DRIVING EVOLUTION

Automatic Code Generation For Autonomous Vehicles

Anthony Matarazzo
Bodo Seifert

AGENDA

01 DURA COMPANY OVERVIW

02 Requirements Modeling in SysML

03 Application Code Generation Using SCADE

04 SIL Testing

05 Conclusion

DURA AT A GLANCE

$1.4 B

SALES

* 2017 DURA + GAS

9,500

EMPLOYEES

36

LOCATIONS

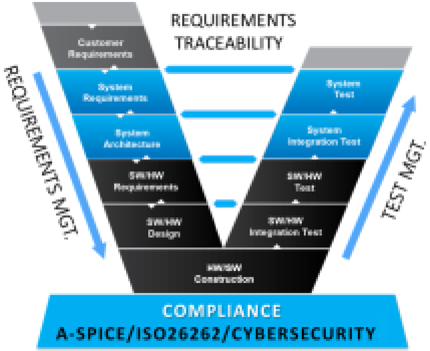
14

COUNTRIES

Agenda

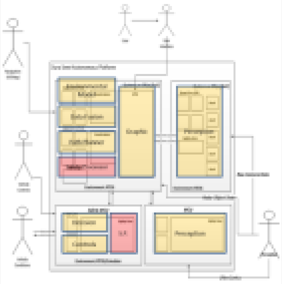
- Requirements Modeling in SysML
- Application Code Generation Using SCADE
- SIL Testing
- Conclusion

SYSTEMS ENGINEERING & USE CASE METHODOLOGY



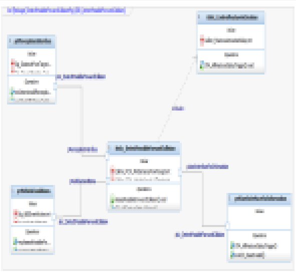
- Use cases are utilized for creation of **System Requirements** (Use Case and Supplemental)
- SYSML Diagrams are developed and traced to use case requirements via rational model manager, the diagrams allow for a **System Architecture** to be generated.
- Systems team delivers package to the application team for development.
- System Integration Testing is performed through MIL, SIL and HIL. All test cases are stored in Rational Quality Manager and traced back to each supplemental requirement.

Rhapsody: Use Case Diagram



System Architecture

Doors NG: Requirements



Rhapsody: IBD Diagram

Harmony Process

Systems Engineering Process

Model based systems engineering utilizes SysML following the IBM Harmony process to define the system behavior and interfaces in the IBM Jazz platform.

Example: Design A Hot Tub

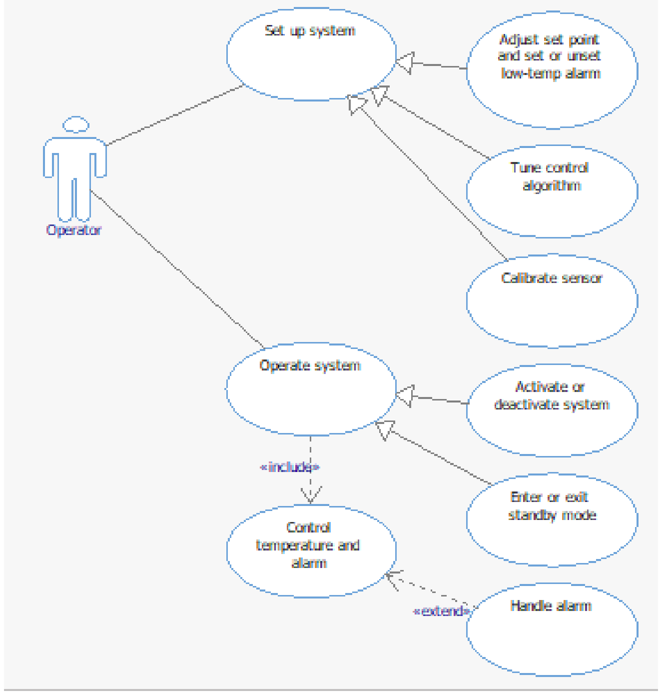
Feature:
I want a hot tub that I can fill with water and then turn on. Within one day I want the temperature of the water to reach a comfortable temperature. I want to be able to set the water temperature. I want to be able to turn the hot tub off when not in use.

Credits: IBM Spa Tutorial in Rhapsody

Very simple system

Task: Control the temperature of a given amount of water with a heat source.

Defining Requirements In IBM Rhapsody: Use Cases



User story:
I want a hot tub that I can fill with water and then turn on. Within one day I want the temperature of the water to reach a comfortable temperature. I want to be able to set the water temperature. I want to be able to turn the hot tub off when not in use.

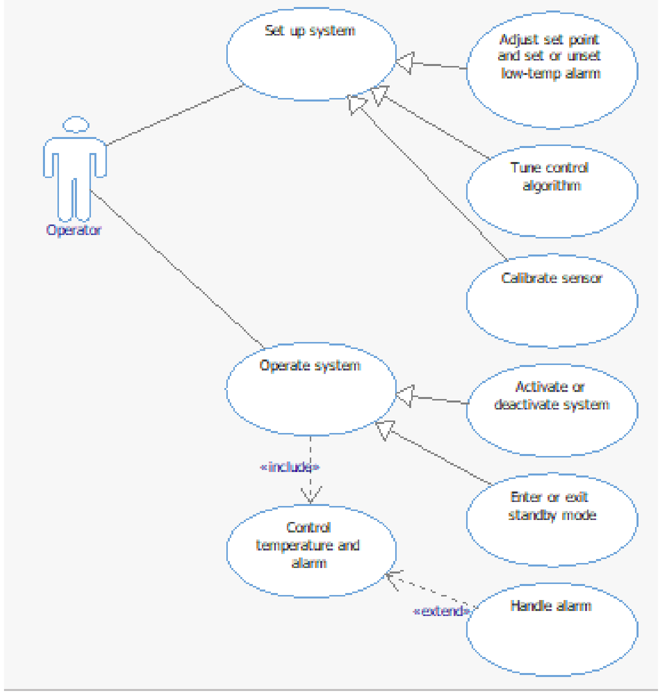
Use cases can be used to decompose

Use Cases

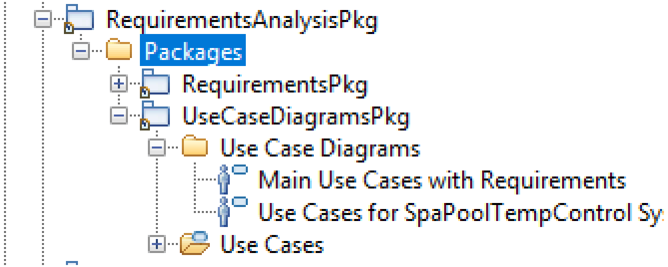
Contains use cases and requirements.
Also contains Rationales (e.g. results from other projects that determine a design constraint).

*) The screenshots are part of the IBM Spa Tutorial

Defining Requirements In IBM Rhapsody



IBM stores all individual elements in packages.
The actor package contains all actors.
The requirements analysis package includes the use cases and the requirements.
Use cases are defined graphically.

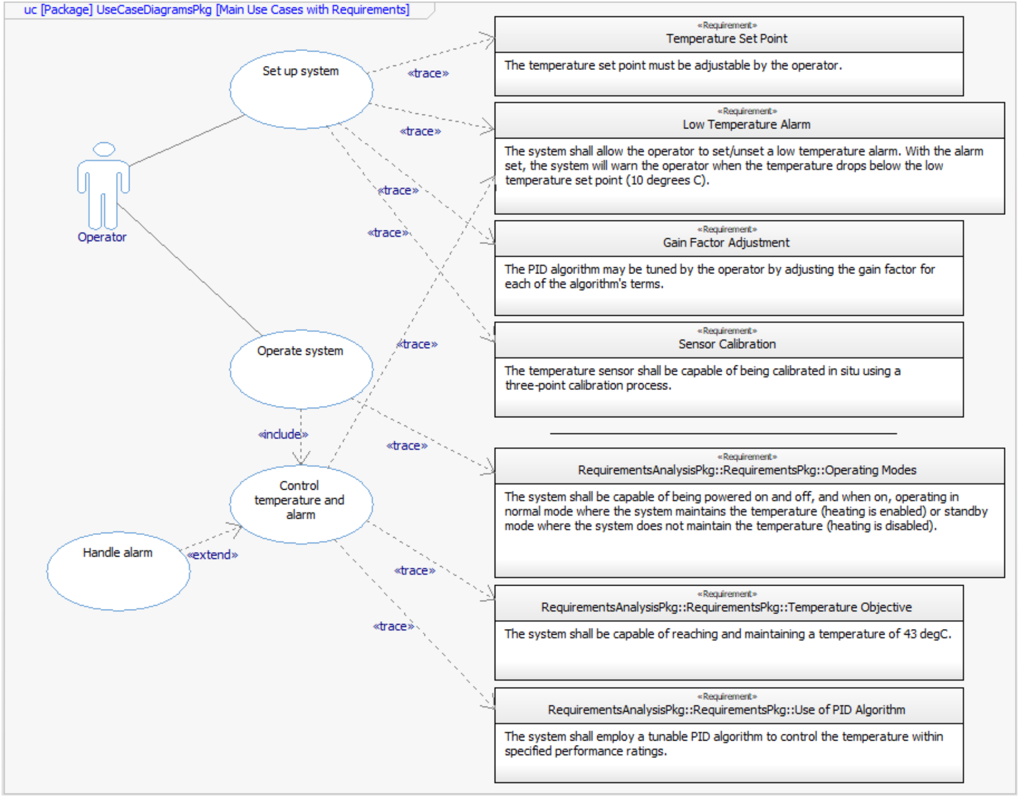


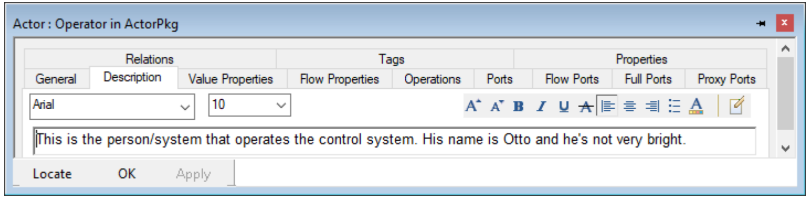
Requirements Analysis Package

Contains use cases and requirements.
Also contains Rationales (e.g. results from other projects that determine a design constraint).

*) The screenshots are part of the IBM Spa Tutorial

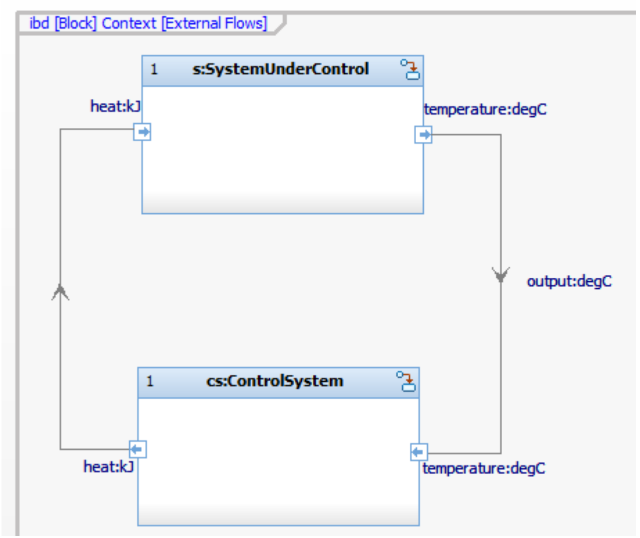
Linking Requirements To Use Cases





A use case analysis allows the team to focus on a limited and clearly defined portion of the feature.
This increases the chances to look at all facets of the feature and to capture a larger portion of requirements.

USE IBD To DESCRIBE THE SYSTEM

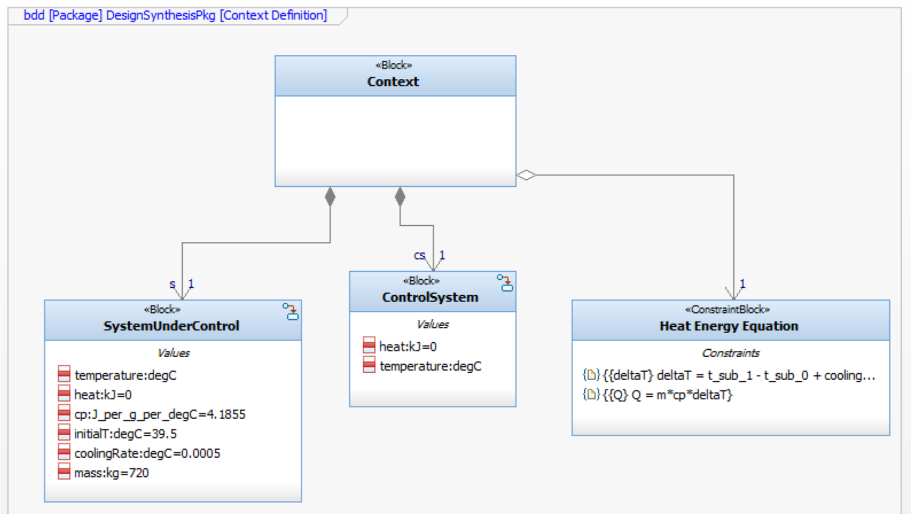


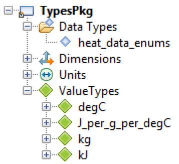
Rhapsody uses an Internal Block Diagram to describe the system.
In this case we chose a model that describes the System Under Control with one input (heat) and one output (temperature).
The other model describes the Control System with one output (heat) and one input (temperature).
Both models are black boxes in this representation but we can drill down into both models.
This is an example for a simple system architecture with only 2 blocks and 2 Interfaces.

System Architecture

After requirement analysis we are able to define a system architecture with blocks that define functionality and well defined interfaces.

Define System Context





The context is defined by the use cases, the system under control, the control system and physical constraints:

$$\Delta T = \frac{Q}{mc_p}$$

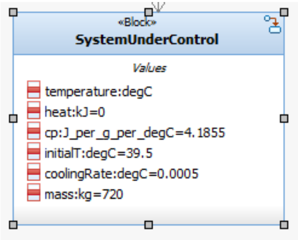
- where deltaT is the change in temperature of the water, degC
- Q is the heat input, kJ
- m is the mass of the water = 720 kg
- c_p is the heat capacity of the water = 4.1855 J/g/degC

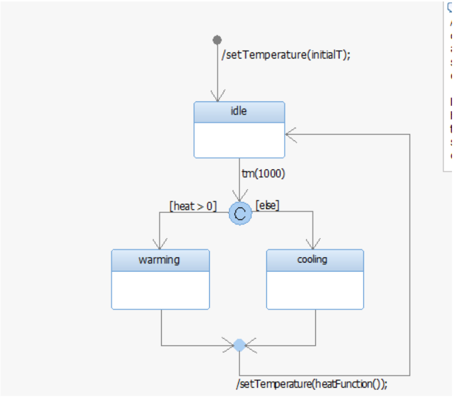
We need to transfer into the time discrete domain by using the heat output of the heater with 19 kJ/s for Q.

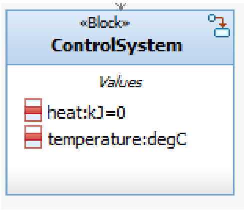
Define system context

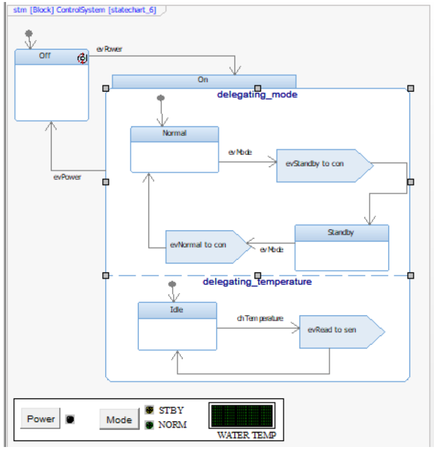
Define required value types.
Define main elements described in the model

State Charts for the Elements





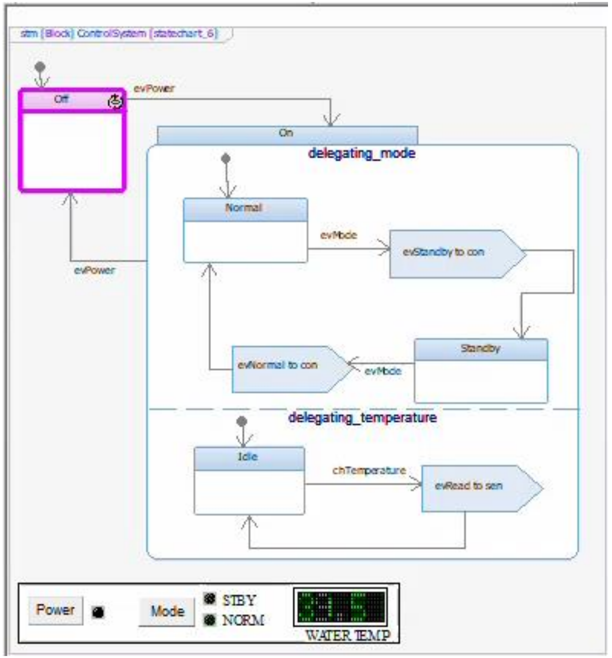




Define Behavior

State charts can be used to define the behavior of the blocks in the system architecture.

Simulating State Charts

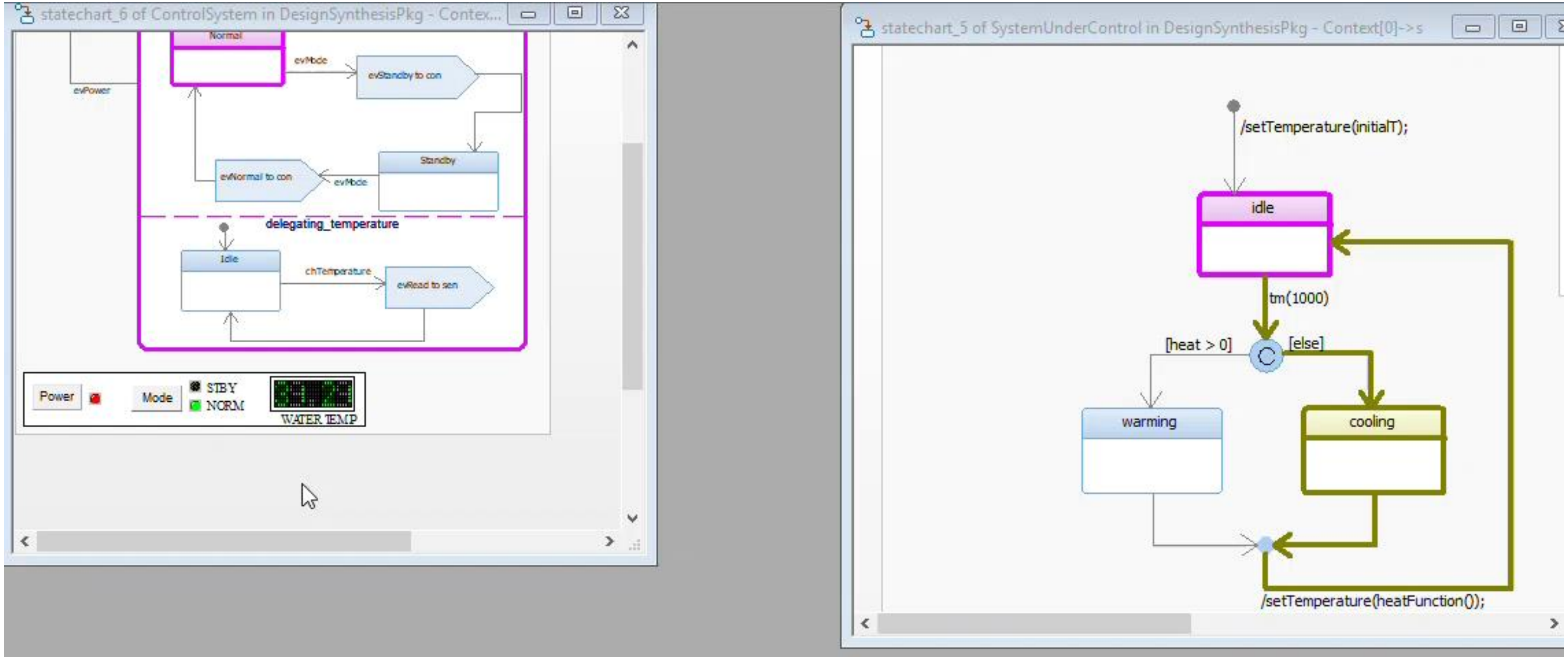


A panel allows to visualize variables and enables user interaction.
Here we have a simple control panel on the bottom of the screen that allows us to turn the system on and off and to change the system mode. The LED show the system state and the temperature display shows the simulated water temperature.
The state chart provides an overview of the logic we defined and we can either run it or set break points or step through it.

Simulate Behavior

Since we modeled the requirements we can now evaluate how the model behaves and get buy-in from stakeholders.

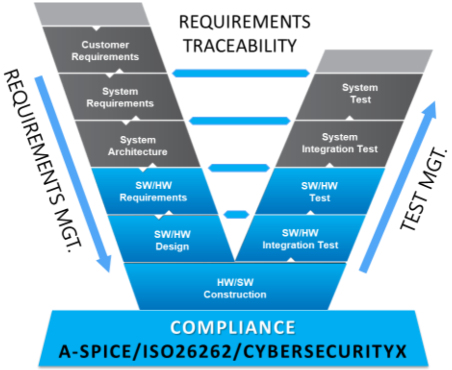
Simulating State Charts: SystemUnderControl



Complete System Overview

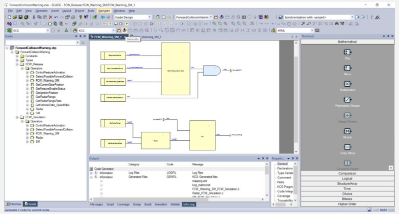
Rhapsody allows the developer to understand the state of the complete system by visualizing each sequence, state and activity diagram.

APPLICATION SOFTWARE ENGINEERING

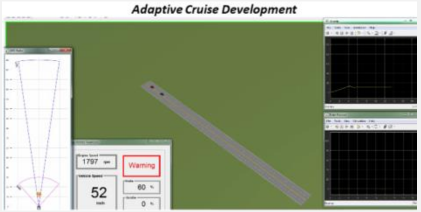


Process Overview

1. Application Team create software requirements and software design stored in **DOORS Next Generation** and traces to system requirements
2. Application Team uses the **Internal Block Diagram** generated by systems engineering to import to **ANSYS SCADE**
3. The **ANSYS SCADE** model is enhanced and traced to the system requirements in Doors NG
4. The **ANSYS SCADE** code generator produces the ISO 26262 compliant code in .c, .h, .m files, and s-functions that can be ported into a **Matlab/Simulink** s-function
5. The s-function is then run in SIL (e.g. **ANSYS VRX** or **TASS Prescan**) for "System Integration Tests"



ANSYS SCADE Model



Matlab/Simulink

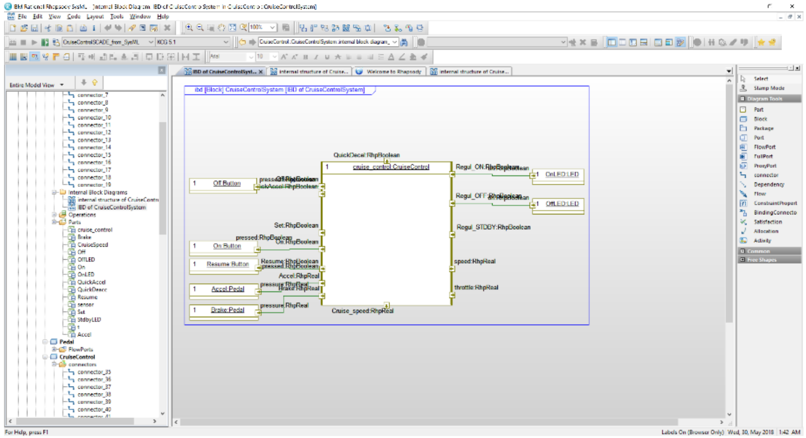
TASS Prescan or ANSYS VRX (Vehicle Simulation)

Apps Software Engineering Process

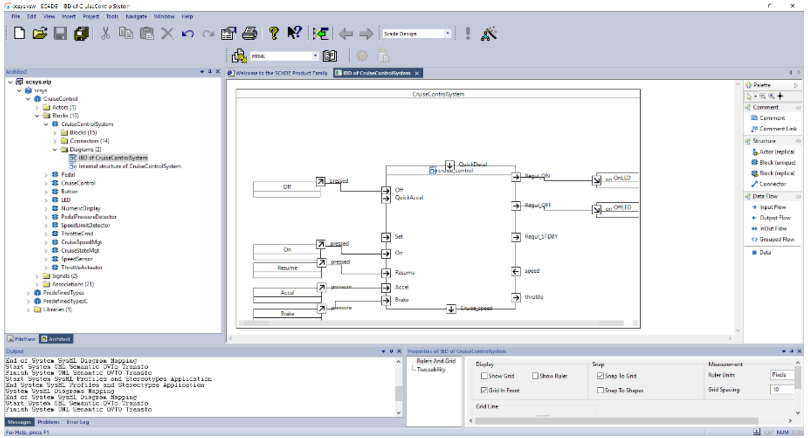
The Application SW defines the SW requirements and SW architecture traced to the system architecture. The generated code is ISO26262 certified. V&V is handled through MIL and SIL and HIL.

Import IBD to SCADE

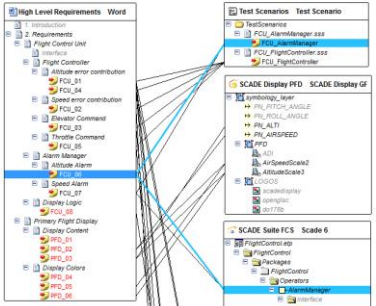
Rhapsody



SCADE

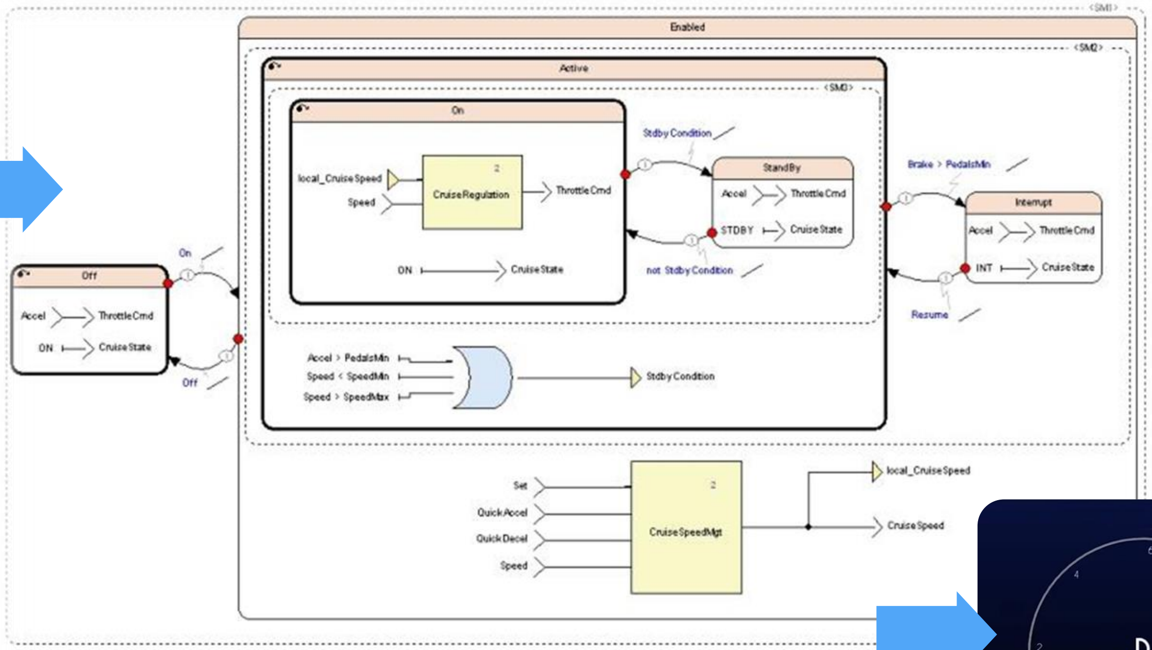


Strong Integration >>



1. Efficient
2. Traceable
3. Consistence

Develop the Application Model in SCADE

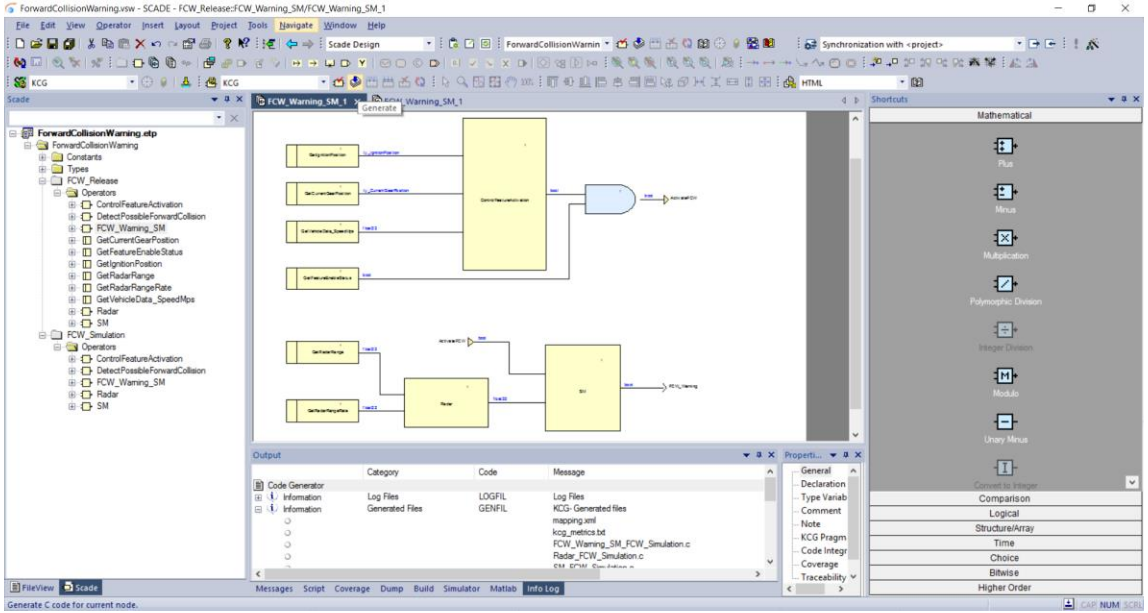


Top Level of the Cruise Control application

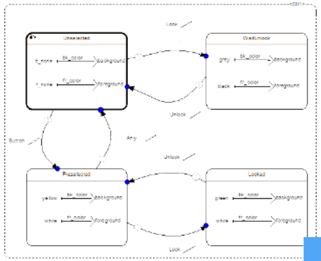
Verify functional safety and accuracy of the model.

Automated HMI Display design-level connection for rapid prototyping of behavioral logic and graphic components in embedded applications

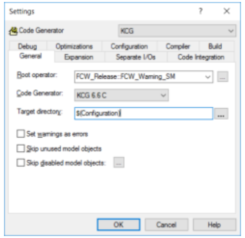
Compile into 26262 compliant code



Model



» One Click »



Source Code

```
[...]
void Button_ABC_N(inC_Button_ABC_N *inC,
    outC_Button_ABC_N *outC)
{
  /* ABC_N::Button::SM1::SSM_SM1_dispatch_sel */
  SSM_Button_SM1_ST SSM_SM1_dispatch_sel;

  if (outC->init)
    {
      outC->init = kcg_false;
      SSM_SM1_dispatch_sel = SSM_SM1_Unselected__ABC_N;
    }
  else
    {
      SSM_SM1_dispatch_sel = outC->M_pre_;
    }
  switch (SSM_SM1_dispatch_sel) {
    case SSM_SM1_Locked__ABC_N :
      outC->foreground = white_ABC_N;
      outC->background = green_ABC_N;
      if (inC->Unlock)
        {
          outC->M_pre_ = SSM_SM1_Preselected__ABC_N;
        }
      else
        {
          outC->M_pre_ = SSM_SM1_Locked__ABC_N;
        }
      break;
    case SSM_SM1_WaitUnlock__ABC_N :
      outC->foreground = black_ABC_N;
      outC->background = grey_ABC_N;
      if (inC->Unlock)
        {
          outC->M_pre_ = SSM_SM1_Unselected__ABC_N;
        }
      else
        {
          outC->M_pre_ = SSM_SM1_WaitUnlock__ABC_N;
        }
      break;
[...]
```

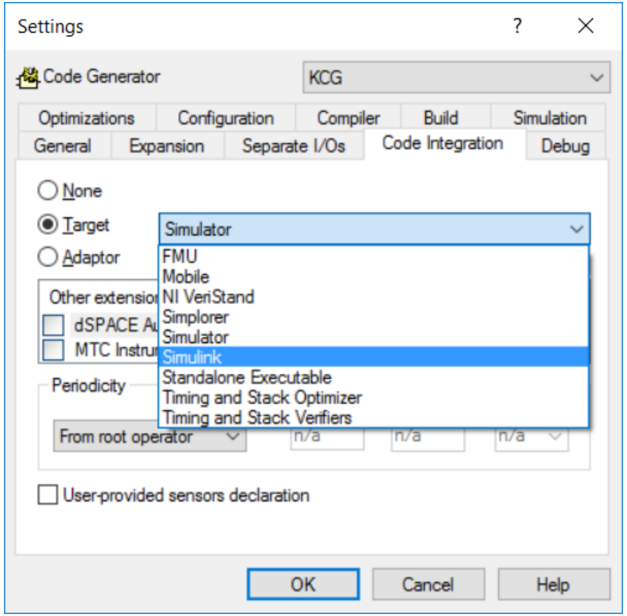
All application code is generated using an ISO 26262 complaint and qualified code generator by TUV (no additional certs needed)

Port Code to different targets/development tools

ANSYS

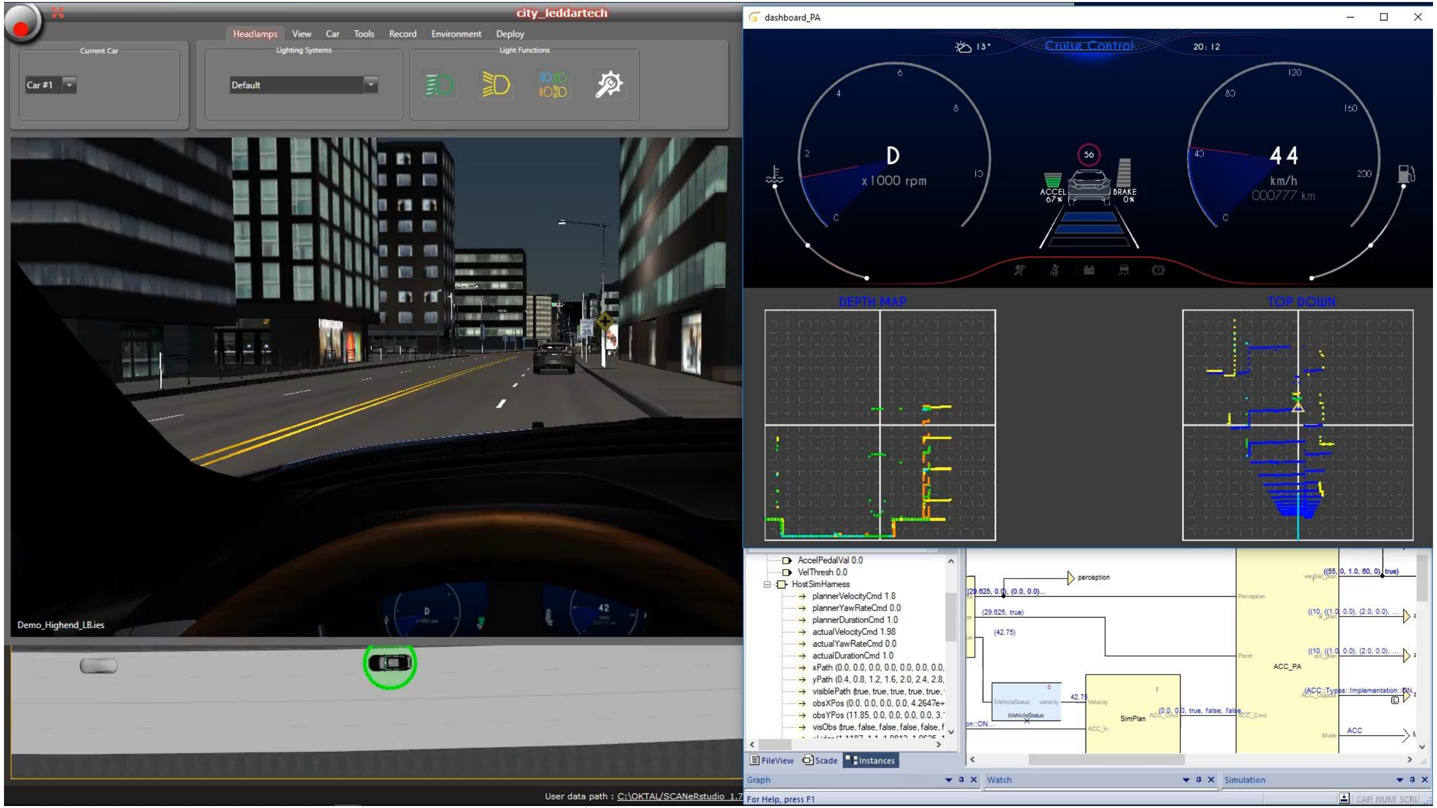
```
[...]
void Button_ABC_N(inC_Button_ABC_N *inC,
    outC_Button_ABC_N *outC)
{
  /* ABC_N::Button::SM1::SSM_SM1_dispatch_sel */
  SSM_Button_SM1_ST SSM_SM1_dispatch_sel;

  if (outC->init)
   {
    outC->init = kcg_false;
    SSM_SM1_dispatch_sel = SSM_SM1_Unselected__ABC_N;
   }
  else
   {
    SSM_SM1_dispatch_sel = outC->M_pre_;
   }
  switch (SSM_SM1_dispatch_sel) {
   case SSM_SM1_Locked__ABC_N :
    outC->foreground = white_ABC_N;
    outC->background = green_ABC_N;
    if (inC->Unlock)
     {
      outC->M_pre_ = SSM_SM1_Preselected__ABC_N;
     }
    else
     {
      outC->M_pre_ = SSM_SM1_Locked__ABC_N;
     }
    break;
   case SSM_SM1_WaitUnlock__ABC_N :
    outC->foreground = black_ABC_N;
    outC->background = grey_ABC_N;
    if (inC->Unlock)
     {
      outC->M_pre_ = SSM_SM1_Unselected__ABC_N;
     }
    else
     {
      outC->M_pre_ = SSM_SM1_WaitUnlock__ABC_N;
     }
    break;
[...]
```

MATLAB



All generated code is portable (ANSI C, compiler, platform and OS independent, MISRA compliant)

Test Code with SIL (ex: ANSYS VRX)



Summary

- Use cases aid requirements development
- Requirements are defined as executable SysML models
- The Internal Block Diagram can be ported to other modeling tools (SCADE)
- SCADE allows code definition through models
- Models are evaluated for FuSa
- Code is automatically MISRA compliant
- The SCADE S-Function can be integrated in MatlabSimulink
- A number of tools can be used for SIL (e.g. TASS Prescan)

Address The Challenge Of Unexpected Behavior In Modern Vehicles

THANK YOU

Bodo Seifert

Director

ADAS / HMI

DURA