Model-Based Design and Automatic Code Generation for Safety-Critical Software Development

Michael Beine

dSPACE GmbH

MAE Conference · November 2009 · Reading, UK
Model-Based Development – Established Tool Chain

- MATLAB®, Simulink®, Stateflow®
  - Established modeling environment
  - Simulink, Stateflow for control design

- TargetLink
  - Worldwide leading Autocode Generation tool
  - Direct implementation of your Simulink / Stateflow control algorithm
Recent Automotive Examples

- Audi quattro with Active Sport Differential

- Dynamic Performance Control in new BMW X6
TargetLink has been used successfully for many years to develop safety-critical software.

„TargetLink effortlessly fulfills the rigorous requirements for model-based development issued by European and American aviation authorities.“

„Using TargetLink, we have successfully carried out several software developments according to DO-178B that were certified for safety level A.“

Andreas Alaoui, Nord-Micro
• TargetLink for all algorithms (flight control, autopilot, flight management, calculation of flight data, navigation, signal consolidation in the triplex redundant system)

• 45% of the source code of the flight control computer was generated automatically using TargetLink
TÜV SÜD Automotive GmbH, German certification authority, evaluated TargetLink, with the following result:

- TargetLink code generator is fit for purpose to develop safety-related software according to:
  - IEC 61508
  - ISO DIS 26262
  - derivative standards such as EN 50128¹.

TÜV granted the following certificate

¹ EN 50128, standard for safety-related railway software, for example, is considered as a sector-specific standard derived from IEC 61508.
TargetLink Reference Workflow for the development of safety-critical systems

- provides guidance on how to fulfill functional safety requirements with model-based development methods and tools

- is based on best practices and experiences from many different production projects
Topics covered

- Requirements Traceability
- Software Architecture Considerations
- Modeling and Coding Guidelines
- Software Unit and Integration Testing

Mapping to IEC 61508 and ISO 26262
Model-Based Software Development

- Design and Implementation
  - From textual requirements via executable specifications and code generation to object code

- Design and Code Verification
  - Verification und validation of model and code
MISRA AC TL

- Special focus on functional safety
- Guidance for using TargetLink to avoid pitfalls on both, model level and code level
- Available at MISRA web store see [www.misra.org.uk](http://www.misra.org.uk)
Modeling Guidelines

- Select guidelines
  - From available standard guidelines documents
  - Determine project-specific guidelines
- Document which guidelines have been selected
- Ensure and document that guidelines are followed
Tool Support for Automated Guideline Checking

- Model Examiner (MXAM) from Model Engineering Solutions
  - Checks for MISRA AC TL, dSPACE TargetLink and MAAB guidelines
  - Easy integration of project-specific guidelines and model checks
  - Reports of all detected guideline violations
  - Further tools available, e.g. StyleChecker from AFT
Coding Guidelines

- MISRA C:2004 guidelines
- MISRA C Compliance Document for TargetLink
- Commercial of the shelf MISRA C compliance checker tools available
  - Legacy or handwritten code that is part of the model is also checked
- Violation in generated code have to be compared to known and accepted violations described in the MISRA C compliance document
- Reports for Off-PC reviews

**Overview**

**Application:**

- **fuelsys**

**TargetLink subsystems:**

- **fuelsystem**
  - Multirate-RTOS Data
- **fuelratecontroller**
  - Intake-AirflowEstimation() 
  - FuelRateCalculation() 
  - SensorCorrection() 
  - SensorCounter_data() 
  - FuelingMode_data() 
  - Cn26_Running_seq()
  - Tab2DS1712T4169_a()
  - TabldcS1771_a()
  - Tab2Dlntr171_a()

**Stateflow Machines:**

- **fuelsys_s6**
  - control logic

**Step function: FuelRateCalculation()**

- **Fuel Rate Calculation**
  - Description: fuel rate calculation
  - Source file: fuelratecontroller.c
  - Subsystems: fuelsys_s6/fuelratecontroller/Subsystem/fuelratecontroller/Fuel Calculation
  - Sample time: 0.01 sec
  - Function comment:
    - Function to calculate the fuel rate from the estimated airflow, depending on the fueling mode.
Model and Code Reviews

- Code with hyperlinks to the model and data dictionary

---

### TargetLink code view for model fuelsys_integration

**List of C Files**
- AirflowCalculation.c
- AirflowCalculation_Globals.c
- FuelCalculation.c
- FuelCalculation_Globals.c
- GlobalVars.c
- SensorCorrection.c
- SensorCorrection_Globals.c
- fuelratecontroller.c

**List of H Files**
- AirflowCalculation.h
- AirflowCalculation_Globals.h
- FuelCalculation.h
- FuelCalculation_Globals.h
- GlobalVars.h
- SensorCorrection.h
- SensorCorrection_Globals.h
- fuelratecontroller.h
- tlBasetypes.h
- tlDefines_FC.h
- tlDefines_a.h
- tlDefines_b.h
- tlDefines_sc.h
- tlTypes.h

---

```c
636 : /* State transition from fuelratecontroller/control
637 : econtroller/control logic.FuelingMode.FuelDisabled
638 : Ca26 Running ex();
639 : SIBFS control logic a.Ca23 X FuelDisabled = 1;
640 : fuel mode = DISABLED;
641 : SIBFS control logic a.Ca25 X Shutdown = 1;
642 : }
643 : else {
644 : if (speed > 40152 /* 628. */) {
645 : // Show DD Variable (Pool)
646 : Mode.Fuel
647 : // Show Block in Model (fuelratecontroller/.../speed)
648 : ed = 1;
649 : // Show Block in Model (fuelratecontroller/.../speed)
650 : SIBFS control logic a.Ca24 X Overspeed = 1;
651 : }
652 : else {
653 : switch (SIBFS control logic a.Ca26 X Running ns)
654 : case Ca27 X LowEmissions id: {
655 : /* Begin execution of state fuelratecontro
656 : g.LowEmissions */
657 : switch (SIBFS control logic a.Ca27 X LowEm
658 : case Ca28 X Normal id: {
659 : /* Begin execution of state fuelrate
```
Model-Based Software Development

- Implicit division into
  - Model level (Design)
  - Code level (Implementation)
Verification of the model

- Model is correct, meets requirements, contains no unintended functionality

Verification of Code

- Equivalence between model and generated code

Result

- Code is correct and meets requirements
Simulation as basis for software test

Model-in-the-Loop
- Controller model
  - Plant model or stimuli

Software-in-the-Loop
- C Code on Host-PC
  - C Code on target µC
  - EVM

Processor-in-the-Loop
- C Code
  - Compile
  - Link
  - Obj. code

Basics of Model-Based Verification and Validation
Verification and Validation Methods

- Requirements Based Test and Simulation
- Formal Verification

- Back-to-back Tests
  - Model (MIL) ⇔ Code (SIL/PIL)

Result
- Code is correct and meets requirements

Diagram:
- Requirements → Modeling → Controller model → Modeling → Implementation model → Code Generation → Source code → Compile Link → Obj. code
- Design Verification → Code Verification
Back-to-back Testing

- Core testing method
- Assure that the code correctly implements the verified model
Back-to-back Testing

- Core testing method
- Assure that the code correctly implements the verified model
Back-to-back Testing – Tool Support

- **Embedded Tester$^{\text{BASE}}$**

  - Automatic Test Execution & Regression
  - Automatic Test Evaluation
  - Automatic Regression Reporting
Structural Testing

- Model Coverage
  - Simulink V&V Toolbox

- Code Coverage
  - Statement / Decision
  - MC / DC

Modified Condition/Decision Coverage

- Test Vector Generation

Model-in-the-Loop
Controller model

Software-in-the-Loop
C Code on Host-PC

Processor-in-the-Loop
C Code on target μC
The complete Reference Workflow document is available from dSPACE

Please contact your dSPACE representative for a copy
Model-based development is widely applied in the industry

The TÜV certified Autocode generator TargetLink is used with success to develop safety-critical software in several industries, including Aerospace and Defense

TargetLink Reference Workflow based on best practice industry experiences provides guidance on the application of model-based development for safety-critical systems.
Thank you for listening!

Michael Beine

dSPACE GmbH · E-mail: mbeine@dspace.de

MAE Conference · November 2009 · Reading, UK