D-Case Collaboration with Modeling Environment

Demonstration

SysML Model Specification

Table of Contents

version: 1.0
revision: 1

Change History

Modification Date	Modifications
2014/01/06	Created

1 Scope

This document describes the outline of SysML models for Demonstration of D-Case Collaboration with Modeling Environment.

2 SysML

SysML (Systems Modeling Language) describes requirements for system, and structure or behavior of system for specifying, analyzing, designing, and verifying system.

Below URL describes the detail about SysML.

http://www.omgsysml.org/

3 Demonstration

This demonstration has SysML modeling for Cruise Control System of Automotive.

Models are focused on Safety Requirements and Reliability Requirements of Functional Safety.

This model includes 5 kinds of diagrams as below.

1. Use Case Diagram
2. Requirement Diagram
3. Block Definition Diagram
4. Parametric Diagram
5. State Machine Diagram

3.1 Use Case Diagram

Use Case Diagram includes association of System User and Operation.

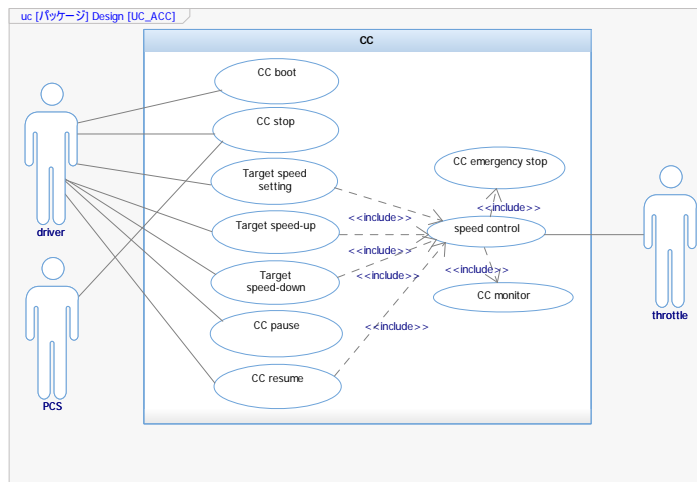It clearly identifies the border of User and system (Figure 1).

Figure 1 Use Case Diagram

3.2 Requirement Diagram

Requirement Diagram includes structure of requirements for System.

Use Case Diagram mainly includes functional requirements, and Requirement Diagram includes both functional requirements and non-functional requirements (Figure 2).
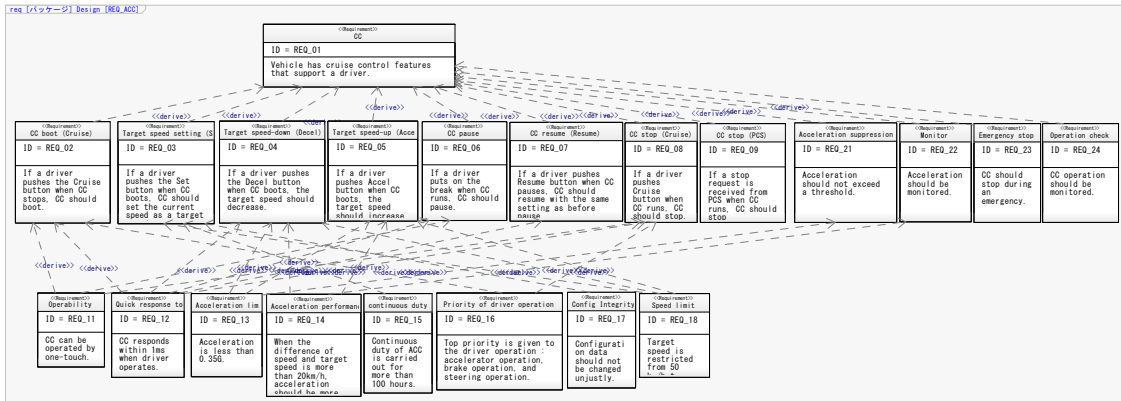

Figure 2 Requirement Diagram

3.3 Block Definition Diagram

Block Definition Diagram includes static structure of System.

It identifies blocks of system components, association of blocks, and hierarchy of blocks (Figure 3).
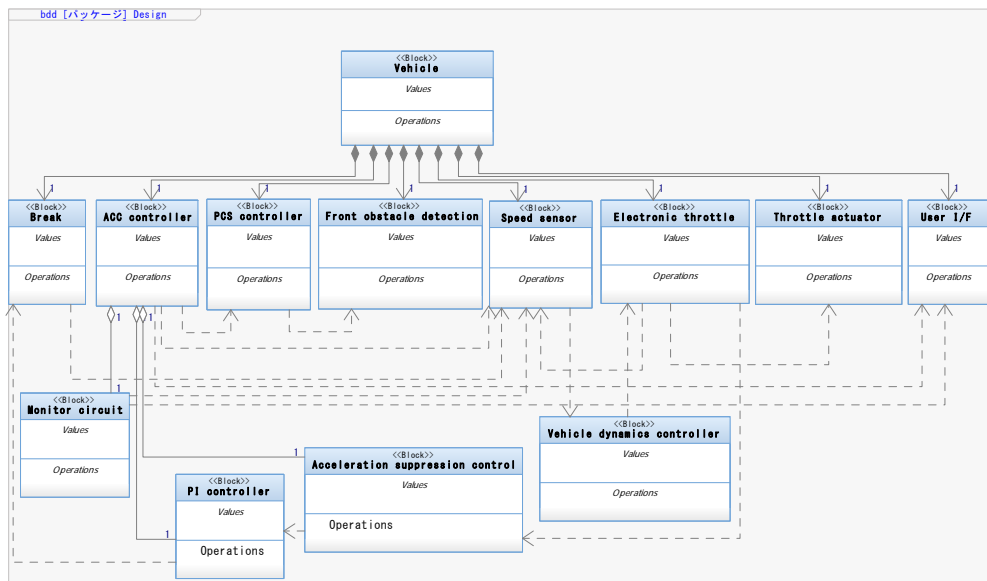

Figure 3 Block Definition Diagram

version: 1.0
revision: 1

3.4　Parametric Diagram

Parametric Diagram includes restrictions, preconditions of System, parameter, and mathematical formula.

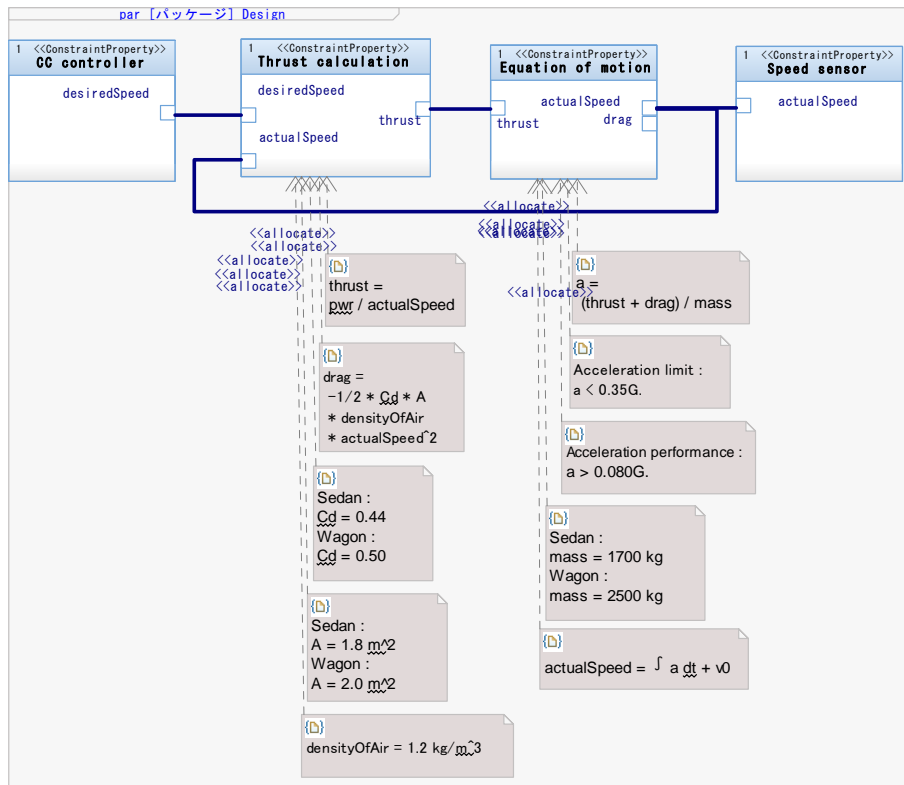It identifies external environment and characteristics of System (Figure 4).

Figure 4 Parametric Diagram

3.5　State Machine Diagram

State Machine Diagram includes behavior of System.

It identifies state transition of blocks (Figure 5).

version: 1.0
revision: 1

stm [Block] AccController [statechart_0]

running

init

cycle1ms

evPowerOFF

tm(1)/
if (this->isWorking) {
    double diffVelocity = (this->targetVelocity - this->velocity) / 3.6; // [m/s]
    this->sumDVelocity += diffVelocity;
    this->power += Kp * diffVelocity + Ki * this->sumDVelocity;
    if(this->power > this->maxPower) this->power = this->maxPower;
    if(this->power < -this->maxPower) this->power = -this->maxPower;
}
else {
    this->power = 0.0;
}

evAccPowerRequest(this->power) to itsArbitrationController

Off

evAccCruiseBtn
evAccCruiseBtn

evAccOFF

On

Unset

evAccSetBtn[valid(this->velocity)]/
set(this->velocity);

evAccBreakPedal
evAccResumeBtn

Sleeping

evAccSetBtn/
set(this->velocity);

evAccAccelBtn/
accel();

evAccDecelBtn/
decel();

Working

evAccSetBtn/
set(this->velocity);

evAccAccelBtn/
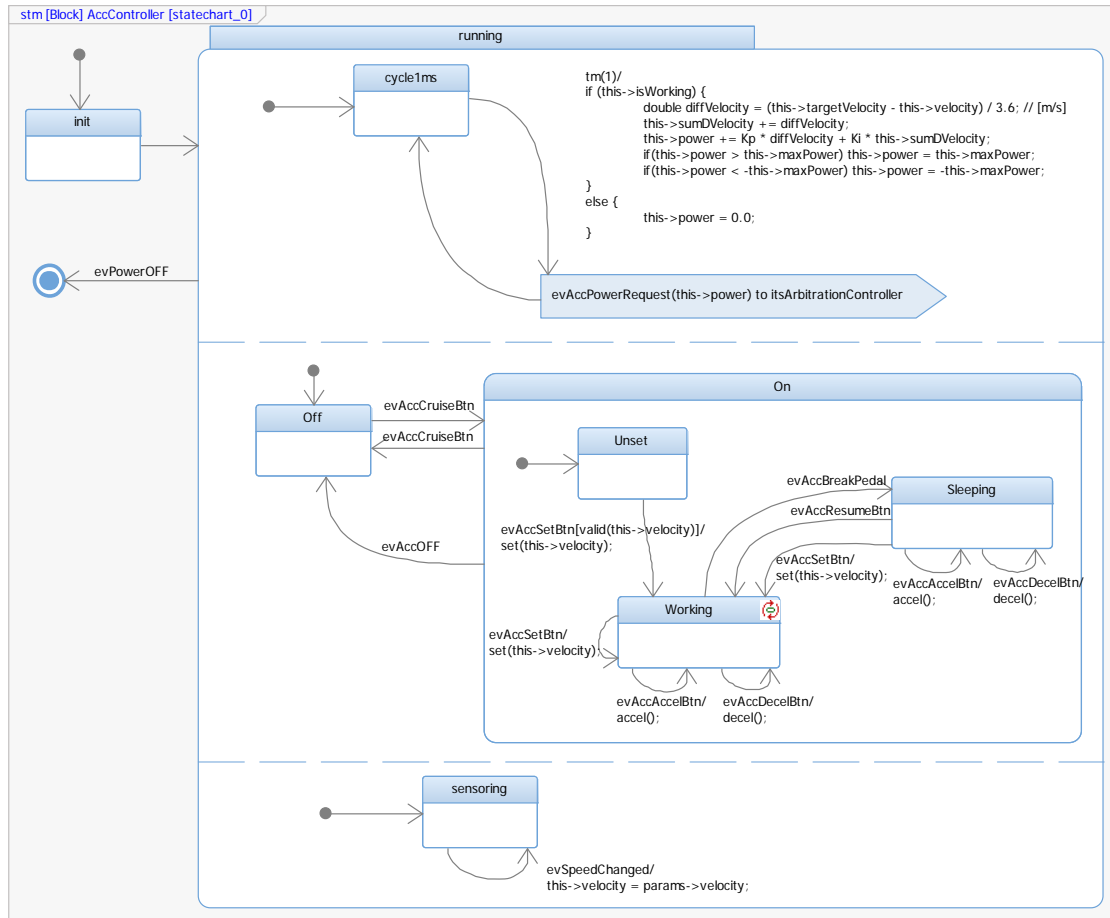accel();

evAccDecelBtn/
decel();

sensoring

evSpeedChanged/
this->velocity = params->velocity;

Figure 5 State Machine Diagram