Model-based Development with Giotto@Simulink

Wolfgang Pree
University of Salzburg, Austria
www.SoftwareResearch.net

A joint project of
W. Pree, G. Stieglbauer and C. Kirsch

Contents

Giotto@Simulink tool chain
- S/G Translator:
  model transformation,
  Giotto code generation
- illustrated by the development of a throttle control system
Giotto-based development process

Application
Control Problem

Modeling Tools
Mathematical Model
(Simulink)

S/G Translator
RTW Embedded Coder

Program Generation
Implementation Tools
Embedded Software Model
Timing Program
(Giotto)
Functionality Program
(C)

Giotto Compiler
C Compiler

Compilation/Linking

Embedded System
Real-Time Code

S/G Translator generates a SL model with Giotto semantics for simulation

generated code


Case study: code generation from a Giotto@Simulink model of a throttle control system
S/G Translator

- model transformation for simulation
- model transformation for functionality code generation
- generation of Giotto program

S/G Translator tool

SL model

SL model with Giotto semantics

SL model with drivers for integration with E-machine

Giotto program

SL model with drivers for integration with E-machine

Giotto program
Step 1: S/G model for simulation

S/G Translator generates a SL model with Giotto semantics for simulation

required input for the S/G translator

Plant modeled using any SL blocks

controls

GiottoProgram 1..* GiottoTask * SL Block

S/G translator is fully compliant with the current SL syntax
Step 2a: S/G model for the generation of functionality code that seamlessly integrates with the E-machine

Step 1

Application

Control Problem

Modeling Tools

Mathematical Model (Simulink)

S/G Translator: generates a SL model with Giotto semantics for simulation

S/G Translator RTW Embedded Coder

Program Generation

Implementation Tools

Embedded Software Model

Timing Program (Giotto)

Functionality Program (C)

Giotto Compiler

C Compiler

Embedded System

Real-Time Code

S/G Translator tool

SL model

SL model with Giotto semantics

SL model with drivers for integration with E-machine
preparation for linking timing code and functionality code (I)

Functionality program (SL → C code)

C compiler

Functionality library

Functionality wrappers

Giotto compiler

E code

E machine

intercepts calls

preparation for linking timing code and functionality code (II)

Giotto program segment

GiottoTask1( ... ) output ( ... ) state ( ... ) {
    schedule GiottoTask1();
}

Functionality wrapper

void task_GiottoTask1() {
    GiottoTask1();
}

Functionality code

void _GiottoTask1_output_1=GiottoTask1_input1+GiottoTask_input_2;
preparation for linking timing code and functionality code (III)

transport and convert values between task ports:
- via global variables (Simulink/RTW)
- via the Giotto driver concept

Giotto drivers are called by the E-machine

Step 2b: generation of the Giotto program

Step 1 √

Application

Control Problem

Modeling Tools

Mathematical Model (Simulink)

S/G Translator: generates a SL model with Giotto semantics for simulation

Step 2a: √

SL model for generating glue code

Program Generation

Implementation Tools

Embedded Software Model

Timing Program (Giotto)

Functionality Program (C)

Giotto Compiler

C Compiler

Compilation/Linking

Step 2b: Giotto program

Embedded System

Real-Time Code

S/G Translator tool

Step 2c: generation of the functionality program with the RTW Embedded Coder

Step 1

Application
Control Problem

Modeling Tools
Mathematical Model (Simulink)

S/G Translator

RTW Embedded Coder

Step 2a:
SL model for generating glue code

Step 2b:
Giotto program

S/G Translator: generates a SL model with Giotto semantics for simulation

Step 2c:
C program

how long it took to ...

- upgrade the S/G Translator: 4 p. months
  - a redesign that streamlines the architecture and makes the tool fully compliant with SL syntax: 2.5 m
  - generation of SL model for glue code generation: 1m
  - reimplementation of the C# version in Java: 0.5 m

- implement the ETC case study: 0.7 p. months
Future plans

Next steps

short term:
- illustrate composition and time safety checks in the realm of the ETC case study
- integration of Giotto modes into Simulink

mid-term:
- S/G-based prototype implementations of more complex control system components
- concepts for control system product families
The end

Thank you for your attention!