

Car Control System: A Top-Down Approach

Presented by

Ravi Shankar

Director, Center for Systems Integration

www.csi.fau.edu

ravi@cse.fau.edu, 416 Science & Engr Bldg

Reference

Chapter 6 from:

Embedded Real Time Systems: A Specification
and Design Methodology

By

Jean Paul Calvez

Wiley, 2006

Note: US terminology and units are a bit different!

Our Story

- Motorola – 6 years, \$1.15 M, Multi-Disciplinary Group at CSI
- One Pass to Production – 24 Months to 24 Hours
- Top-down, Bottom-up, and Middle-out methodology
- Top-down: Abstraction to realization
- Bottom-up: Component Based Design
- Middle-out: Quality of Service based Optimization

Why This Presentation?

- Most engineering professionals think bottom-up.
 - *This extends product development time 2 to 3 times. System speed today allows easy top-down design – optimization is less of an issue*
- Bottom-up was reasonable when systems were simple and sequential. Today's systems are complex and concurrent.
 - *Systems can fail in mysterious ways. Trees in the forest...*
- Yesterday's simple systems are building blocks for today's complex systems.
 - *Don't reinvent the wheel – Others will ship product before you do.*
- Even a well engineered product which does not meet user needs is DOA.
 - *Listen to the user's needs – you need the user to buy your product.*



Car Control System: User Requirements (See Fig 6.5)

- Manage cruise control *(Translate to US Terminology)*
 - “Obviously” inactive when engine is started – pg. 64
 - Use **Activation** button to cruise, if speed > 50 Km/h - controls regulator valve
 - Press **Brake** or **Accelerator** pedal to exit cruise control
 - Return to cruising speed with **Back_To_Previous_Speed**; **Stop** cancels Return
 - Use bistable **Acceleration** button to Start and End **Acceleration**, when regulator is active, to gradually increase speed automatically.

Car Control System: Requirements (continued)

- Monitor fuel consumption and speed
 - Determine # of tire rotations for known distance – with **Start_Km_Measurement** and **Stop_Km_Measurement**
 - (Measure tire rotations with 10 pulses per rotation) – *Estimate in the class*
 - Display speed, and average speed (AS) after **Start_Run** with **Request_AS**.

Car Control System: Requirements (continued)

- Help with maintenance of the vehicle
 - Use [Add_Fuel](#) to inform system about the fuel amount just added when tank is filled. System calculates and displays the consumption between two fillings
 - Monitor the kilometers run by the vehicle and inform the driver when maintenance is due, at 7,500 Km; 15,000 Km; and 30,000 Km.
 - Slow flashing message at -250 Km; constant message at -50 km; and clear message by pressing the [Maintenance_Carried_Out](#) button.

Derived Requirements

(section 6.1.5)

- Both Accelerator pedal and Acceleration control simultaneously influence electronic valve control (for fuel injection). Higher speed demand is higher priority
- Valve control: 0 to 8 V, for fully closed to fully open – continuous control
- Cruise control: Adjust valve opening to maintain speed, up to fully open; close if the speed $>+3$ Kmph; and update once per second “to avoid stability problems.”

Derived Requirements: Continued (section 6.1.5)

- >10 seconds to fully open, to prevent “too sudden acceleration.”
- Fast closing at any speed is OK (?)
- During **Acceleration**, maintain acceleration at 2 Kmph/s. Close valve if 3 Kmph/s. Open valve fully if 1 Kmph/s. Otherwise, proportional opening.
- User interface (UI) with function keys and LCD screen
- “Intelligent” UI – all magnitude inputs must be validated (fuel quantity, etc.) – *e.g., Can't add 900 liters!*

Engineering Specifications:

6.2.1 Model the environment

- See Figure 6. 1:
 - External Actors – The driver (and calibrator) and the vehicle with its engine and tires. *Note the ‘controls’ and ‘observes’ relationships – Figure 6.1*
 - Events to trigger on – by driver/calibrator – start and stop vehicle; regulator activation and stop; return to previous speed; Brake and Accelerate; Start and Stop Km measurement; Start Run, get AS (average speed); Maintenance done; and Add fuel
 - *UML – can help much – can make explicit the knowledge base that may be implicit. But could not have covered the example in one lecture*

6.2.1 Model the environment (Continued)

- Model the environment - continued
 - Conditions / constraints to be met: From vehicle: $S > 50$ Kmph; $S == SR$, the set cruising speed for cruising; and Car in D (Drive) gear; From driver: Fuel Quantity; Internal: 10 seconds for full range of valve.
 - Observations to be displayed: Distance covered (D); Vehicle Speed (S); Average Consumption (avC); Flashing display at -250 Km; and Continuous display at -50 Km.

6.2.1 Model the environment : (Continued)

- Entity Behavior: Need simple behavior models of the driver/calibrator and the vehicle.
 - Driver generates events and observes vehicle's reactions and info.
 - Vehicle – Describe speed in terms of Accelerator pedal position (P_Accelerator), electronic control (Ctl_Valve), road slope, vehicle load, etc. Higher difference between P_Accelerator and Ctl_Valve has higher priority

6.2.1 Model the environment: (Continued)

- System delimitation (see Fig 6.1)
 - Driver controls via events listed earlier
 - Driver observes via list of observations given earlier – but add display of “fuel quantity added.”
 - Vehicle is controlled by electronic control of valve position.
 - Vehicle is observed via its speed, distance traveled, and gear-in-drive mode.

6.2.2 Functional Specifications

- Abstract or indirect descriptions of system functions, not actual code.
- 3 Functions: Regulate vehicle speed at cruising speed, as needed; provide driving info; and provide maintenance help.
- Speed Regulation: Figure 6.2 for vehicle behavior
- Driving Info: Figure 6.3 for driver behavior
- Maintenance Help: Figure 6.4: driver and system behavior

6.2.3 Operational and Technical Specifications

- Interfaces – With Vehicle – incremental counter for S and D; Boolean for gear and braking; continuous control for valve.
- Interfaces – with User – LCD, Keys, +/- keys to input fuel quantity. Get avC by pressing the “add fuel” key – See Figure 6.6
- Timing – For stability, valve control step for < 1s, response to user input ~1s; brake – immediate
- Implementation – Use MCU, battery powered; non-volatile memory to save some info (D, Calibration, ..)

6.3 Functional Design

- Define system boundaries, abstract / high level design, technology independent
- All commands are messages – that is, no handshaking; Same for Display. *Advantages*
- Define mode relations – synchronization, state variables, message transfer.
- System inputs and outputs: Page 74
- Flashing alternates “Maintenance” and “Null”

6.3.2. Initial Functional Structure

- Phases: Analysis, System Decomposition, Solution Construction, and Validation.
- Analysis: Three FSMs shown in Figures 6.2, 6.3, and 6.4 are independent of each other. System internal variables used: SR, DV, TV, DC.
- System Decomposition: Supervision is driver related, not persistent; Control is vehicle related and has states.
- Solution Construction: Figure 6.8. *Buffer...*

6.3.2.B Solution Construction: Figure 6.8

- Periodic Step activates Speed-Control
- Modes of operation: Stop, Regul, Accel. In figure 6.2 – Regul for states (1) and (2); Accel for state (3); and stop for others.
- CTL list on pg. 74. Also add Brake and Gear position. All external events are messages.
- Validation and Improved Version – Figure 6.9
- Refinement – Speed_Control activated by 1 second STEP; Supervision is sequential activated by CTL message – incorporates 3 FSMs.

6.3.4 Speed-Control Function

- States (1) and (2) of Figure 6.2 are one mode/state: Regul. Internal transition between the two phases is via (S # SR).
- Figure 6.10
- High level code: Page 78-79

6.3.5 Supervision Function

- 3 FSMs – simpler because of state sequence implementation in Speed-Control
- High level code on pages 80-81

6.3.6 Maintenance and Time Generation Functions

- High level code on pages 81-82 for maintenance
- High level code on page 82 for time generation

6.4 Implementation Specification

- Bring in technology dependencies; Hardware-software partitioning
- S and D replaced with Imp (for Impulses) – Figure 6.12 and High level code on page 85.
- 9 parallel functions implemented in ‘sequential’ machine. Hardware interrupts. Figure 6.16 for prioritization.
- Both C and IMP are timed interrupts. See Figure 6.16. NB_IMP in code on page 85 is shared. Needs **mutual exclusion** to access. Good practice, but needed?
- PWM for Ctl_Valve implementation
- Timing checks – Page 89, bottom. Will meet

Conclusions

- Stages: Requirements, Specifications, Functional Design, and Implementation
- Section 6.5, page 91: Customer has 3 up front questions – Development time, Development cost, and production cost.
- The analysis shown here helps you address these questions and develop the product within these constraints!