Towards a
Real-Time Systems Compiler

Fabian Scheler
Martin Mitzlaff
Wolfgang Schröder-Preikschat
Horst Schirmeier

Department of Computer Sciences IV
Distributed Systems and Operating Systems
Friedrich-Alexander University Erlangen-Nuremberg

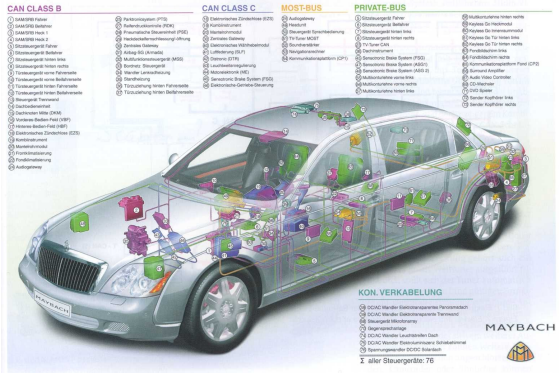
http://www4.cs.fau.de/~scheler
scheler@cs.fau.de

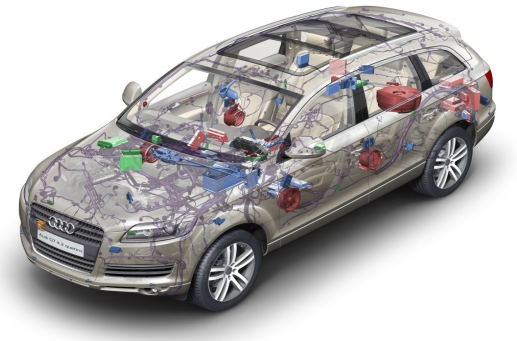
Recent Cars – Future Cars

- **event-triggered** communication
  - mainly CAN
- **federated** architecture
  - OSEK/VDX
- **fail stop** semantics
  - ABS, ESP

→

- **time-triggered** communication
  - FlexRay
- **integrated** architecture
  - AUTOSAR
- **fail operational** semantics
  - {Steer,Break}-by-Wire





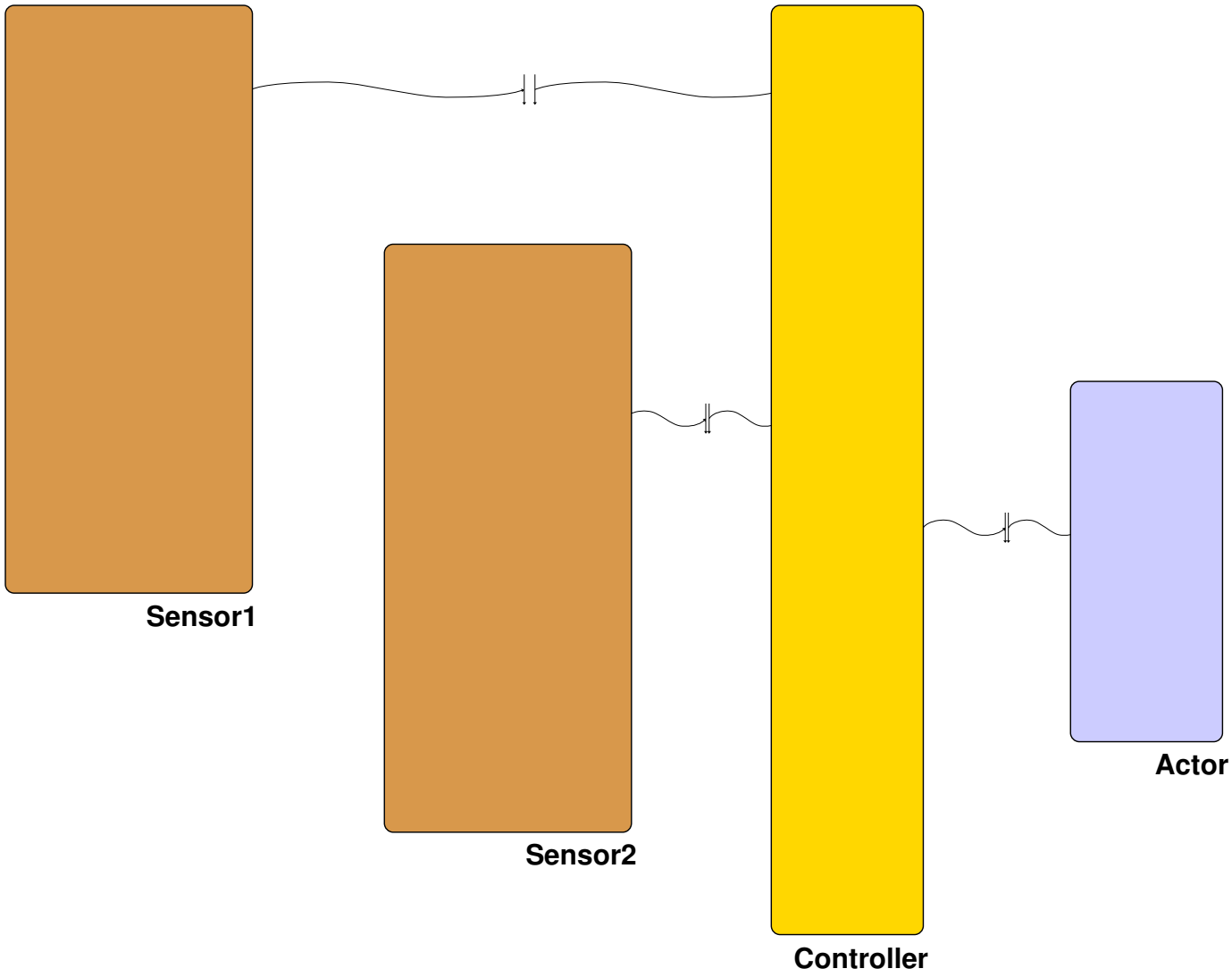
Consequences

- Migration
  - from an **event-triggered** to a **time-triggered** environment
- Legacy Applications
  - if possible: reuse (e.g. via virtual CAN-networks on top of FlexRay)
    - non-safety-critical subsystems, e.g. comfort applications
  - otherwise: migration/porting
    - safety-critical subsystems, e.g. ABS, ESP
- ➔ Porting
  - labour intensive
  - source of errors

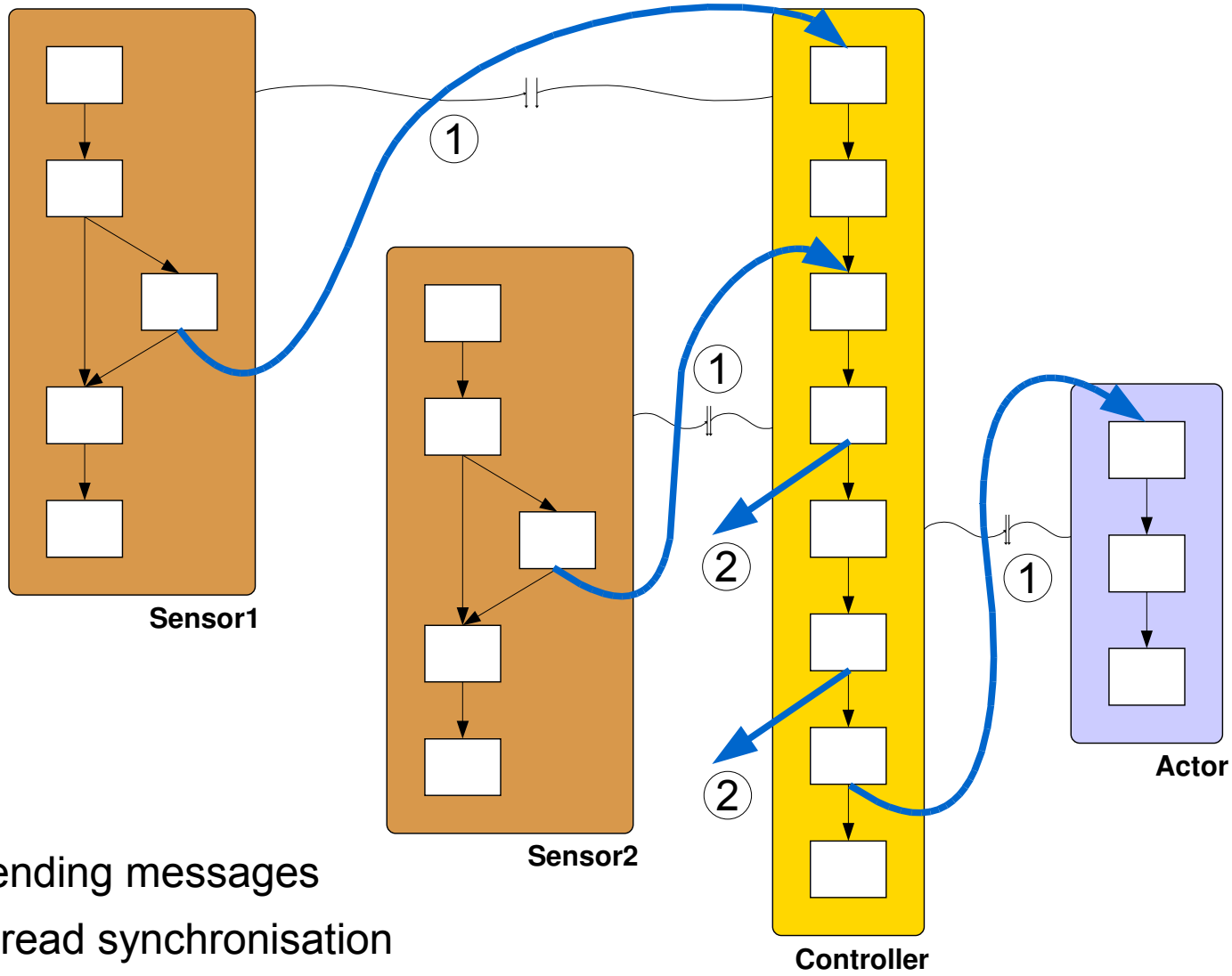
Overview

- **Why Migration is a Problem**
- Aiding Migration
- Atomic Basic Blocks
- The Real Time Systems Compiler
- Current Status & Future Work

A simple scenario



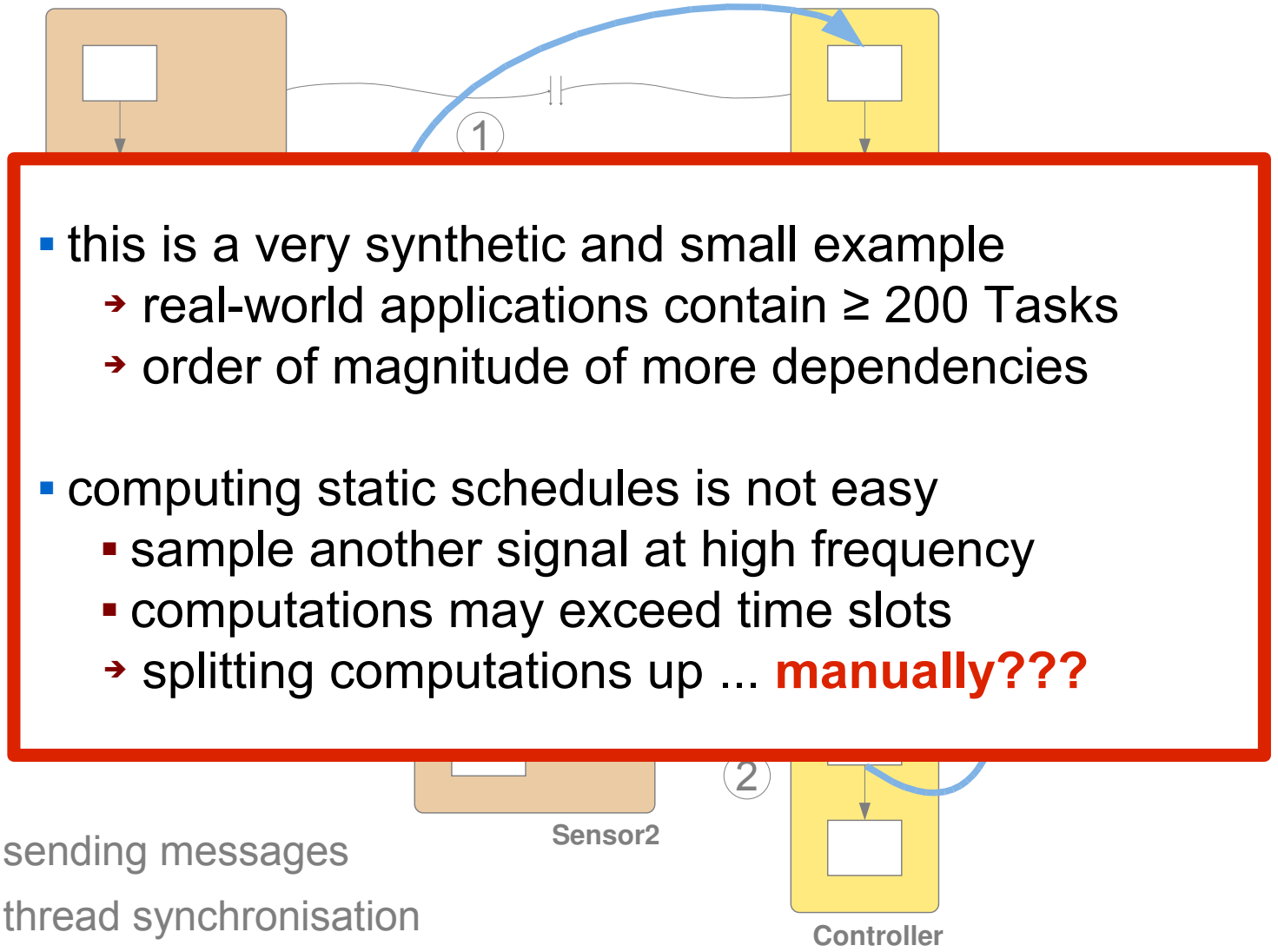
Explicitly Modelled Dependencies



① sending messages

② thread synchronisation

Explicitly Modelled Dependencies



- this is a very synthetic and small example
  - ➔ real-world applications contain ≥ 200 Tasks
  - ➔ order of magnitude of more dependencies
- computing static schedules is not easy
  - sample another signal at high frequency
  - computations may exceed time slots
  - ➔ splitting computations up ... **manually???**

1 sending messages

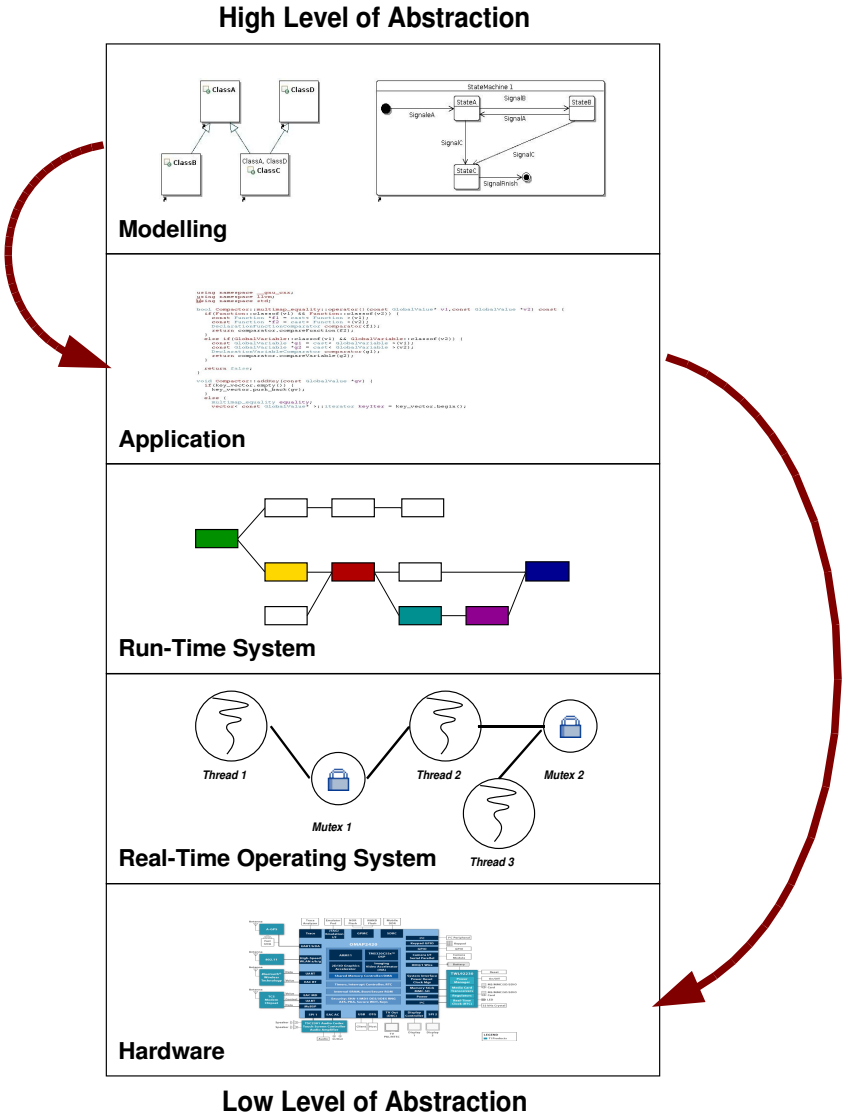
2 thread synchronisation

Overview

- Why Migration is a Problem
- **Aiding Migration**
- Atomic Basic Blocks
- The Real Time Systems Compiler
- Current Status & Future Work

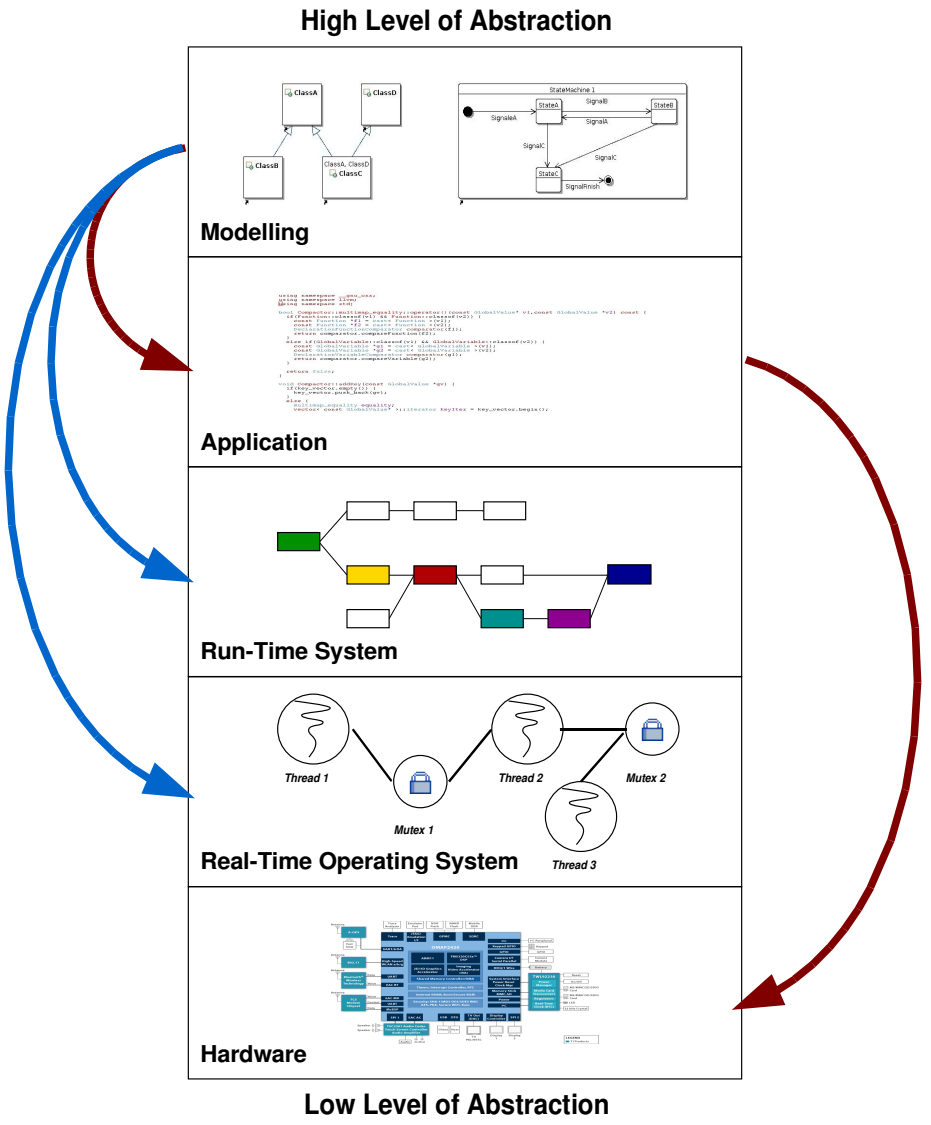
Lowering Transformations

- automated & generic



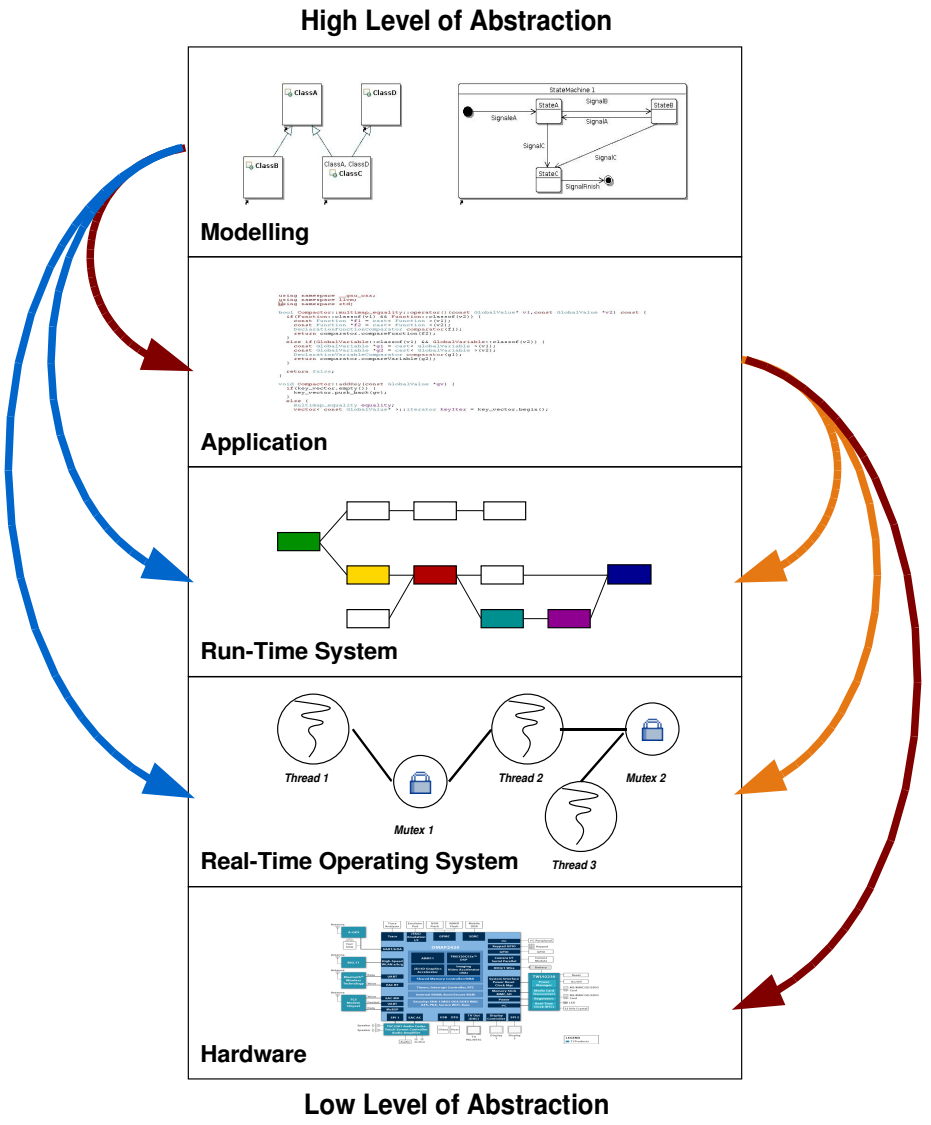
Lowering Transformations

- automated & generic
- automated



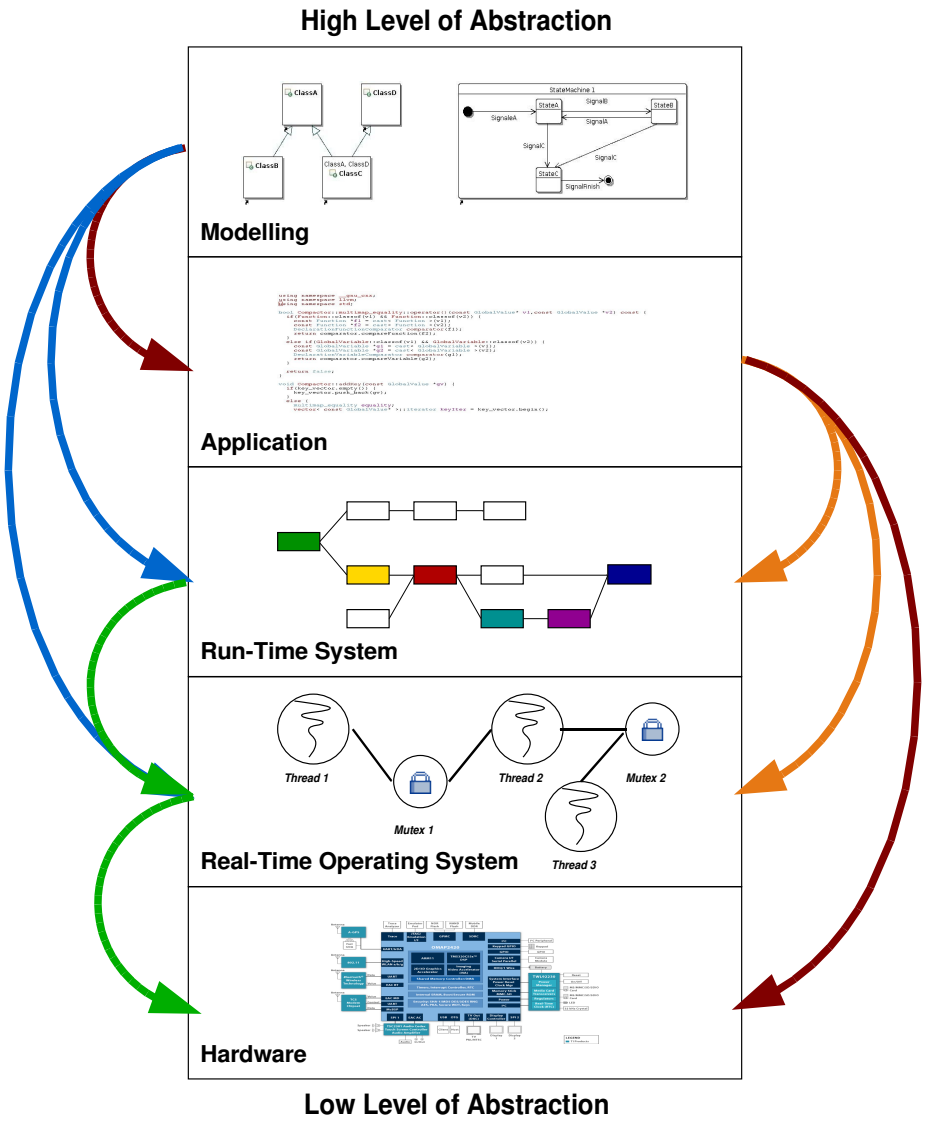
Lowering Transformations

- automated & generic
- automated
- manual



Lowering Transformations

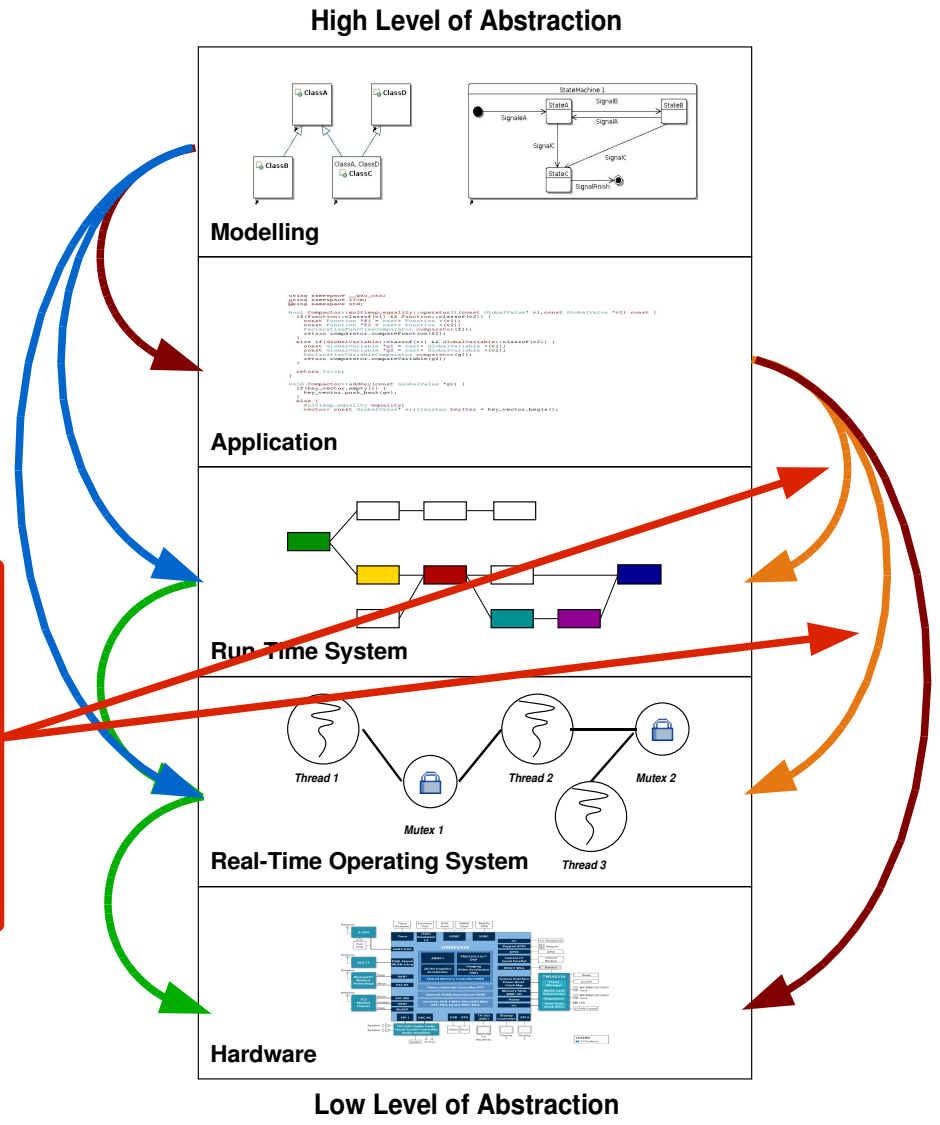
- automated & generic
- automated
- manual
- inherently manual



Lowering Transformations

- automated & generic
- automated
- manual
- inherently manual

➔ automate the **manual** steps (not the **inherently manual** ones)



Overview

- Why Migration is a Problem
- Aiding Migration
- **Atomic Basic Blocks**
- The Real Time Systems Compiler
- Current Status & Future Work

Approach

- decouple Application and OS/Run-Time System
  - intermediate representation
  - independent of the employed control flow abstraction

➔ combination of different
  - front ends and
  - back ends

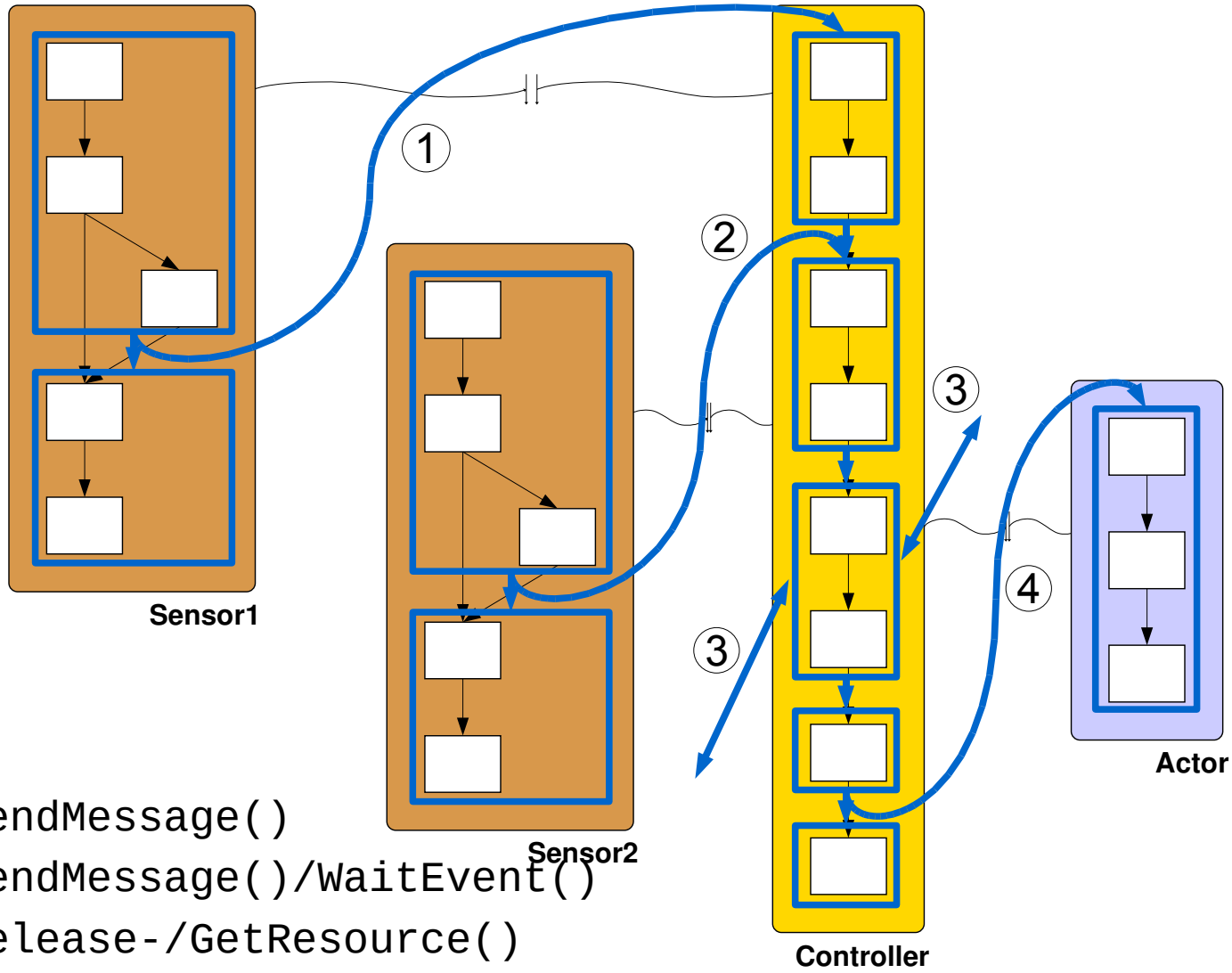
- similar to compiler construction

➔ intermediate representation
  - control flow graphs (CFG)
  - basic blocks

Atomic Basic Blocks

- specification of dependencies across different CFGs
  - data dependencies
  - explicitly modelled logical and temporal dependencies
  - mutual exclusion
- ABB-graph is superimposed on forest of CFGs
- ABBs aggregate several basic blocks
- ABB boundaries
  - forking/joining other CFGs
  - being joined by another CFG
  - critical sections

Atomic Basic Blocks



① SendMessage()
② SendMessage()/WaitEvent()
③ Release-/GetResource()
④ SendMessage()

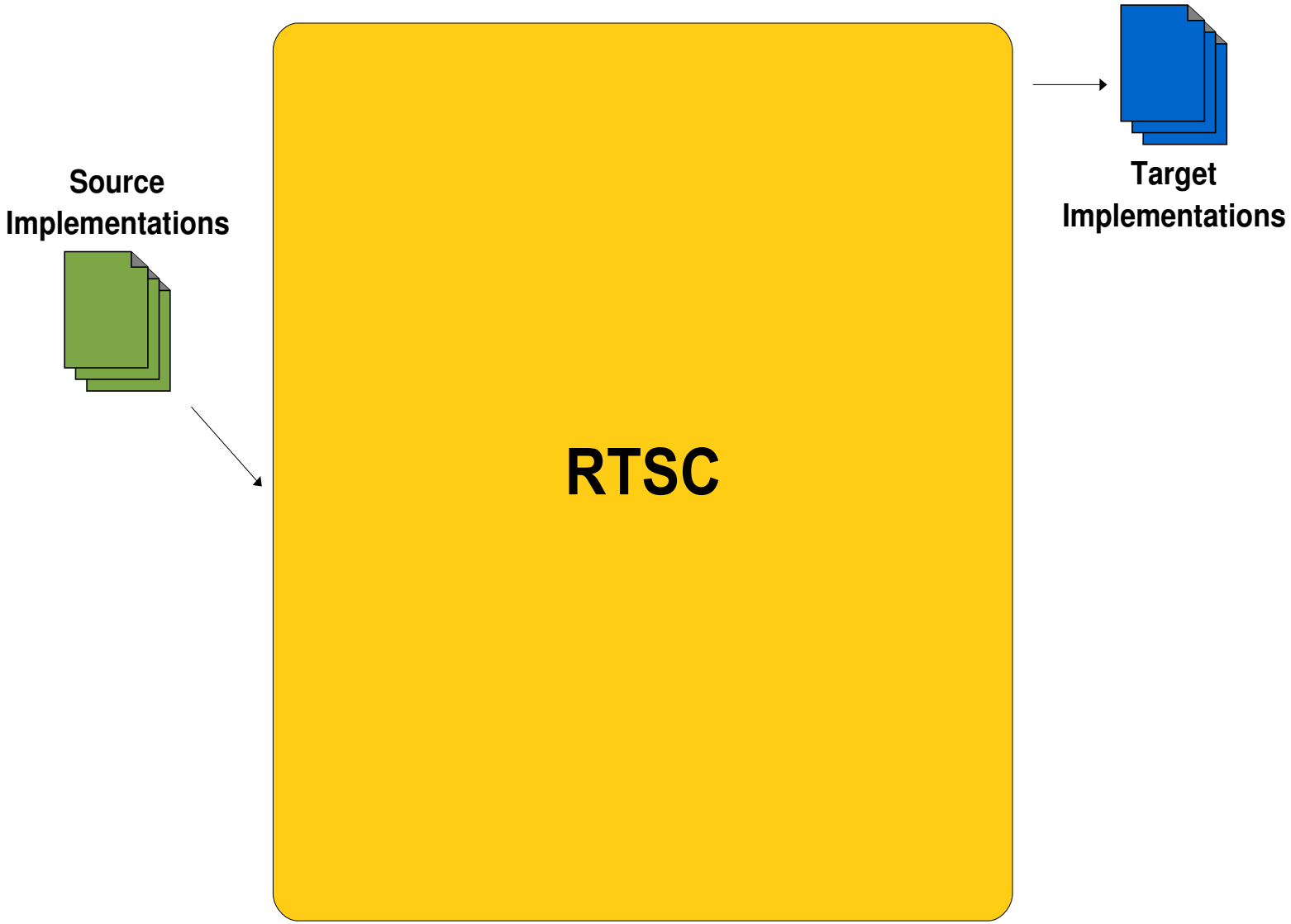
Overview

- Why Migration is a Problem
- Aiding Migration
- Atomic Basic Blocks
- ➔ **The Real Time Systems Compiler**
- Current Status & Future Work

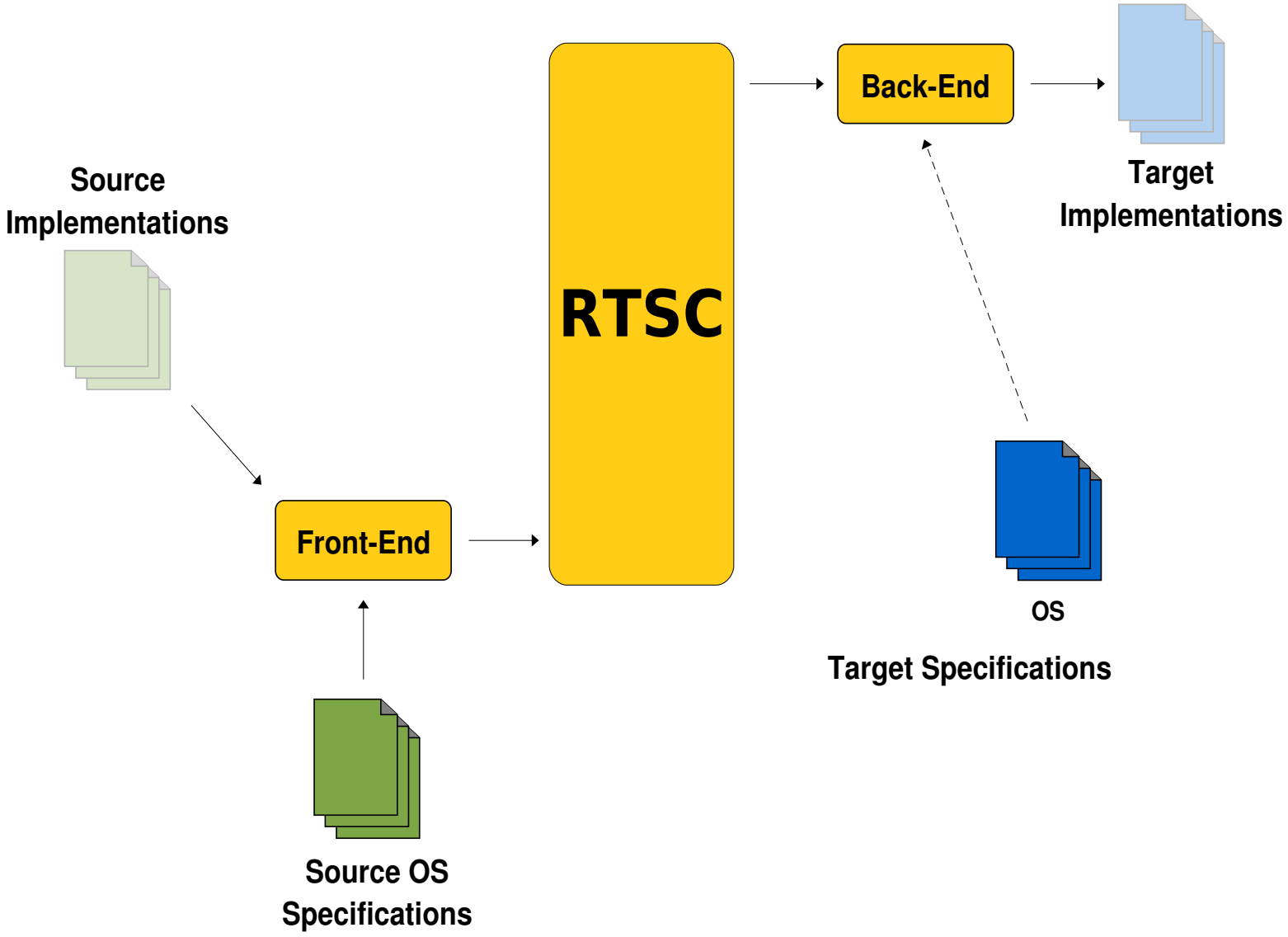
The Real-Time Systems Compiler

- OS aware compiler
  - based on LLVM
- uses ABBs as intermediate representation
  - ABBs are implemented on top of the LLVA
    - intermediate representation employed by the LLVM
    - typed assembler

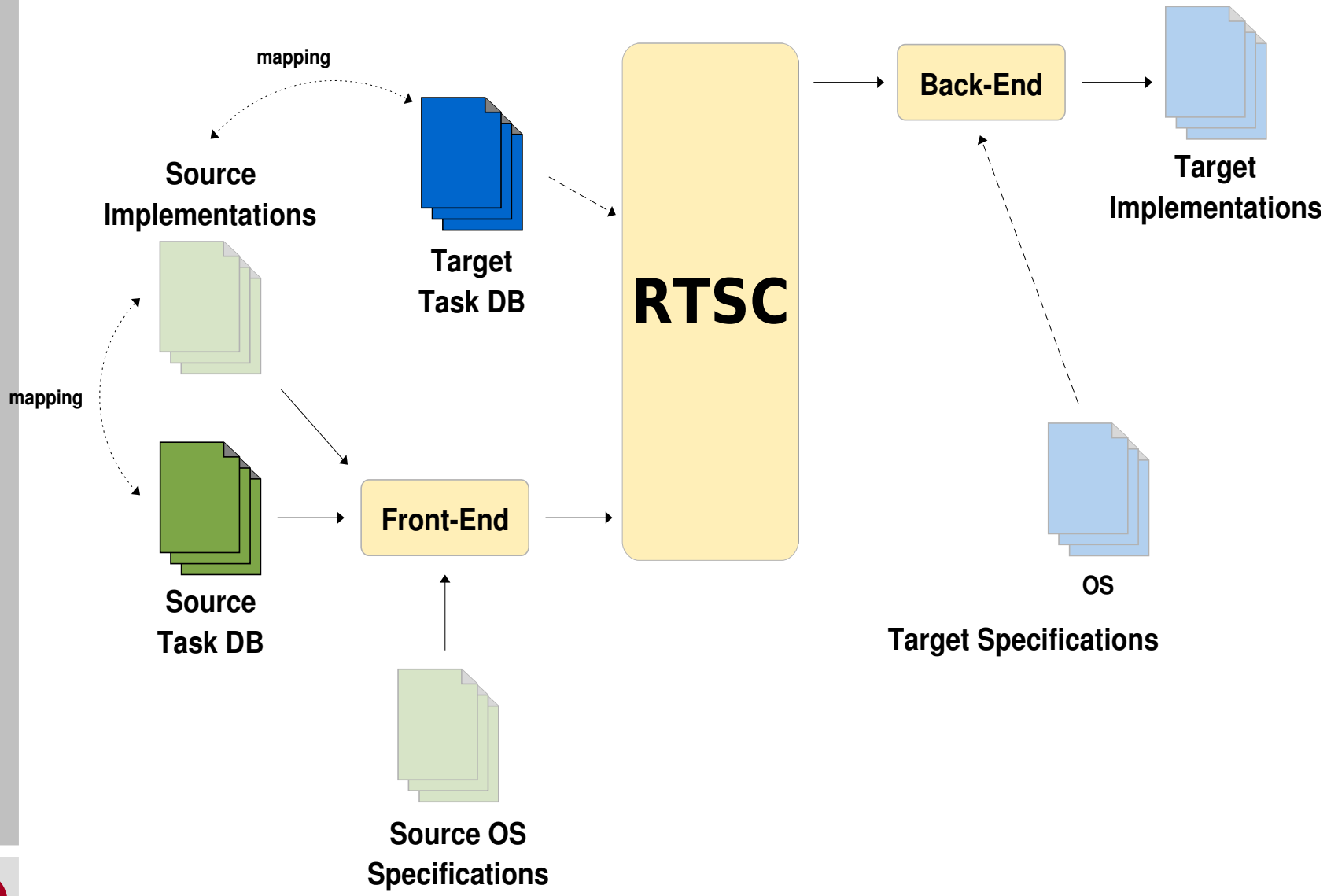
Input & Output



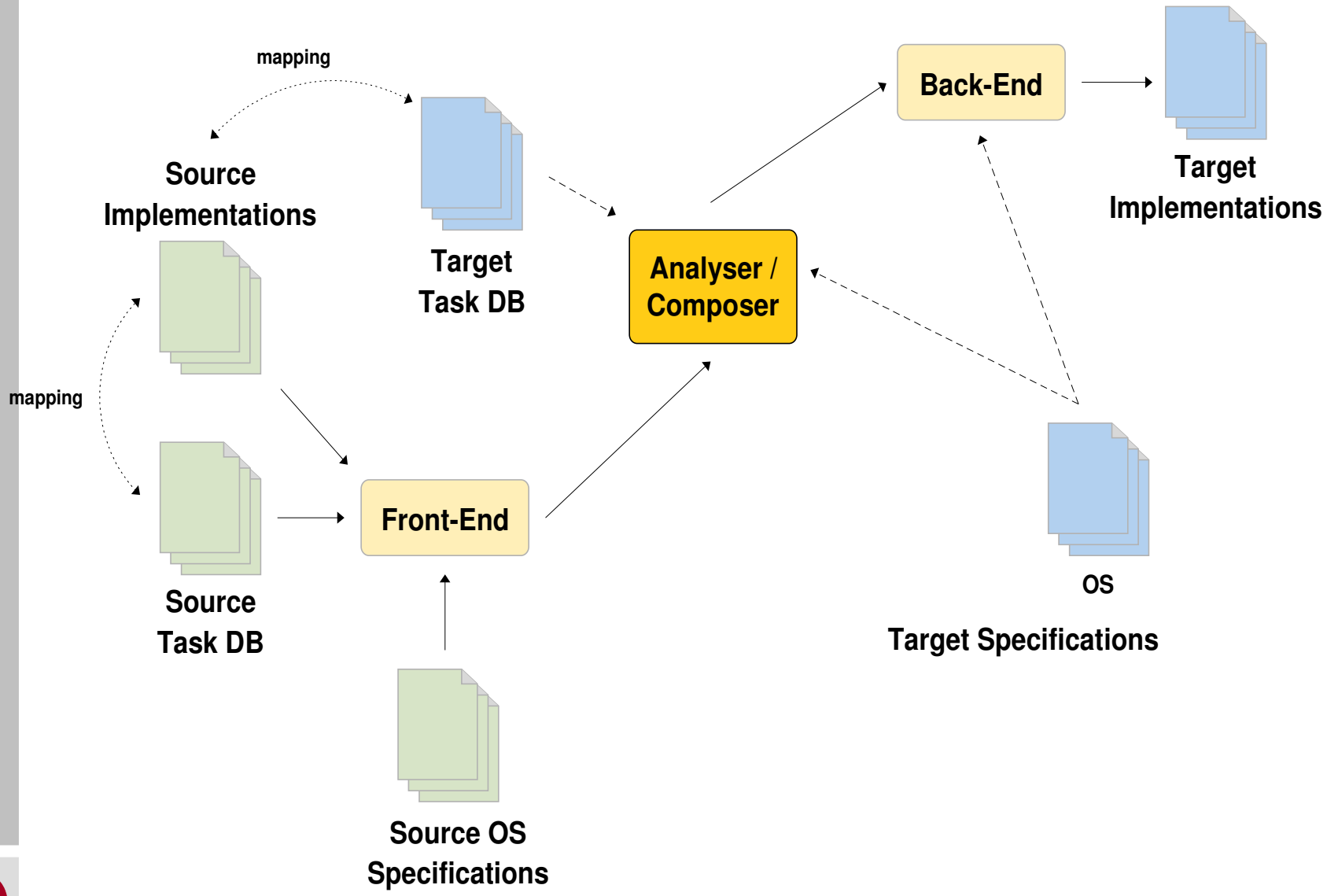
OS-specific Front- and Back-End



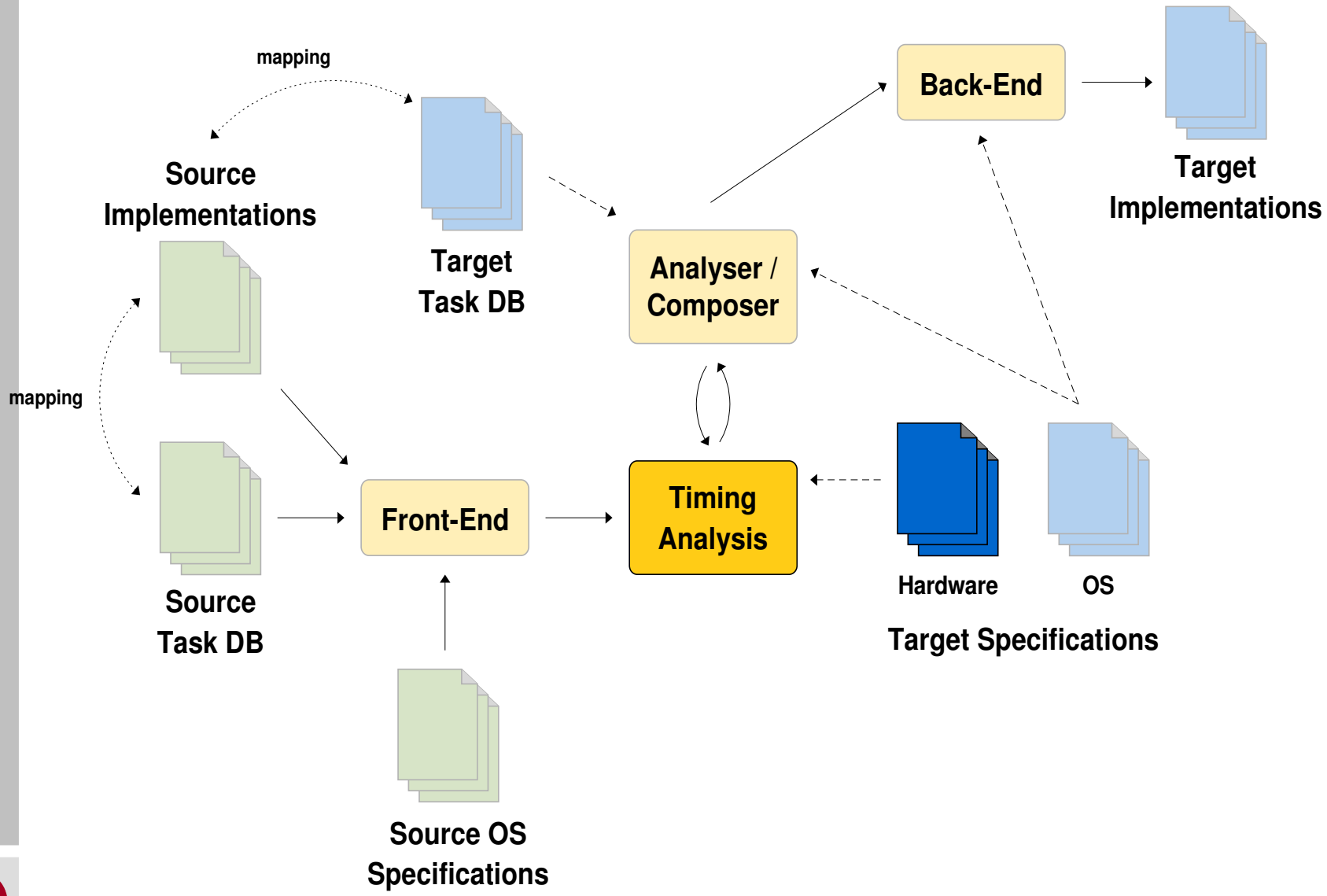
Specifying Events



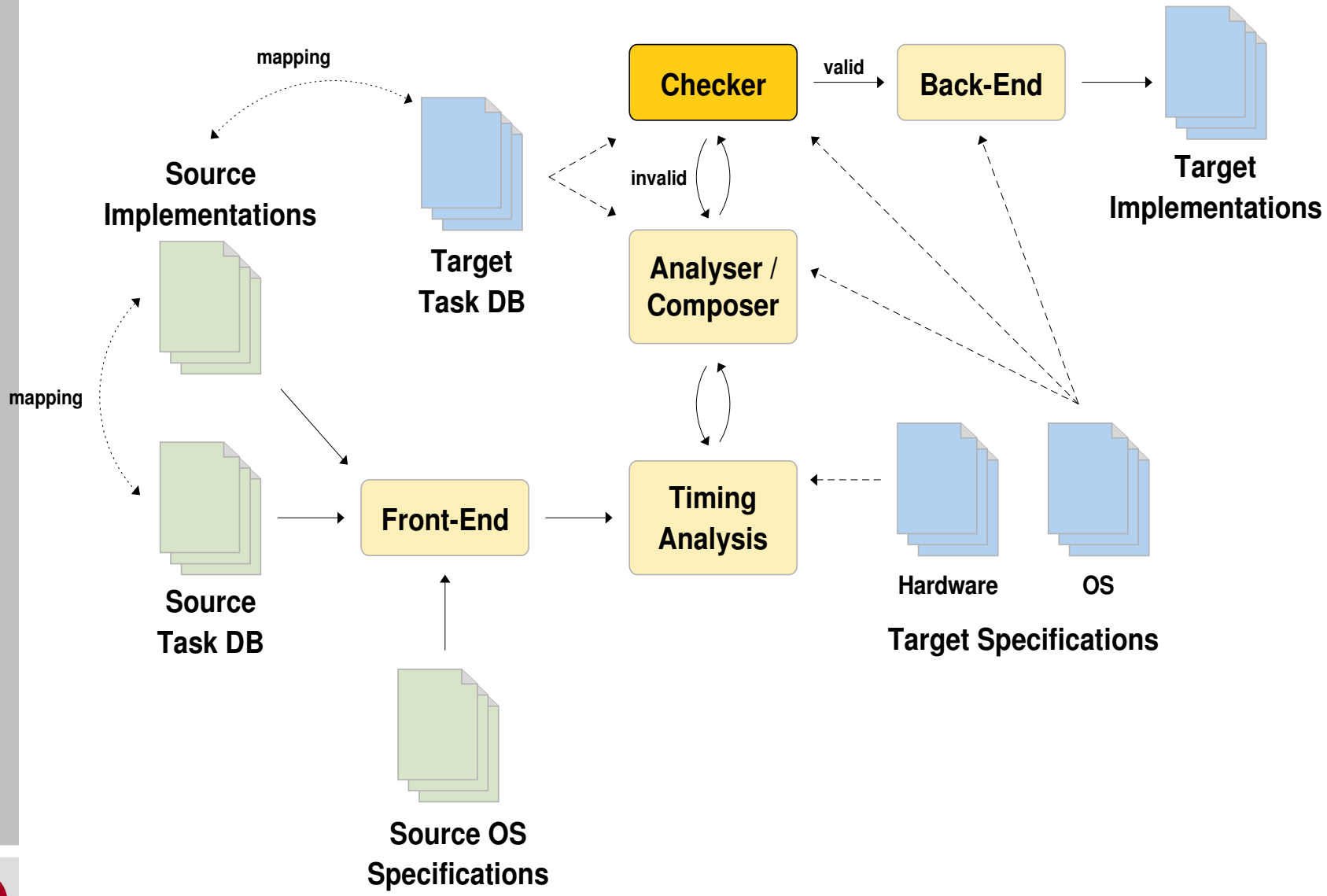
Mapping ABBs to OS mechanisms



WCET Analysis



Schedulability Analysis



Overview

- Why Migration is a Problem
- Aiding Migration
- Atomic Basic Blocks
- The Real Time Systems Compiler: Overview
- ➔ **Current Status & Future Work**

Future Work

- current status
  - early prototype of a C front-end
  - not that exciting: no ABBs except global data dependencies
  - framework for creating ABB-graphs
- future work
  - OSEK OS and OSEK ttOS front-end and back-end

Future Work

- current status
  - early prototype of a C front-end
  - not that exciting: no ABBs except global data dependencies
  - framework for creating ABB-graphs
- future work
  - OSEK OS and OSEK ttOS front-end and back-end

Thank you very much for your attention!