APPENDIX G

A Cruise Control System

G.1 Case Description

We will design a simple cruise control system (CCS) for a car with an automatic transmission. The CCS must be able to maintain the current speed of the car and to accelerate the car upon the request of the driver. It must also measure the current car speed and display this to the driver.

The CCS receives signals from several sensors.

- A rotation sensor connected to the front axle generates a fixed number of pulses per rotation of the axle. The frequency varies from 0 to 1500 Hz.
- A timer generates a pulse every 100 msec.
- A sensor connected to the brake pedal sends a signal when the pedal is pressed and another signal when it is released.
- A sensor connected to the gear box sends a signal when the box switches to its highest gear and another signal when it leaves the highest gear.
- A sensor connected to the engine sends signals when the engine switches on and off.
- There are five buttons on the dashboard to operate the cruise control: Start cruising, Stop cruising, Start accelerating, Stop accelerating, and Resume.

In addition the CCS is connected to an actuator and a display.

- An actuator connected to the throttle can put the throttle valve in any position, indicated by an integer value ranging from 0 (valve closed) to 255 (valve fully open). The driver can put the throttle in any of these positions too by using the accelerator pedal. If both the actuator and the pedal try to set the throttle in a certain position, the throttle will set itself to the largest of the two values. This is a property of the throttle that can be assumed by the CCS.
- The CCS can send a value representing a speed to a display on the dashboard, to be displayed to the driver.

G.1.1 *Cruise Control*

The cruise control function allows the driver of a car to maintain speed without pressing the accelerator pedal. The driver can switch on the cruise control by pressing the Start cruising button on the dashboard of the car. The cruise control will switch on only when the engine is running, the car is in its highest gear, the speed is at least 40 km/h, and the brake pedal is not depressed. When the cruise control is switched on, the CCS sets the desired speed to the currently measured speed and then attempts to maintain the measured speed within a margin of 1.5 km/h of the desired speed.

The Start accelerating button allows the driver to accelerate smoothly without pushing the accelerator pedal. When this button is pushed and the cruise control is on, the car will accelerate smoothly, keeping within 0.25 and 0.4 m/sec^2, until the Stop accelerating button is pushed.

When the cruise control is on, the driver can also accelerate by pushing the accelerator pedal. When the driver releases the accelerator pedal, speed returns to the speed set before the pedal was pushed or to the most recent speed achieved by pushing the Start accelerating button.

When the cruise control is on, the CCS must change the position of the throttle slowly enough that it will take at least 10 seconds from the moment the throttle is fully closed to the moment it is fully opened, or the reverse.

If the driver pushes the brake pedal when the cruise control is on, then the cruise control will switch itself off immediately. The same happens when the driver switches from the highest gear to a lower gear or the engine switches off. In all these cases, the throttle should close itself completely immediately, rather than in a smooth manner by small steps. If the driver pushes the Resume button when the cruise control is still on and the brake pedal is released, the car drives in its highest gear, and the speed is at least 40 km/h, the cruise control will then take the car back to the speed set before it was switched off.

If the driver presses the Stop cruising button, the cruise control switches off and closes the throttle smoothly. The Start accelerating and Resume buttons will have no effect until the cruise control is switched on again.

G.1.2 *Measurement*

The distance driven by the car is measured by comparing the number of times the front axle has turned to the number of rotations per kilometer, called the calibration factor. The speed is obtained by dividing distance by time. The CCS has to compute the speed regularly and display it on the dashboard. For the driver's comfort, it is necessary to compute the speed of the car every second. If this period were longer, the CCS would cause the car to change speed jerkily.

G.1.3 *Calibration*

The distance traveled by a car when the front axle turns once depends on the size and wear of the tires. The factory sets the calibration factor to an initial value. A

mechanic can (re)set this value as follows. Three buttons under the hood of the car signal the start, stop, and cancellation of calibration. After the start button is pushed, the mechanic drives the car for exactly one kilometer and presses the stop calibration button. The CCS then stores the calibration factor to convert the number of rotations of the front axle into a measure of distance. If the mechanic makes a mistake, he or she can push the third button, cancel calibration. Calibration can only be done when cruise control is off and vice versa.

The following cruise control description is based on a case given by Awad et al. [1] and on solutions designed by David Jansen [2, 3].

G.2 Context

The system boundary is easy to determine because we merely need to list all the devices that the CCS interfaces with and the interfaces of these devices to the CCS. So we start with the context diagram, shown in Figure G.1.

To avoid cluttering up the diagram, we distinguish only two types of buttons. These are partitioned into their subtypes in Figure G.2. This is part of the subject domain ERD; the rest of the subject domain ERD is not interesting because all other subject domain entity types are visible as external entities in the context diagram already. Moreover, each subject domain entity type has exactly one instance: There is exactly one brake, one button of each type, one axle sensor, and so on. The rest of the subject domain ERD is therefore not shown.

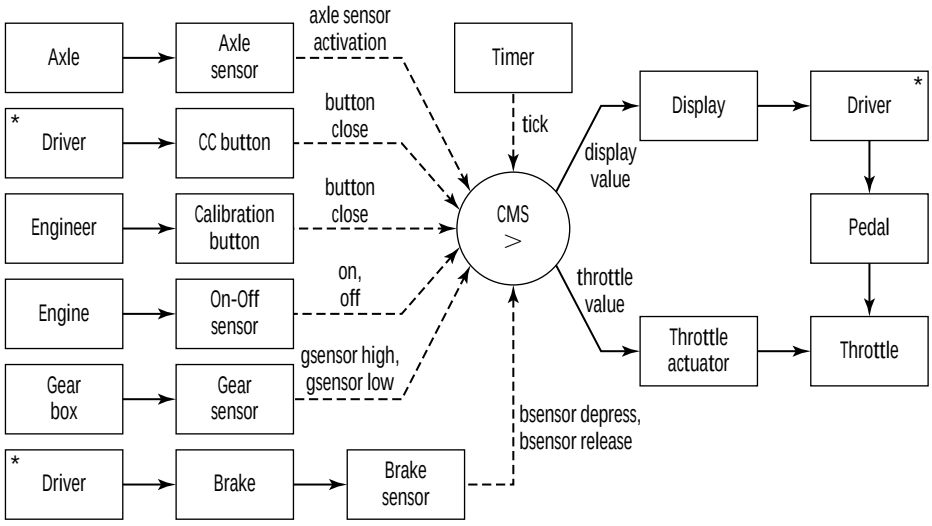

Figure G.1 Context diagram of the CCS. Duplicated nodes are labeled with an asterisk (*). Bold arrows represent flow of material or energy. The other arrows represent flow of information (data or events).

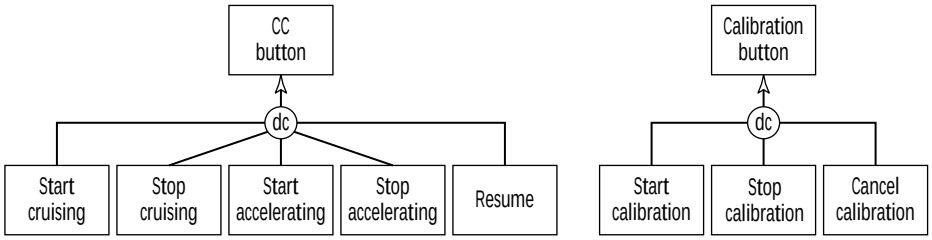

Figure G.2 Classification of buttons in the subject domain.

Name: Car Management System

Acronym: CCS

Purpose: Maintaining and displaying the speed of a car.

Responsibilities:

- Control the speed of the car.
- Calibrate speed measurement.
- Measure and display speed.

Exclusions:

- The CCS does not test whether the current or desired speed is safe not whether it is allowed (not higher than the maximum speed on this part of the road).
- The CCS does not change gears, not even when this would be appropriate (e.g. when driving uphill or downhill).
- The CCS does not brake, even when this would bring the current speed closer to the desired speed (e.g. when driving downhill).

Figure G.3 Mission statement of the CCS.

G.3 Mission Statement and Functions

Figures G.3 and G.4 show the mission statement and function refinement tree of the CCS. The responsibilities appear as children of the mission in the function refinement tree. They are groups of functions that are managed independently by the CCS. The leaf functions of the tree are atomic responses triggered by a stimulus. I have identified the definitive responsibilities and leaf functions only after naming a first version of the various models. For example, we will see that the leaf functions indicate atomic transitions in the statechart of the controller.

G.4 Requirements-Level Architecture

The desired behavior of the CCS can be represented by an STT that relates stimuli and the current CCS state to responses and the next CCS state. However, the state

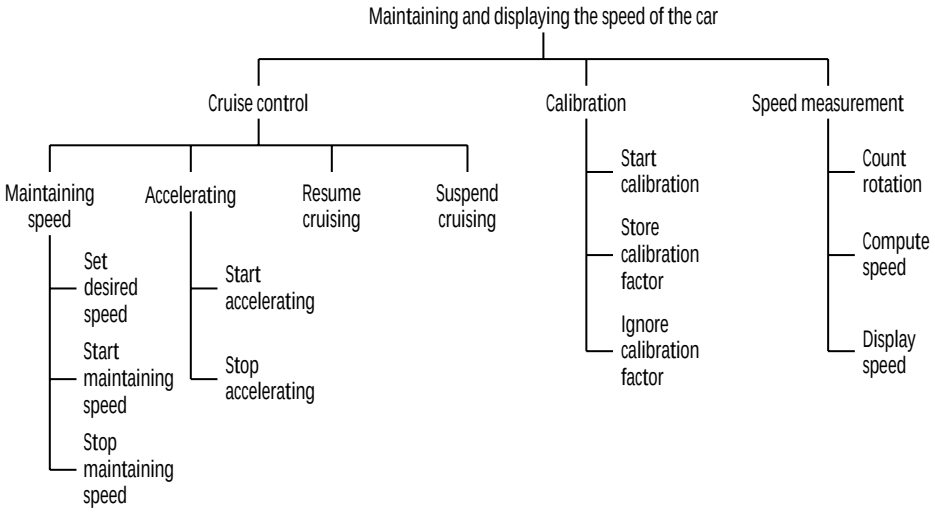

Figure G.4 Function refinement tree of the CCS.

of the CCS has a hierarchical structure that is well suited to representation by a statechart, so we represent the desired CCS behavior by a statechart.

In addition, stimuli and responses need some management by interface objects in the CCS, for example, to check the sanity of input signals or maintain a representation of the state of a sensor. So we also need objects in the CCS to manage the interfaces with sensors and actuators. This leads to a requirements-level architecture structured by device-oriented decomposition and by functional decomposition. The architecture diagram in Figure G.5 shows the following components (see Section 19.4 for these decomposition guidelines):

- Device components, some of them (stateless) transformations, some of them (stateful) objects.
- Functional components, namely the calibrator, cruiser, coordinator, and speed computation.
- A behavioral component, namely the coordinator (which has also been identified by functional decomposition).

The flows press calibration buttons and press cruise buttons in Figure G.5 are composite. They contain one event flow for each button. So within press cruise buttons there is, for example, an event flow press(Start cruising), which is the event that the Start cruising button is pressed.

The components of the requirements-level architecture are defined in Figure G.6. Note that:

- To indicate the difference and relationship between devices and the software objects that deal with them, the software object corresponding to device D is called D S.

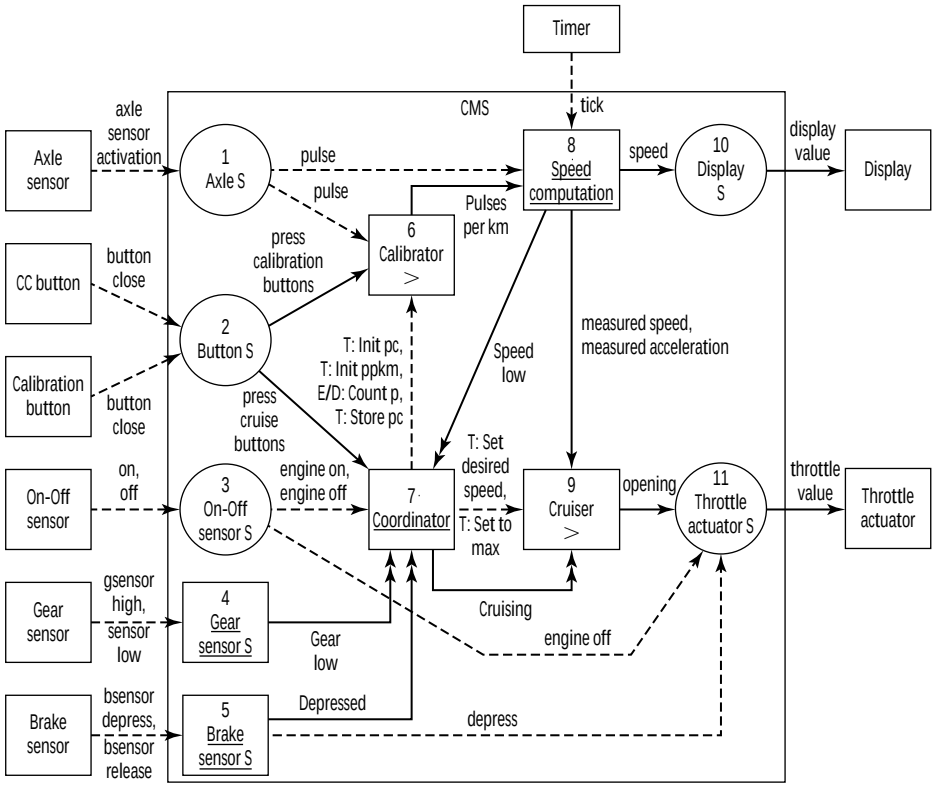

Figure G.5 Requirements-level architecture of the CCS.

- The Brake sensor S and Gear sensor S objects only function in a desirable way if two assumptions are satisfied, namely that when the CCS is started, (1) the gear is low and (2) the brake is not depressed. This makes the CCS unsafe. In a safer version, we need a sensor that senses the initial states of the gear and brake and we need software objects that initialize the Gear sensor S and the Brake sensor S accordingly. This is an easy addition that does not change the requirements-level architecture fundamentally.
- Speed computation is done every second, as required.
- The Throttle actuator listens to the Cruiser unless it is overriden by the brake, as required.

The Cruiser and Calibrator are defined in Figures G.7 and G.8. Component definitions are listed in Figure G.9. Note that:

- Init ppkm initializes the calibration factor Pulses per km to a value remembered from the most recent calibration. This value must be stored even when the CCS is off, and we do not specify its source here. It could be stored in some hardware device that maintains this value even when the engine is off and the battery is dead.

1: Axle S. Data transformation that accepts axle sensor activitation signals and outputs pulses, where each pulse is supposed to correspond to one rotation of the axle to which the sensor is connected.
2: Button S. This stands for a set of data transformations, one for each CC buttons and Calibration buttons (Figure G.2). Each transformation accepts button close signals and outputs press(button identifier) events, where for each data transformation, the button identifier is the identifier of the button to which this transformation is connected.
3: On-Off sensor S. Data transformation that accepts on and off signals from the On-Off sensor and outputs engine on and engine off events.
4: Gear sensor S. Software object that maintains the output Boolean variable Gear low. • Gear low is initialized to True. It is set to True when the object receives the signal Gear low and it is set to false when the object receives the signal Gear high. • **Assumption**: The gear is low when the CCS is started.
5: Brake sensor S. Software object that maintains the output Boolean variable Depressed and that accepts signals bsensor depress and bsensor release from the brake sensor. • Brake sensor S maintains a Boolean output variable Depressed, which is set to True when the software object receives a bsensor depress signal, and it is set to False when it receives a bsensor release signal. Depressed is initialized to False. • When it receives a bsensor depress signal, it produces a depress event. • **Assumption:** The brake is not depressed when the CCS is started.
6: Calibrator. See Figure G.7.
7: Coordinator. See Figure G.10.
8: Speed computation. • Software object that accepts pulse events and reads the variable Pulses per km. • Every second, it computes the average speed and acceleration in the previous second and stores these in two local variables called measured speed and measured acceleration. • It outputs the measured speed value as the variable speed. • It maintains a Boolean output variable Speed low that is True when the measured speed $<$ 40 km/h and False when measured speed $\geqslant$ 40 km/h.
9: Cruiser. See Figure G.8.
10: Display S. Data transformation that accepts a value of the variable speed and outputs a value that would set the display to represent the speed value.
11: Throttle actuator S. Data transformation that accepts a value for the desired throttle opening and the depress event. • If the depress or engine off events occur, the throttle opening is set to 0. • Otherwise, the throttle opening is set to the desired opening.

Figure G.6 CCS component definitions.

Note that this presupposes that the factory does a calibration, or in any case sets the calibration factor to a reasonable value, before the CCS is ever switched on.

- The coordinator must ensure that these data transformations are sequenced properly. For example, Init pc must be performed before Count pulses.
- If the coordinator is in state Cruising, the Throttle control computes the desired throttle opening. Computing the throttle opening according to the three constraints is a control theory problem whose solutions are known by automotive

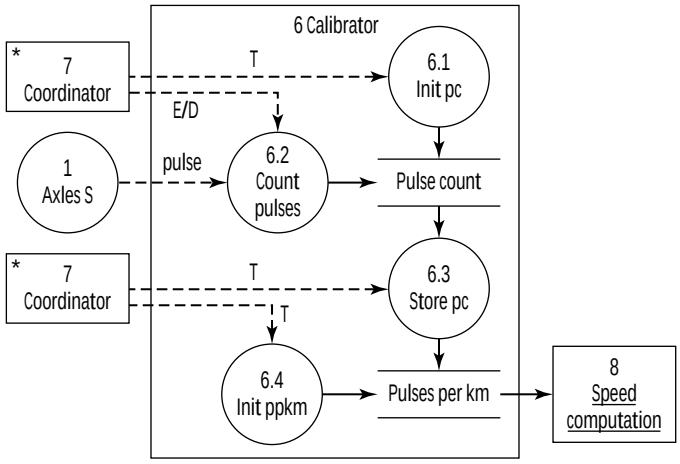

Figure G.7 The Calibrator subsystem (asterisks indicate duplicate diagram elements).

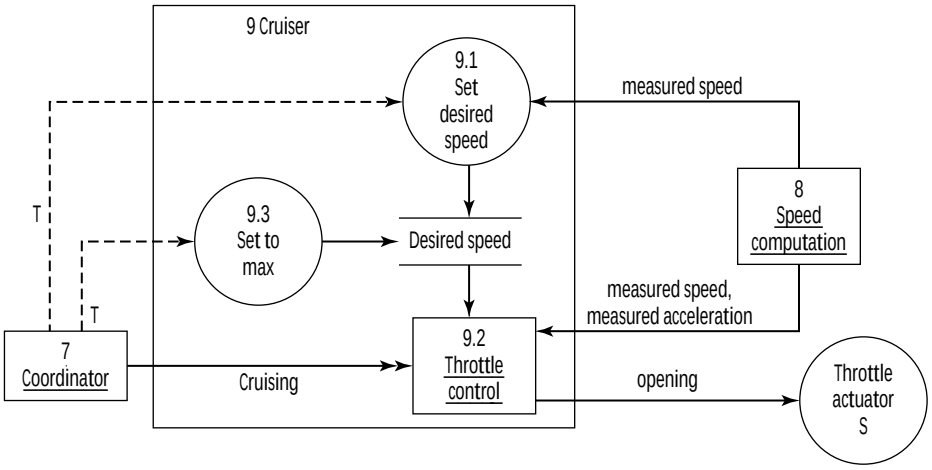

Figure G.8 The Cruiser subsystem.

engineers. (We will not discuss these solutions here.) If the coordinator is not in state Cruising, then it must smoothly reduce the opening value of the throttle so that the driver takes over control of the throttle by means of the accelerator pedal. This is done in steps of 25.5 points per second. This agrees with the requirement that it should take at least 10 seconds to change throttle position from fully open (255 points) to fully closed (0 points).

- The transformation Set to max sets the desired speed to a maximum value. This will cause the CCS to set the throttle in accelerating position.

6.1: Init pc. Data transformation. When triggered, sets Pulse count to 0.

6.2: Count pulses. Data transformation. When enabled, it responds to the reception of a pulse by adding 1 to the contents of Pulse count.

6.3: Store pc. Data transformation. When triggered, it sets Pulses per km to Pulse count.

6.3: Init ppkm. Data transformation.

- When triggered, it sets Pulses per km to an initial value retrieved from persistent memory (not shown here).
- **Assumption:** Whenever the cruise control is activated, there has been a calibration earlier in the life of the car.

9.1: Set desired speed. Data transformation. When triggered, sets Desired speed to measured speed.

9.2: Throttle control. Software object with a local variable *V* that contains the value of the most recently set throttle valve opening. The input variable Cruising indicates whether the Coordinator is cruising.

- The initial value of *V* is estimated based upon the currently measured speed.
- Do the following every second.
  - If the Coordinating is in state Cruising, compute the desired throttle position as a function of measured speed and measured acceleration according to the following three constraints:
    * The speed should deviate less than 1.5 km/h from the desired speed;
    * Any acceleration should be in the range [0.25 m/sec^2, 0.4 m/sec^2] and any deceleration should be in the range [−0.4 m/sec^2, −0.25 m/sec^2].
    * The throttle should not change more than 25.5 points per second.

    Assign the resulting value to *V*, and output this value opening.
  - If the Coordinator object is not in state Cruising, output an opening that is 25.5 points less than the previous one.

9.3: Set to max. Data transformation. Sets the desired speed to some predetermined maximum.

Figure G.9 Calibrator and Cruiser component definitions.

So far, the functions of the CCS have been described independently of their behavior. Figure G.10 describes the behavior of the coordinator. The state hierarchy (Figure G.11) reflects the functionality of the CCS (Figure G.4). The state hierarchy shows that when the coordinator is on, the CCS is either calibrating, cruising, or idle. Speed measurement is an activity going on all the time, even when the coordinator state is off. When cruise control is active, the CCS may either be cruising or suspended; when it is cruising, it either maintains currently measured speed or accelerates the car.

The statechart in Figure G.10 shows that when entering the state On, the coordinator initializes the calibration factor Pulses per km and enters the Idle state. Entering the Idle state, it initializes the Pulse count to 0. The coordinator then can move to the Calibrating or to the Cruise control state, but not to both at the same time. This is as required. Note that we assume that the Start calibration and Start cruising buttons

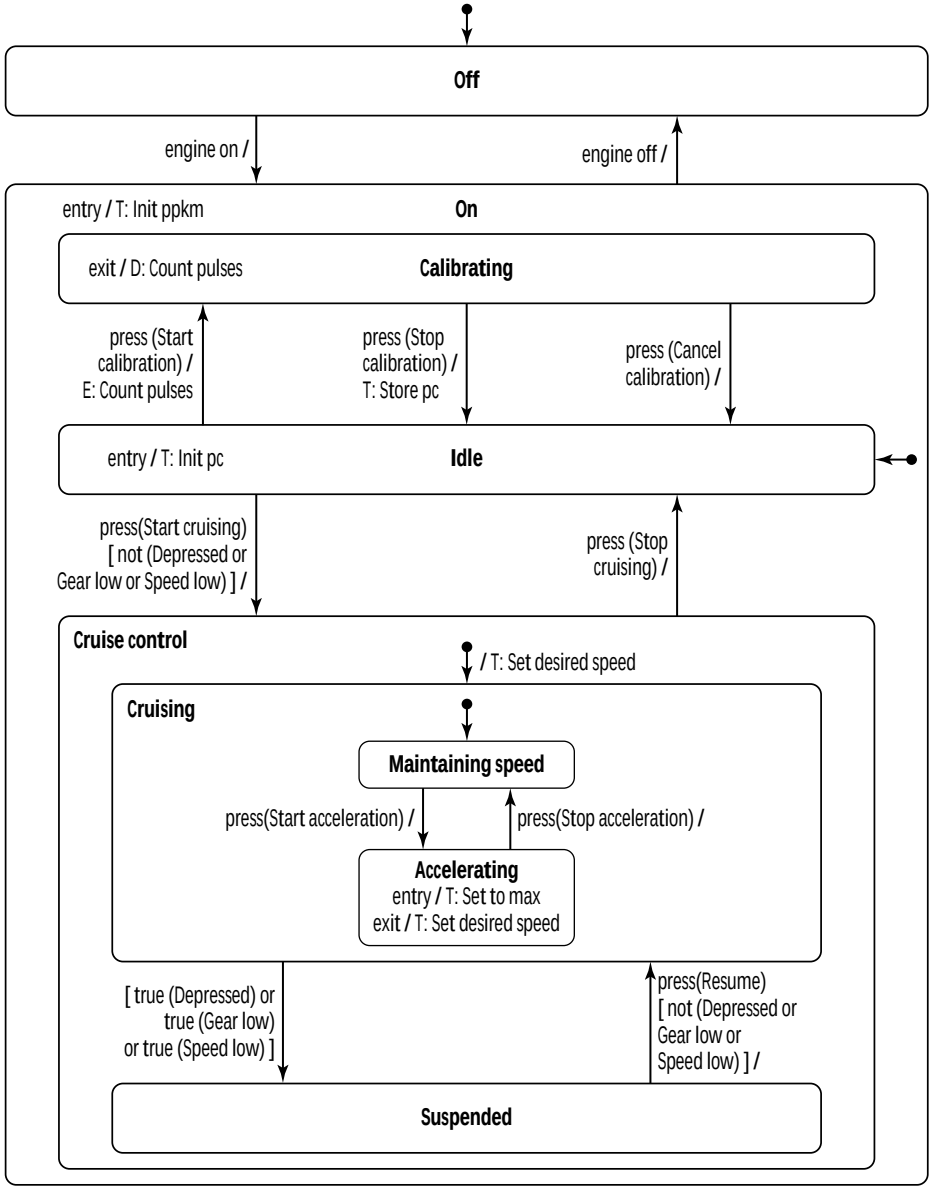

Figure G.10 State-based behavior of the CCS.

are never pushed simultaneously. If they were, the behavior of the coordinator is nondeterministic. This is a reasonable assumption.

Upon entering Calibration, Count pulses is enabled. This transformation is enabled only when the coordinator is in state Calibrating and disabled otherwise. Upon successful termination, the resulting calibration factor is stored in Pulses per km.

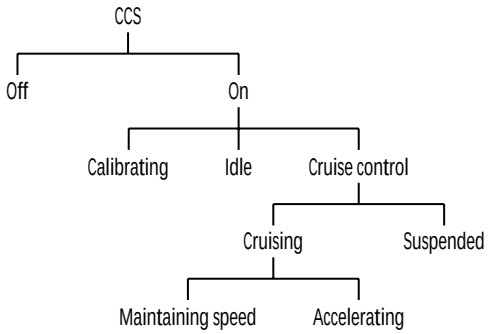

Figure G.11 State hierarchy of the CCS.

Upon entering Cruise control, the coordinator sets the desired speed to the currently measured speed and enters the Cruising state. This enables the Cruiser. This subsystem remains enabled as long as the coordinator is in state Cruising and disabled otherwise. When the coordinator is in state Cruising, the Cruiser maintains current speed. When the coordinator enters the substate Accelerating, the desired speed is set to some maximum, which causes the cruiser to accelerate until that maximum is reached. If the coordinator exits the Accelerating state, the desired speed is set to the speed then measured.

If during cruising, the brake is depressed, the gear is switched low, or the speed falls below 40 km/h, the Cruiser is disabled and the coordinator enters the state Suspended. If press(Resume) occurs and the conditions are met, the Cruiser is enabled again.

At all times, if an engine off event occurs, the coordinator enters the Off state. This may involve any of the exit actions in the diagram, such as disabling Count pulses, disabling the Cruiser, and setting the desired speed.

G.5 Assumptions

We have made a number of assumptions, which are summarized and numbered in Figure G.12. Analyzing these we identify some possibilities for improvement of the controller design. A1 is crucial and must be guaranteed by the car manufacturer. The cruise controller can do nothing to ensure this assumption. Assumptions A2 and A3 are unsafe and must be eliminated by adding sensors that measure the initial position of the gear and brake. Assumption A4 is crucial, but the cruise controller can do nothing to ensure it. However, it could check it with the help of the environment, for example, by adding a device that is set when the controller is calibrated and whose state survives periods when the engine is off or the battery is dead. A4 must be guaranteed by the business procedures followed at the factory before the car is released to its first buyer. Assumption A5 is harmless and does not need to be checked.

A1 The throttle sets itself to the largest value of (Accelerator-pedal-generated throttle position, CCS-generate throttle position).

A2 The gear is low when the CCS is started.

A3 The brake is not depressed when the CCS is started.

A4 Whenever the cruise control is activated, there has been a calibration earlier in the life of the car.

A5 Start calibration and Start cruising are never pressed simultaneously.

Figure G.12 Assumptions about the environment.

A further improvement is the addition of a sensor to measure the current throttle opening. This can be used to improve the initialization of the Throttle opening object.

References

1. Awad, M., J. Kuusela, and J. Ziegler. (1996). *Object-Oriented Technology for Real-Time Systems: A Practical Approach Using OMT and Fusion*. Prentice-Hall.
2. Jansen, D.N. (1999). *Statemate Solution of the Cruise Control Problem*. Technical report, Department of Computer Science, University of Twente, May.
3. Jansen, D.N. (2001). *Cruise Control Specification Using Rhapsody*. Technical report, Department of Computer Science, University of Twente, December.