

Model-based Development with Giotto@Simulink

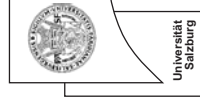
Wolfgang Pree
University of Salzburg, Austria
www.SoftwareResearch.net

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

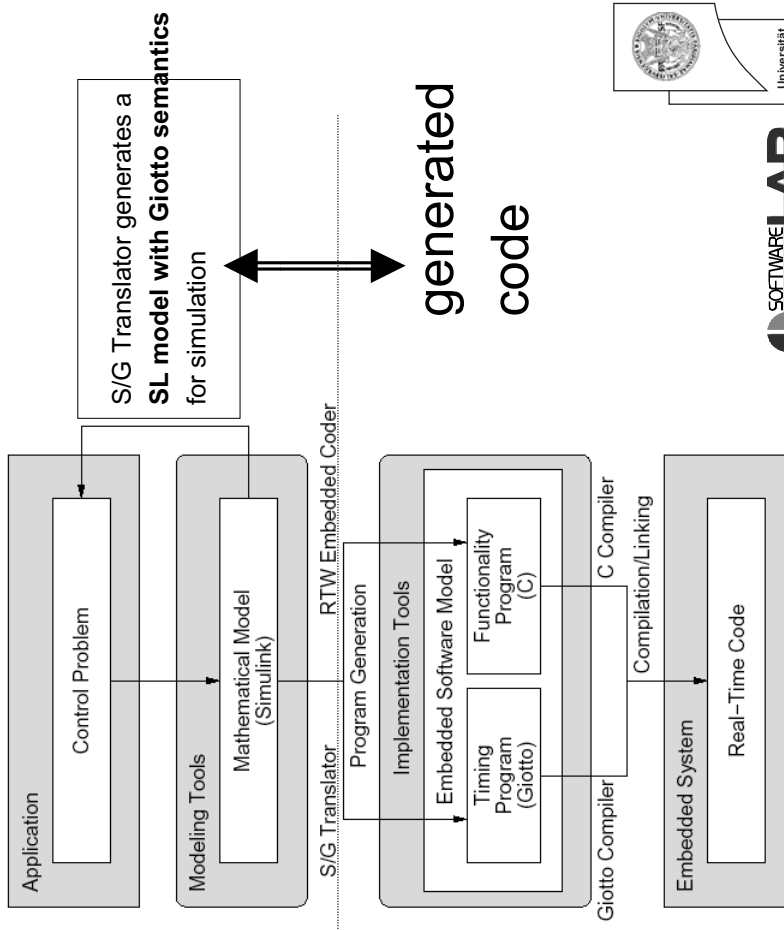


Contents

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



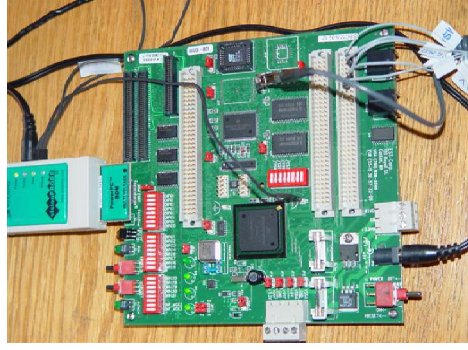
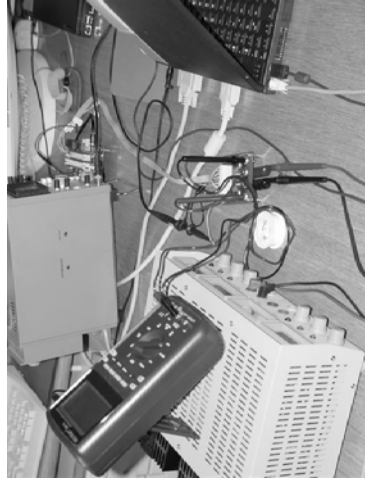
Giotto-based development process



3

(from the Giotto paper published in the IEEE Control Systems Magazine, Feb. 2003)

Case study: code generation from a Giotto@Simulink model of a throttle control system



4

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

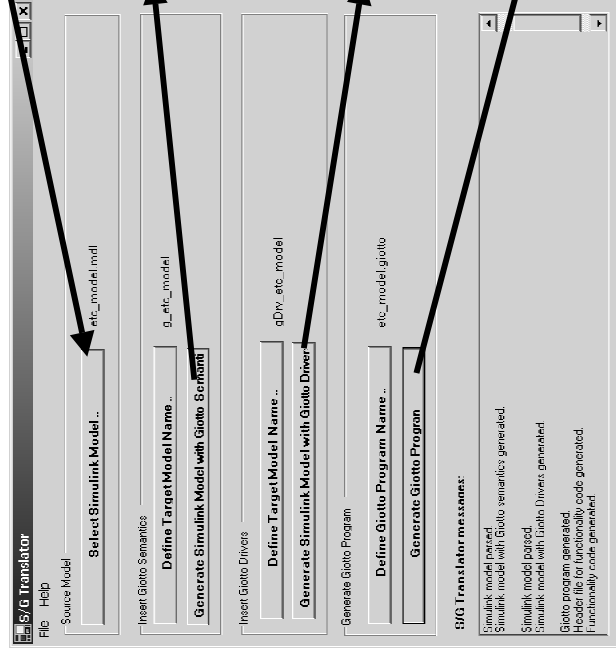
S/G Translator

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

5

© 2003, W. Pree, G. Sitegbauer, C. Kirsch

S/G Translator tool

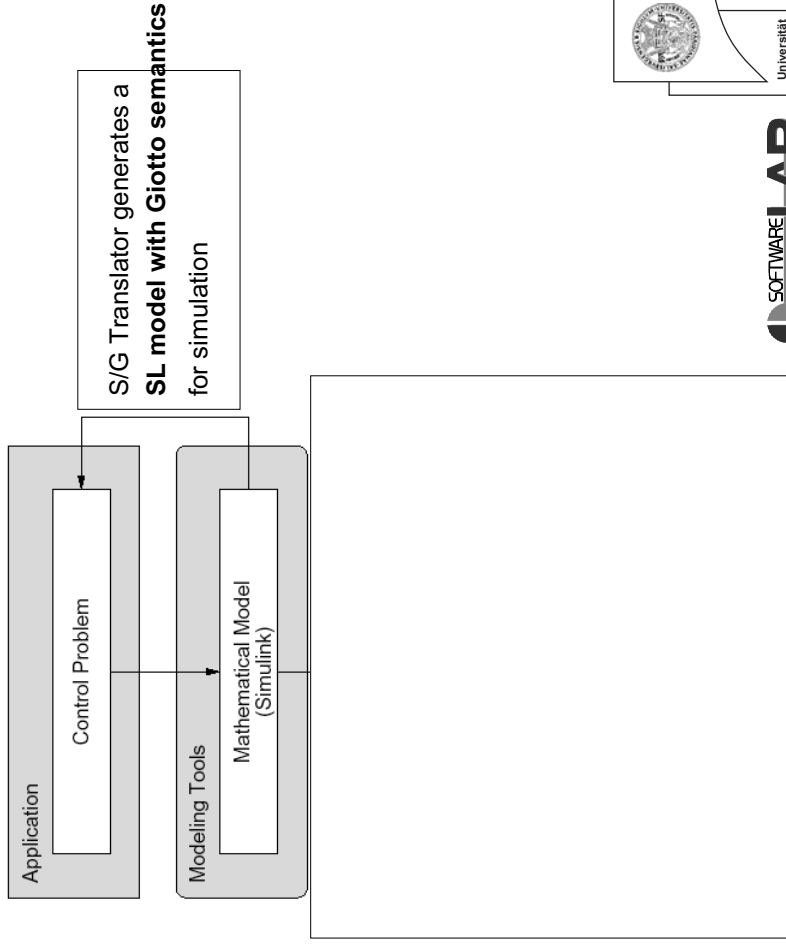


6

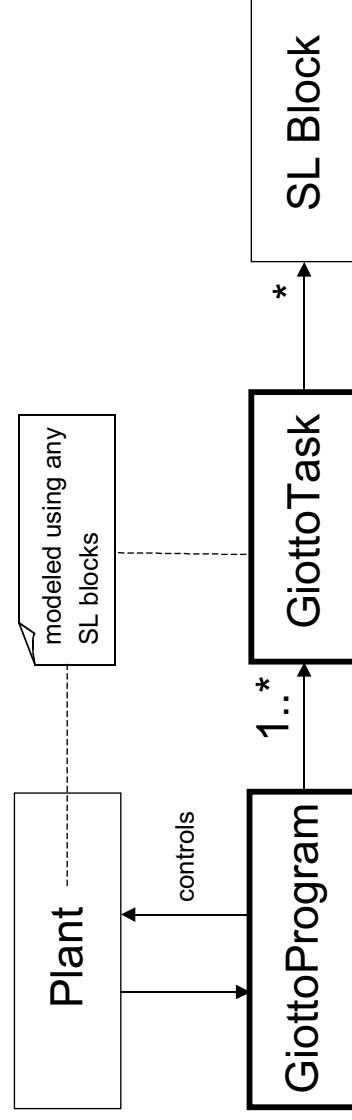
© 2003, W. Pree, G. Sitegbauer, C. Kirsch

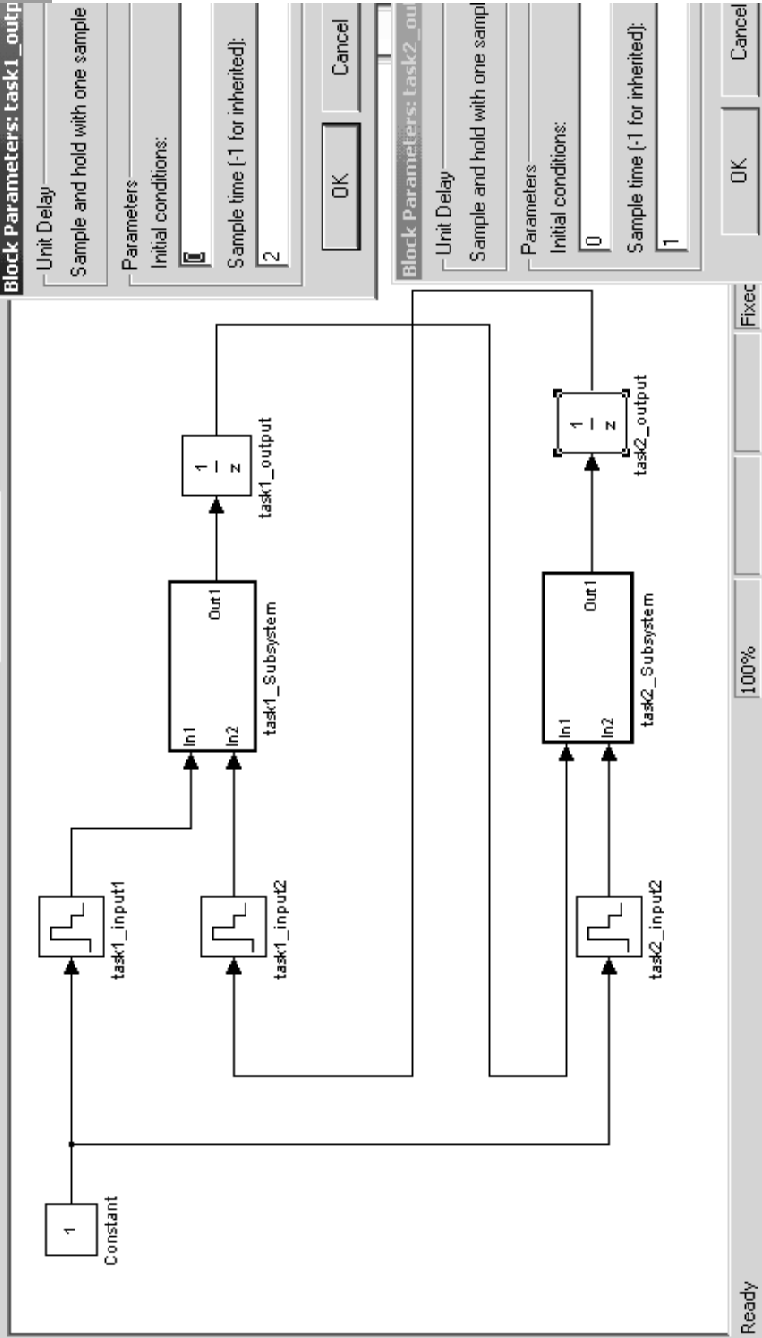
Step 1: S/G model for simulation

Step 1

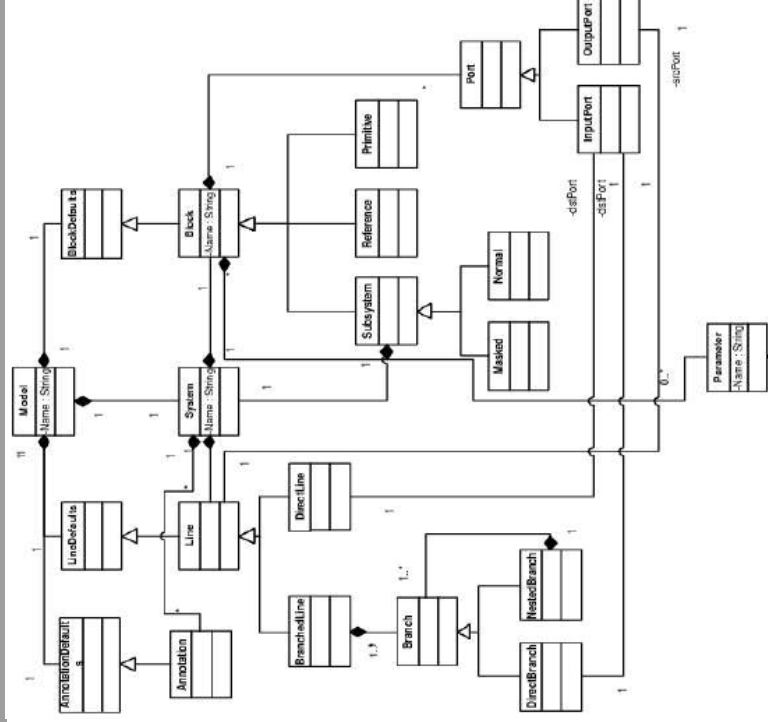


required input for the S/G translator



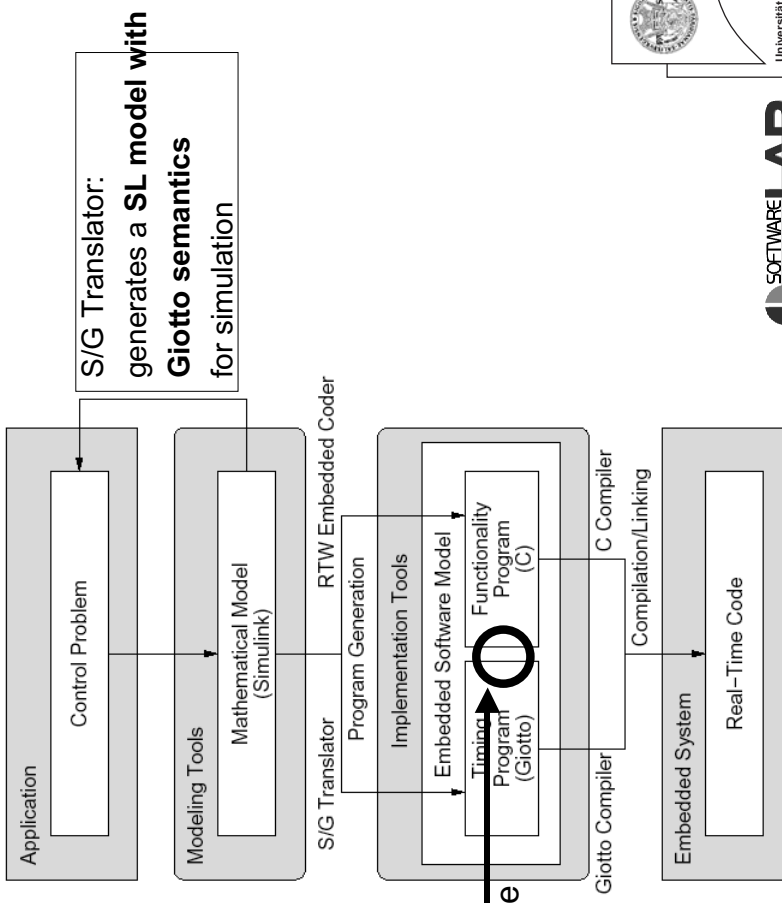


S/G translator is fully compliant with the current SL syntax



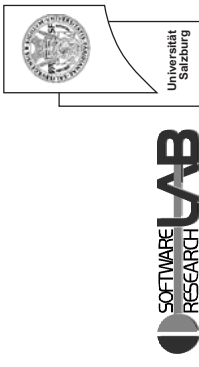
Step 2a: S/G model for the generation of functionality code that seamlessly integrates with the E-machine

Step 1 ✓

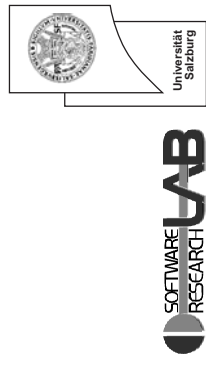
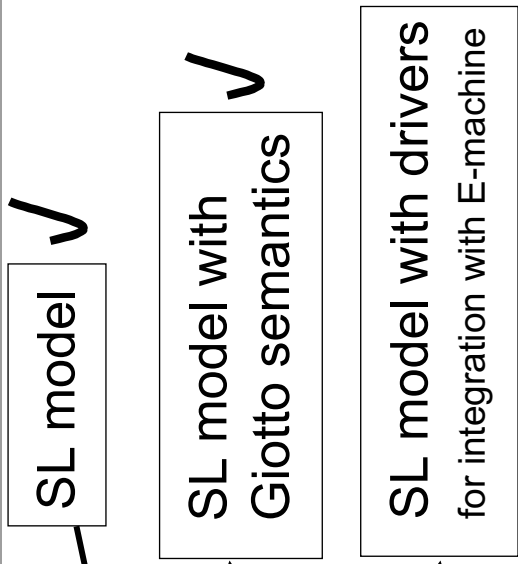
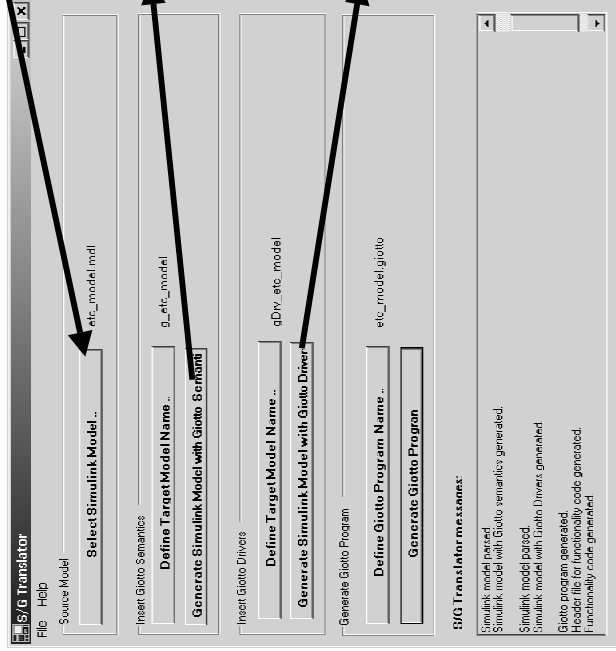


Step 2a:
SL model for
generating glue code

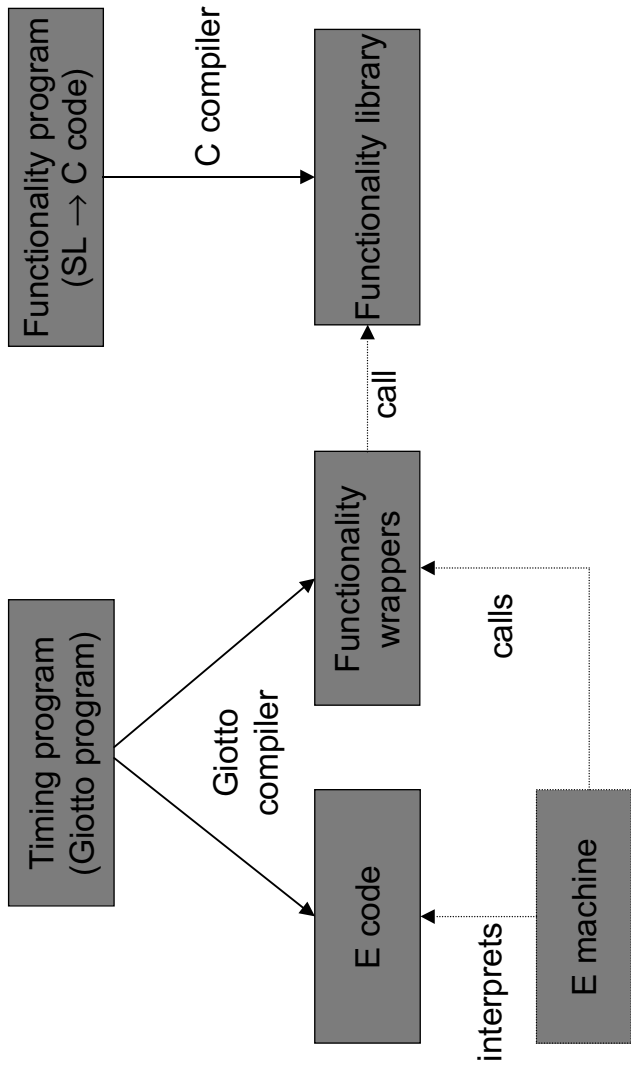
(from the Giotto paper published in the IEEE Control Systems Magazine, Feb. 2003)



S/G Translator tool



preparation for linking timing code and functionality code (I)



13

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

preparation for linking timing code and functionality code (II)

- Giotto program segment

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

- Functionality wrapper

```
void task_GiottoTask1 () {  
    GiottoTask1 ();  
}
```

- Functionality code

```
void GiottoTask1 (void) {  
    local_GiottoTask1_output_1=GiottoTask1_input1+GiottoTask_input_2;  
}
```

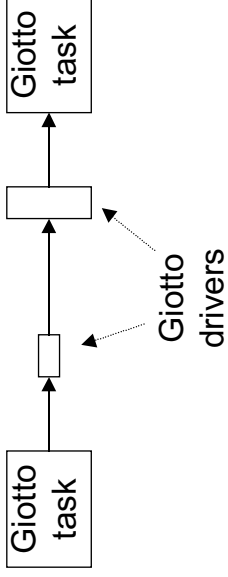
14

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

preparation for linking timing code and functionality code (III)

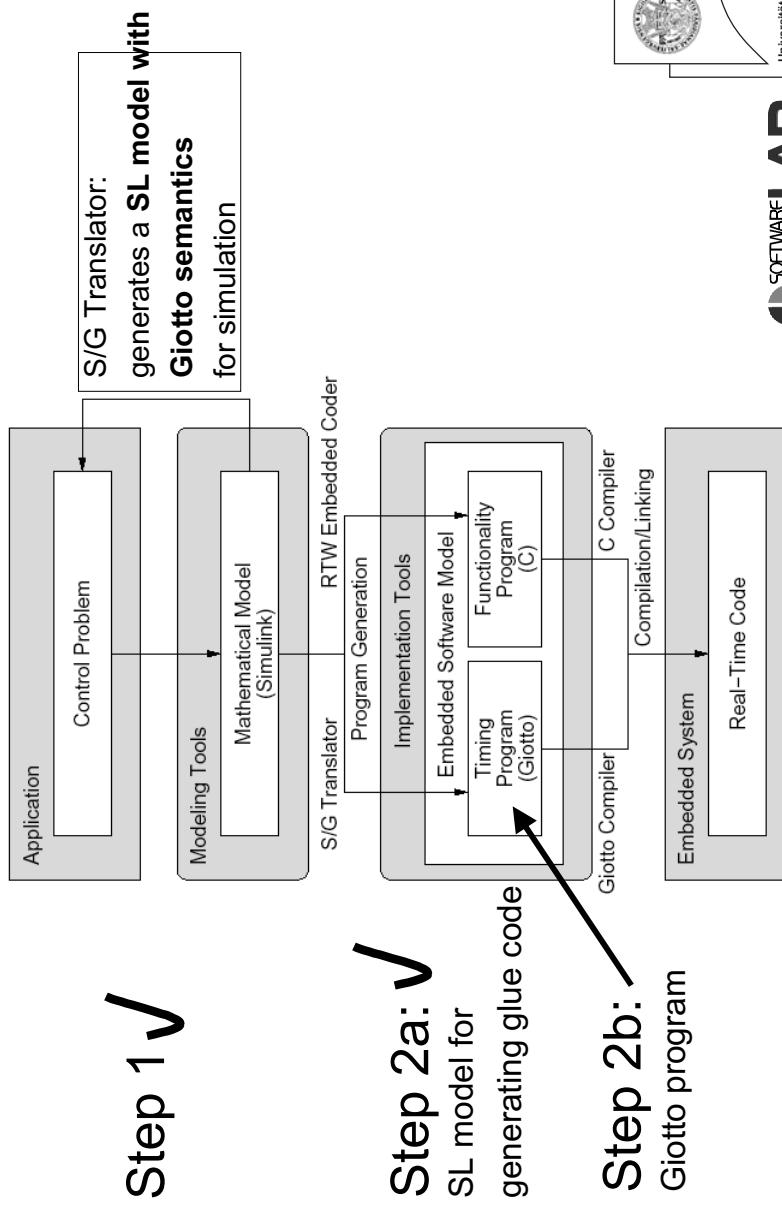
transport and convert values between task ports:

- 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



S/G Translator tool

The screenshot shows the S/G Translator tool interface. It has a menu bar (File, Help) and several sections:

- Source Model:** A text field containing "etc_model.mod".
- Define Target Model Name:** A text field containing "d_etc_model".
- Generate Stimulink Model with Giotto Semantics:** A button.
- Insert Giotto Drivers:** A section with a text field containing "dDrv_etc_model" and a button "Generate Stimulink Model with Giotto Drivers".
- Generate Giotto Program:** A section with a text field containing "etc_model.giotto" and a button "Generate Giotto Program".
- S/G Translator messages:** A scrollable area showing messages like "Simulink model parsed", "Simulink model with Giotto semantics generated", "Simulink model parsed", "Simulink model with Giotto Drivers generated", "Giotto program generated", and "Giotto program with Giotto semantics generated".

Four arrows point from the interface to boxes on the right:

- From the "Source Model" field to a box: **SL model** ✓
- From the "Generate Stimulink Model with Giotto Semantics" button to a box: **SL model with Giotto semantics** ✓
- From the "Generate Stimulink Model with Giotto Drivers" button to a box: **SL model with drivers for integration with E-machine** ✓
- From the "Generate Giotto Program" button to a box: **Giotto program**

Logos for SOFTWARE RESEARCH LAB and Universität Salzburg are present at the bottom right.

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

The flowchart illustrates the RTW Embedded Coder process:

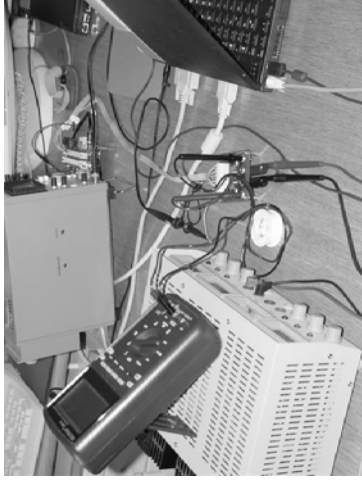
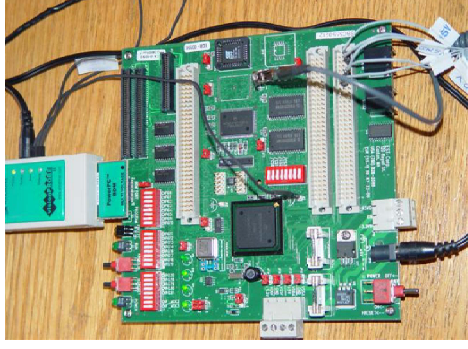
- Application:** Contains a **Control Problem**.
- Modeling Tools:** Takes the Control Problem and produces a **Mathematical Model (Simulink)**.
- S/G Translator:** Takes the Mathematical Model and produces an **RTW Embedded Coder**.
- Program Generation:** Takes the RTW Embedded Coder and produces an **Embedded Software Model**.
- Implementation Tools:** Takes the Embedded Software Model and produces a **Timing Program (Giotto)** and a **Functionality Program (C)**.
- Giotto Compiler:** Takes the Timing Program and Functionality Program and produces **Compilation/Linking**.
- Embedded System:** Contains the final **Real-Time Code**.

Annotations and steps:

- Step 1:** ✓ S/G Translator: generates a **SL model with Giotto semantics** for simulation. (Points to the S/G Translator box)
- Step 2a:** ✓ SL model for generating glue code. (Points to the Implementation Tools box)
- Step 2b:** ✓ Giotto program. (Points to the Giotto Compiler box)
- Step 2c:** C program. (Points to the Embedded System box)

Logos for SOFTWARE RESEARCH LAB and Universität Salzburg are present at the bottom right.

throttle control system @ work



19

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

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**
 - reimplementaion of the C# version in Java: **0.5 m**
- implement the ETC case study: **0.7 p. months**

20

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

Future plans

21

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

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

22

© 2003, W. Pree, G. Siteglbauer, C. Kirsch

The end

Thank you for your attention!