SoftContract: an Assertion-Based Automotive Software Development Process that Enables Design-by-Contract

Jean-Yves Brunel, Paolo Giusto, Luciano Lavagno
Cadence Design Systems
Marco Di Natale
Scuola Superiore Sant’Anna, Pisa, IT
Alberto Ferrari
PARADES GEIE, Roma, IT
Outline

• Automotive software design flow
• Design roles and interactions
• Model-based design and non-functional constraints
• Contract-based design
• Logic Of Constraints
Challenges and Opportunities

• “More than 90% of the innovation in the car will come from Electronics” (Daimler-Chrysler)
• “More than 20% of the cost of the car is by now due to Electronics” (BMW, PSA, ARM)

• Trends:
  – Reduce cost of mechanics by means of electronics
  – Car manufacturers bring in house electronic competence
  – System integration becomes increasingly complex
  – Scenarios with purely software subsystems become likely
Virtual System Integration

Requirements

Function/System Analysis

System Design Partitioning

Virtual Integration

Virtual Vehicle Test

Prototype Test

Comm.-test Subsystem

SW Design Specification

SW Integration

Implementation

Source: Thilo Demmeler, BMW
Virtual Car Design

ASCET Project
Module 1
Module 2

Message1

VCC Architectural Diagram
ECU1

CPU
Memory
Scheduler
Bus Controller
Bus Controller

ASCET Project
Module 1
Module 2

Message1

VCC Behavioral Diagram
ECU3

ASCET Project
Module 1
Module 2

Message1

VCC Architectural Diagram
ECU2

CPU
Memory
Scheduler
Bus Controller
Bus Controller

ASCET Project
Module 1
Module 2

Message1

VCC Architectural Diagram
ECU4

CPU
Memory
Scheduler
Bus Controller
Bus Controller

© Cadence Design Systems
Virtual Car Design
Design Roles and Interactions

- **Constraints**
- **System architect**
  - Simple plant model
  - Kernel benchmarks
  - System specification

- **Algorithm+SW developer**
- **Detailed plant models**
- **Abstract ECU architectures**
- **Refined global architecture**
  - Detailed ECU architectures
  - ECU + RTOS + ...

- **Sub-function specifications**
- **Detailed plant models**
- **Abstract ECU architectures**

- **Sub-function implementations**
- **System integrator**

**System implementation**
Roles: architect

• Roughly equivalent to analyst in classical SW development

• Defines functional and non-functional requirements, interacting with customers. Non-functional examples:
  – end-to-end performance (hard and soft RT)
  – reliability, safety, security, EMI

• Defines software (functional components) and platform (RTOS/CPU/network) architecture

• Specifies non-functional constraints of software and platform components (both blocks and communication links)

• Splits the implementation work among developers, defining functional (including high-level tests), interfacing, performance, security, etc. goals for each sub-project
Roles: developer

• Implements each sub-project, while:
  – satisfying constraints from architect
  – considering effects of his own and other software on overall system performance, safety, etc.
• Uses platform configuration from architect (RTOS, CPU, network) and platform models from platform providers (ISS, RTOS, SW estimation models, network models, …)
• Tests his design against functional and non-functional requirements
• Provides detailed implementation to the integrator
Roles: integrator

• Generally belongs to the same car manufacturing company as the architect
• Assembles software and platform implementations from several different providers
• Verifies that interfacing, functionality (tests) and non-functional (performance, safety, ...) constraints work
• Often uses rapid prototyping techniques for performance (orders of magnitude better than simulation)
Outline

• Automotive software design flow
• Design roles and interactions
• Model-based design and non-functional constraints
• Contract-based design
• Logic Of Constraints
Model-based design

- Standard methodology for automotive embedded software development
- Algorithmic model written by control specialist using Simulink, Ascet-SD, ...
- Algorithmic model used as golden model:
  - for validation, possibly using hardware-in-the-loop
  - for manual code development
  - for (semi-)automated code generation and RTOS customization
- Problem: model captures explicitly only *functional requirements*
Examples of non-functional constraints

- **Clean air regulations**
  - specified as a set of properties that must hold at run time
  - satisfied by engine control algorithms
  - implemented as a set of monitors that check properties and log results

- **Real-time constraints**
  - specified as a set of max rate and max jitter constraints
  - satisfied by a set of priorities in RTOS
  - possibly implemented as watchdogs
Contract-based design

- Addresses multi-team design of real-time safety-critical software
- Explicitly states assumptions (expected) and assertions (guaranteed) on input/output behavior
- Model-based implementation of assertion-checking code throughout simulation, prototyping, calibration and production
Contract-based design

Plant

Controller
Contract-based design

- Assumptions (on environment)
- Assertions (on controller)

Plant

Controller
Contract-based design

Assumptions (on environment)

Plant

Assertions (on controller 1 by design team 1)

Assertions (on controller 2 by design team 1)
Contract-based design

Assumptions (on environment)

Plant

Assumptions (on controller 1 by design team 2)
Contr. 1

Contr. 2

Assertions (on controller 2 by design team 2)
Syntax and semantics of the Logic Of Constraints

- Defined over (infinite) sequences of events (updates) of module ports
- Can use arithmetic and logical operators
- Only one index variable \( i \) per formula (implicitly universally quantified)
- Annotations with time, power, value, ... of events
- Examples:
  - maximum rate: \( D[i].t - D[i-1].t \geq 10 \)
  - maximum latency: \( S[i].t < D[i].t + 25 \)
  - maximum derivative
    \( (D[i].v - D[i-1].v)/(D[i].t - D[i-1].t) < 12 \)
- Can define new events:
  event Edge { S[i-1].v != S[i].v }
Example of contract

• Example: requirement on advanced cruise controller
• The vehicle shall not accelerate or decelerate, after reaching the cruising speed, by more than a 0.5 m/s\(^2\)
• Decomposed between cruise-control and engine-control teams into:
  – torque requested by cruise control
  – torque provided by engine control and engine
  – relation between vehicle acceleration and torque
Example of contract

- Assertion about overall design:
  assert FSM.State[i].v = CCon \rightarrow
  Acceleration[i].v < 0.5

- Cruise Control design team:
  - assume CC.TorqueRequest[i].v < 20 \rightarrow
  Acceleration[i].v < 0.5
  assert FSM.State[i].v = CCon \rightarrow
  CC.TorqueRequest[i].v < 20

- Engine control design team:
  - assume FSM.State[i].v = CCon and
  CC.TorqueRequest[i].v < 20
  assert Engine.Torque[i].v < 20
Conclusions

• Electronics drives real progress and innovation in automotive industry
• Complex interaction between architects, implementors and integrators
• Explicit modeling of non-functional constraints as assertion-assumption pairs eases:
  – interaction between roles and companies
  – integration task
  – delivery of purely software functionalities