Model-Based Development of Safety-Critical Systems

Matthias Regensburger (regensbu@in.tum.de)
Christian Buckl (buckl@in.tum.de)
Overview

- Motivation
- Approach: Template Based Development
- Models used for Code Generation
- Future Work
Motivation

Robot control

Control of windmills

Medical applications
Model-based Development: Existing Tools

- For the application functionality there are good tools available:

  SCADE

  Matlab / Simulink

- But code at system level (fault-tolerance mechanisms, process management & scheduling, inter-process/inter-processor communication) is not generated.
Embedded Systems are heterogeneous

- Proprietary OS: C, Assembler
- Real-Time OS: C, C++
- Common OS: Java, C++

- The code generator must be extensible
- Appropriate meta-models must be designed
Approach: Template Based Development
Code Generation

- Developer
- Application Model
- Templates
- Code Generator
- Application code
- Generated Code
Advantages of This Approach

- Templates can be reused.
- Templates cover specific aspects of the system and can be implemented by specialists.
- Implementing the templates in an application independent way is relatively easy: similar to preprocessor commands.
- Code generator architecture is extensible:
  - new templates can be easily added
  - meta-model can be augmented
Development Process – Tool Chain

- Hardware Model
- Software Model
- Fault Model
- Fault-Tolerance Model
- Safety Model

Combined & Extended Model

Check Rules

User Code

Source Code

Templates

Modelling (by developer)

Combination & Extension of Submodels (automated)

Model Validation (automated)

Code Generation (semi-automated)
Development Architecture

- **Modeling:** Eclipse Modeling Framework (EMF)
  - Domain Specific Language

- **Code Generation:** openArchitectureWare
  - Meta code generator
  - Model validation and Model transformation

- **User Interface:**
  - Graphical Modeling Framework (GMF)
  - EMF dynamic instances
Application Model:

- We use the time-triggered paradigm as execution model
  - Task model is based on the simple task model (periodic tasks with no interaction points).
  - Race conditions are excluded by design \( \not\sim \top \) determinism (necessary for replica determinism).
  - There exist previously known points in time for the execution of fault-tolerance mechanism (prerequisite for a distributed realization).
- State and functionality of the tasks are separated by using global ports
  - Support of automatic voting and synchronization
Models used for Code Generation
Division into 5 Sub-Models

- Hardware Architecture Model
- Software Architecture Model
- Fault Model
- Fault Detection
- Fault Handling
- Fault Recovery

Fault-tolerance mechanisms

Safety Model
Hardware architecture model:

- **Electronic Control Unit (ECU):**
  - Programming language, operating system
  - Internal clock
  - Abstract network interface definition
  - Abstract I/O definition
    - Storage
    - Memory
    - A/D, D/A measurement cards

- **Network**
  - Abstract definition to support different types (CAN, Ethernet, TTP, ...)
  - References to ECU network interfaces
  - Infrastructure informations (Hubs, Switches, etc)
Software architecture model:

```
Technische Universität München
Department of Informatics, Unit VI: Robotics and Embedded Systems

Software architecture model:

```

```text
14.05.2007
15
```
Fault model:

- Based on FMEA

- Fault model is used:
  - for presumption of faults and the fault ranges
  - to check the applied fault-tolerance mechanisms
  - for correct realization of different mechanisms (example: the realization of inter-processor communication depends on the reliability of network medium)
  - for choosing test routines to detect faults
  - for generating certification documents
Fault-tolerance model:

- The developer can specify which mechanisms should be applied within the system

Areas

- Error detection
  - Software: absolute tests, relative tests, …
  - Hardware: memory tests, logic tests, …
  - Timing violations

- Error recovery:
  - Exclusion, Repair, Integration (TMR, hot-/cold-standby)
  - Rollback recovery
  - Reconfiguration

- Error processing
  - System restart, reboot, halt, ignore, readonly
  - User defined
Safety model:

- Safety Integrity Level Specification (SIL) of
  - Hardware components
  - Software parts
  - System Architecture

- Suggestions and prohibitions related to the SIL Level
  - Reconfiguration may not be used for SIL2-SIL4
  - Two channel architecture is partly necessary
Template Language - EXPAND

- Combination of model data and various templates for code generation

- Major statements:
  - FOR, FOREACH
  - IF, ELSE, ELSEIF
  - EXPAND
  - FILE

```c
/****************************
The following structs are defined to handle the local ports
****************************/

``FOREACH tasks AS t»
struct local_ports«t.name»
{
  FOREACH t.inPorts AS p»
    getReferenceDataType(p.type.toString()) in«p.name»;
  FOREACH t.outPorts AS p»
    getReferenceDataType(p.type.toString()) out«p.name»;
}«t.name»_ports;
ENDFOREACH

```
Validation Language - Check

- First order logic
- Syntactical and semantical model analyse
- Step-by-step model analyse (submodel, combined/extended model)

```java
context ECU ERROR "ecu: name not unique: "+name :
   this.eRootContainer.eAllContents.typeSelect(ECU).notExists(
      a|a!=this && a.name == this.name
   )
;
```
Code Generation

- **Status Quo:**
  - Templates for the operating system VxWorks 6.3 and the programming language C are available.
  - Meta-models are specified:
    - Hardware- and software sub-models are supported.
    - As fault-tolerance mechanisms, voting based on a TMR system is available.
  - Different lab applications are currently developed:
    - Inverted pendulum
    - Fault-tolerant elevator control (Hot-Standby)
    - Carrera racing car control
Lab application: A time-critical control application

Balance of a rod by switched solenoids.

- Sample times of 2.5 ms
- Only 100 lines of code (approx. 5%) had to be implemented manually.
Future Work
Ongoing Work

- Implementation of further templates:
  - Support of further fault-tolerance mechanisms
  - Templates for document generation
- Validation of used fault-tolerance mechanisms regarding the fault-model.
- Safety model integration
- Employment of the approach in industrial projects (funded by the German ministry of education and research).
- TÜV: Proof of concept (till end of 2007)