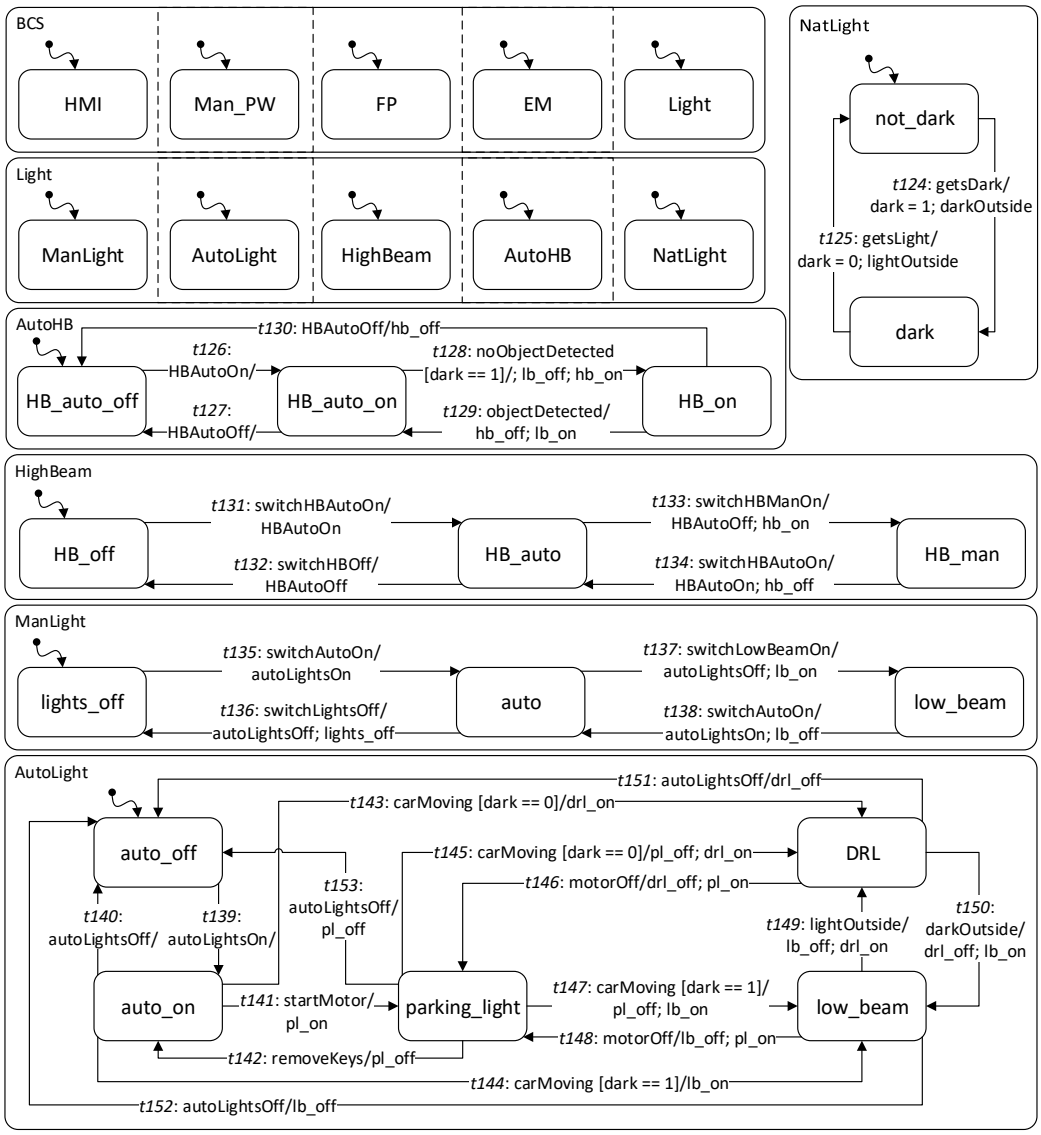

Figure 3.61: Product Model of the Body Comfort System with Automatic Headlights

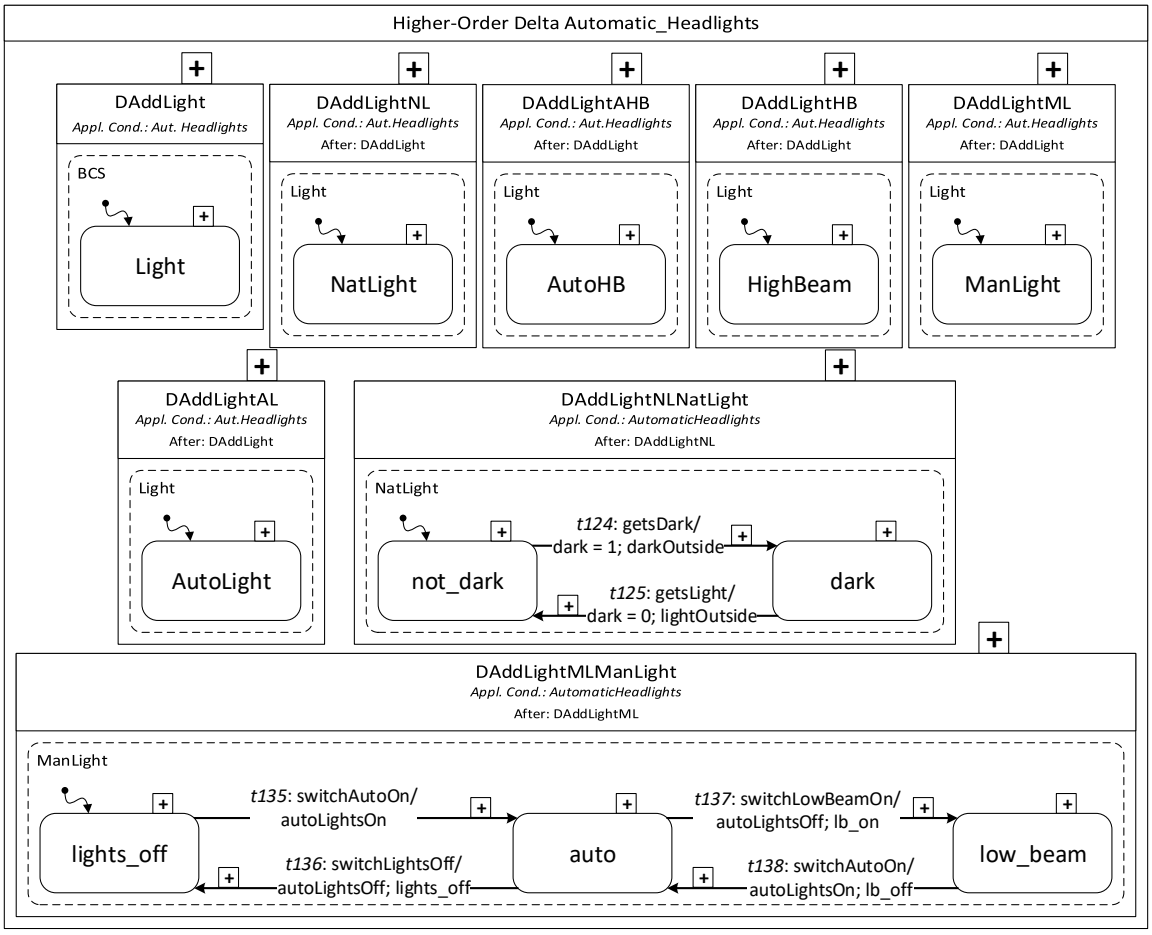

Figure 3.62: First Part of the Higher-Order Delta for Adding Automatic Headlights

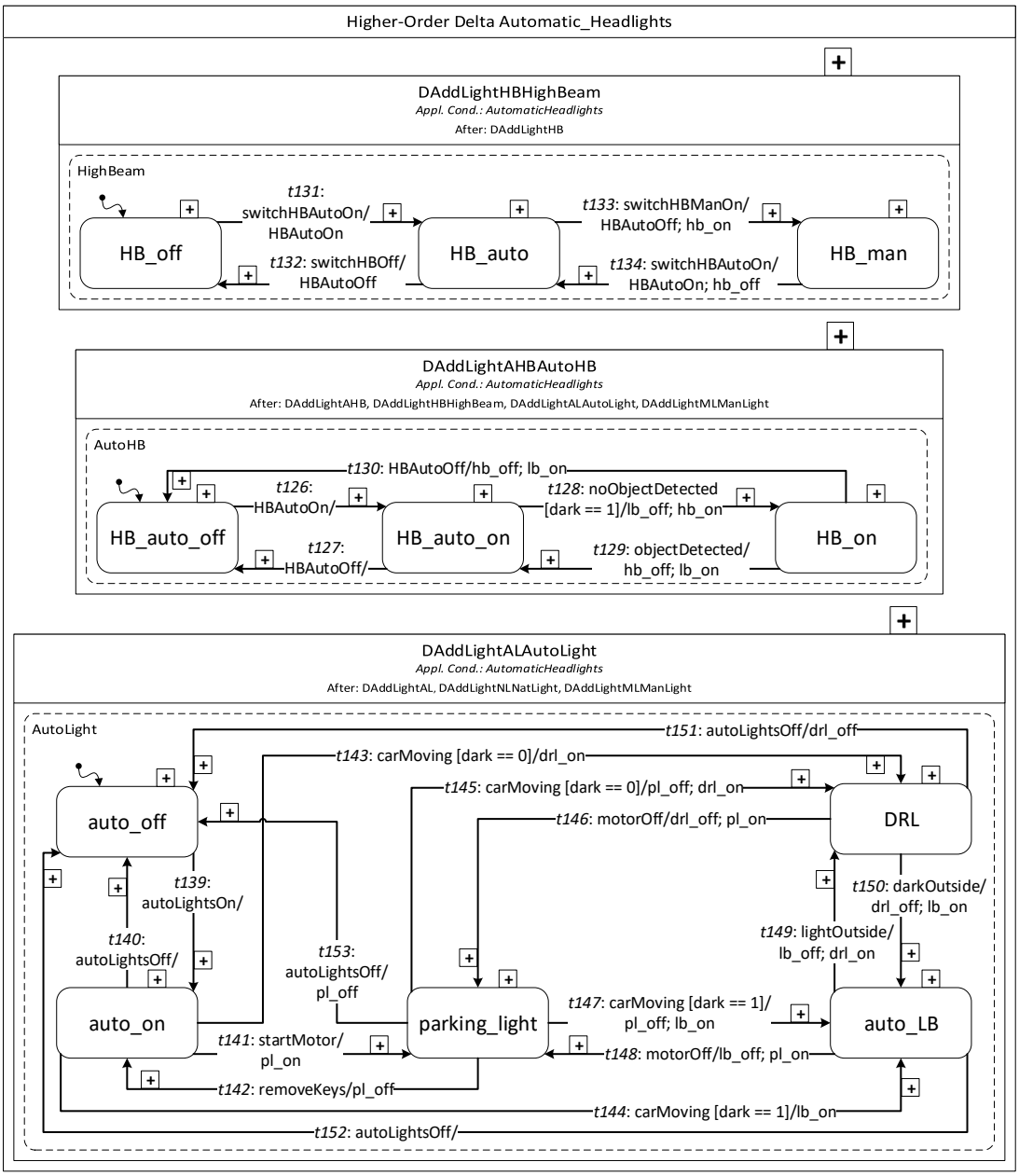

Figure 3.63: Second Part of the Higher-Order Delta for Adding Automatic Headlights

NatLight. DAddLightNL has to be applied prior to DAddLightNLNatLight. Furthermore, the delta DAddLightMLManLight inserts the states lights_off, auto, and low_beam as well as the transitions t135 for activating the automatic light, t136 for manually turning off the lights, t137 for manually turning on the low beam, and t138 to switch from low beam to automatic light, into the substate machine ManLight. DAddLightMLManLight can only be applied after delta DAddLightML. To the substate machine HighBeam, the states HB_off, HB_auto, and HB_man as well as the transitions t131 for activating the automatic high beam, t132 for manually switching off the high beam, t133 for manually activating the permanent high beam, and 134 for switching from the permanent high beam to automatic high beam, are added by delta DAddLightHBHighBeam. Delta DAddLightHB has to be applied prior to DAddLightHBHighBeam. The delta DAddLightAHBAutoHB adds the states HB_auto_off, HB_auto_on, and HB_on to the substate machine AutoHB. In addition, this delta inserts the transitions t126 for activating the automatic high beam, t127 and t130 for deactivating the automatic high beam if the high beam is switched off, or alternatively on, transition t128 for automatically switching on the high beam in case it is dark and no vehicle approaching from the front was detected, as well as 129 for automatically switching off the high beam if another vehicle is approaching from the front. DAddLightAHBAutoHB can only be applied after the deltas DAddLightAHB and DAddLightHBHighBeam.

Furthermore, delta DAddLightALAutoLight inserts the states auto_off, auto_on, parking_light, DRL, and low_beam into the substate machine AutoLight. Additionally, the delta adds 15 transitions to AutoLight. Transition t139 serves to activate the automatic light, while t140 deactivates it. Transition t141 switches on the parking lights when the motor is started, t143 turns on the daytime running lights after activating the automatic light in case the vehicle is already moving and it is not dark outside, while transition t144 turns on the low beam after activating the automatic light in case the vehicle is already moving and it is dark outside. Transition t145 switches from parking lights to daytime running lights if it is light outside and the vehicle starts moving, while t147 switches from parking lights to low beam if it is dark outside and the vehicle starts moving. Transitions t146 and t148 switch from daytime running lights, or alternatively low beam, back to parking lights once the key is removed from the ignition lock. Furthermore, the transitions t149 and t150 switch between daytime running light and low beam in case the lighting conditions change during the ride, and transitions t151, t152 as well as t153 switch off daytime running lights, low beam and parking lights as well as deactivate the automatic light in case the lights are turned off manually. The deltas DAddLightAL, DAddLightNLNatLight, and DAddLightMLManLight have to be applied prior to delta DAddLightALAutoLight.

4 Conclusion

In this technical report, we proposed and documented the evolution of four delta-oriented software product lines. Therefore, we first described the original artifacts of each product line, namely the vending machine SPL [3], the wiper system SPL [3], and the mine pump system SPL [3] as well as the body comfort system SPL [12] including the respective feature model and the behavioral specification modeled as delta-oriented state machines comprising a core state machine and deltas to transform the core in each variant-specific model. Based on those original versions used as initial versions of the corresponding evolution histories, we apply higher-order delta modeling [11] to capture the evolution of each product line. For each evolving SPL, we, therefore, described the causes and characteristics of the developed evolution steps, where the changes to the feature models as well as the delta models were documented en detail.

For the vending machine SPL, the evolution scenarios amount to seven scenarios with, altogether, 41 added, six removed and six modified deltas. The evolution of the wiping system SPL resulted in five scenarios comprising 21 added and two modified deltas. For the mine pump system SPL, three scenarios were defined including 16 added, one removed and five modified deltas. The body comfort system SPL was extended by four scenarios with 52 added and one modified delta. To this end, the product line of the vending machine increased from 20 to 90 possible product variants, whereas the number of valid configurations and variants of the wiper system SPL increased from eight to 84. The mine pump system SPL, starting with 16 product variants, now encompasses 48, whereas the body comfort system SPL has now more than 100,000 variants.

In future research, the proposed evolution histories of the systems facilitate the evaluation of new methods for modeling evolution. Furthermore, the evolution histories may not only be used for delta-oriented techniques, but can also be used as basis for the evaluation of other transformational, annotative, or compositional approaches.

Bibliography

[1] Jörg Becker, Wolfgang Probandt, and Oliver Vering. "Modellierung". In: *Grundsätze ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–3.

[2] Dave Clarke, Michiel Helvensteijn, and Ina Schaefer. "Abstract Delta Modelling". In: *Mathematical Structures in Computer Science* 25.3 (2015), pp. 482–527.

[3] Andreas Classen. *Modelling with FTS: A Collection of Illustrative Examples*. Tech. rep. P-CS-TR SPLMC-00000001. PReCISE Research Center, Univ. of Namur, 2010.

[4] Paul Clements and Linda Northrop. *Software Product Lines: Practices and Patterns*. Addison-Wesley Professional, 2001.

[5] A. Fantechi and S. Gnesi. "Formal Modeling for Product Families Engineering". In: *Software Product Line Conference, 2008. SPLC '08. 12th International*. Sept. 2008, pp. 193–202.

[6] Alexander Gruler, Martin Leucker, and Kathrin Scheidemann. "Modeling and Model Checking Software Product Lines". In: *Formal Methods for Open Object-Based Distributed Systems: 10th IFIP WG 6.1 International Conference, FMOODS 2008, Oslo, Norway, June 4-6, 2008 Proceedings*. Ed. by Gilles Barthe and Frank S. de Boer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 113–131.

[7] Arne Haber et al. "Delta Modeling for Software Architectures". In: *CoRR* abs/1409.2358 (2014).

[8] Jochen Kamischke, Malte Lochau, and Hauke Baller. "Conditioned Model Slicing of Feature-annotated State Machines". In: *Proceedings of the 4th International Workshop on Feature-Oriented Software Development*. FOSD '12. Dresden, Germany: ACM, 2012, pp. 9–16.

[9] J. Kramer et al. "CONIC: an integrated approach to distributed computer control systems". In: *Computers and Digital Techniques, IEE Proceedings E* 130.1 (1983), pp. 1+.

[10] Sascha Lity. "Konzeption und Evaluation eines delta-orientierten modellbasierten Testverfahrens für Softwareproduktlinien". MA thesis. TU Braunschweig, 2011.

[11] Sascha Lity, Matthias Kowal, and Ina Schaefer. "Higher-Order Delta Modeling for Software Product Line Evolution". In: *Proceedings of the International Workshop on Feature-Oriented Software Development*. FOSD '16. to appear. Eindhoven, The Netherlands: ACM, 2016.

[12] Sascha Lity et al. *Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study*. Tech. rep. 2012-07. Technische Universität Braunschweig, 2012.

[13] John McGregor. *The Evolution of Product Line Assets*. Tech. rep. CMU/SEI-2003-TR-005. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[14] Tom Mens, Alexander Serebrenik, and Anthony Cleve, eds. *Evolving Software Systems*. Springer, 2014.

[15] Sophia Nahrendorf. "Entwicklung und Modellierung von Evolutionsszenarien für Delta-orientierte Softwareproduktlinien". Projektarbeit, TU Braunschweig. 2016.

[16] Sebastian Oster et al. "Pairwise Feature-interaction Testing for SPLs: Potentials and Limitations". In: *Proceedings of the 15th International Software Product Line Conference, Volume 2*. SPLC '11. Munich, Germany: ACM, 2011, 6:1–6:8.

[17] Klaus Pohl, Günter Böckle, and Frank J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[18] Ina Schaefer. "Variability Modelling for Model-Driven Development of Software Product Lines". In: *VAMOS*. 2010.

[19] Ina Schaefer et al. "Software Diversity: State of the Art and Perspectives". In: *STTT* 14.5 (2012), pp. 477–495.

Technische Universität Braunschweig
Informatik-Berichte ab Nr. 2012-03

2012-03	J. Rang	An analysis of the Prothero–Robinson example for constructing new DIRK and ROW methods
2012-04	S. Kolatzki, M. Hagner, U. Goltz and A. Rausch	A Formal Definition for the Description of Distributed Concurrent Components - Extended Version
2012-05	M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies and P. Wähnert	Efficient low-rank approximation of the stochastic Galerkin matrix in tensor formats
2012-06	S. Mennike	A Petri Net Semantics for the Join-Calculus
2012-07	S. Lity, R. Lachmann, M. Lochau, I. Schaefer	Delta-oriented Software Product Line Test Models - The Body Comfort System Case Study
2013-01	M. Lochau, S. Mennicke, J. Schroeter und T. Winkelmann	Extended Version of 'Automated Verification of Feature Model Configuration Processes based on Workflow Petri Nets'
2013-02	S. Lity, M. Lochau, U. Goltz	A Formal Operational Semantics of Sequential Function Tables for Model-based SPL Conformance Testing
2013-03	L. Giraldi, A. Litvinenko, D. Liu, H. G. Matthies, A. Nouy	To be or not to be intrusive? The solution of parametric and stochastic equations – the "plain vanilla" Galerkin case
2013-04	A. Litvinenko, H. G. Matthies	Inverse problems and uncertainty quantification
2013-05	J. Rang	Improved traditional Rosenbrock–Wanner methods for stiff ODEs and DAEs
2013-06	J. Koslowski	Deterministic single-state 2PDAs are Turing-complete
2014-01	B. Rosić, J. Diekmann	Stochastic Description of Aircraft Simulation Models and Numerical Approaches
2014-02	M. Krosche, W. Heinze	A Robustness Analysis of a Preliminary Design of a CESTOL Aircraft
2014-03	J. Rang	Apdative timestep control for fully implicit Runge–Kutta methods of higher order
2014-04	S. Mennicke, J.-W. Schicke-Uffmann, U. Goltz	Free-Choice Petri Nets and Step Branching Time
2014-05	A. Martens, C. Borchert, T. O. Geissler, O. Spinzcyk, D. Lohmann, R. Kapitza	Exploiting determinism for efficient protection against arbitrary state corruptions
2014-06	J. Rang	An analysis of the Prothero–Robinson example for constructing new adaptive ESDIRK methods of order 3 and 4
2014-07	J. Rang, R. Niekamp	A component framework for the parallel solution of the incompressible Navier- Stokes equations with Radau-IIA methods
2014-08	J. Rang	The Prothero and Robinson example: Convergence studies for Runge–Kutta and Rosenbrock–Wanner methods
2014-09	J. Wayetens, B. V. Rosic	Comparison of deterministic and probabilistic approaches to identify the dynamic moving load and damages of a reinforced concrete beam
2014-10	B. Rosic, J. Sykora, O. Pajonk, A. Kucerova, H. G. Matthies	Comparison of Numerical Approaches to Bayesian Updating
2018-01	S. Nahrendorf, S. Lity, I. Schaefer	Applying Higher-Order Delta Modeling for the Evolution of Delta-Oriented Software Product Lines

Institute for Software Engineering and Automotive Informatics
Müehlenpfordtstr. 23
38106 Braunschweig