Atomic Basic Blocks

Eine kontrollflussunabhängige Zwischendarstellung für Echtzeitsysteme

Fabian Scheler
Martin Mitzlaff
Wolfgang Schröder-Preikschat

Informatik 4
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander-Universität Erlangen-Nürnberg

http://www4.informatik.uni-erlangen.de/~scheler
Fabian.Scheler@informatik.uni-erlangen.de

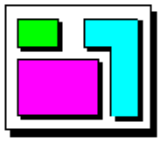
Entwicklung am Automobilmarkt

- **ereignisgesteuerte** Kommunikation
  - CAN

➡

- **zeitgesteuerte** Kommunikation
  - FlexRay

- **föderales** System
  - OSEK/VDX



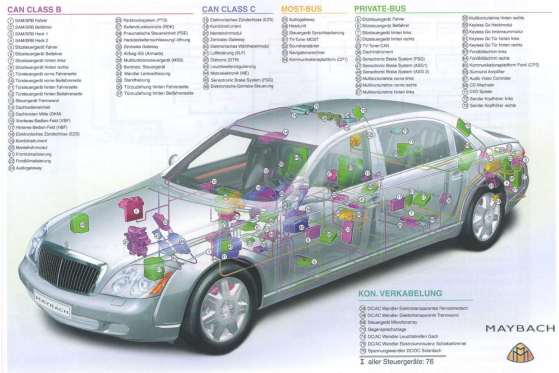
➡

- **integrierte** Systeme
  - AUTOSAR

- **Fail-stop** Semantik
  - ABS, ESP

➡

- **Fail-operational** Semantik
  - {Steer,Break}-by-Wire



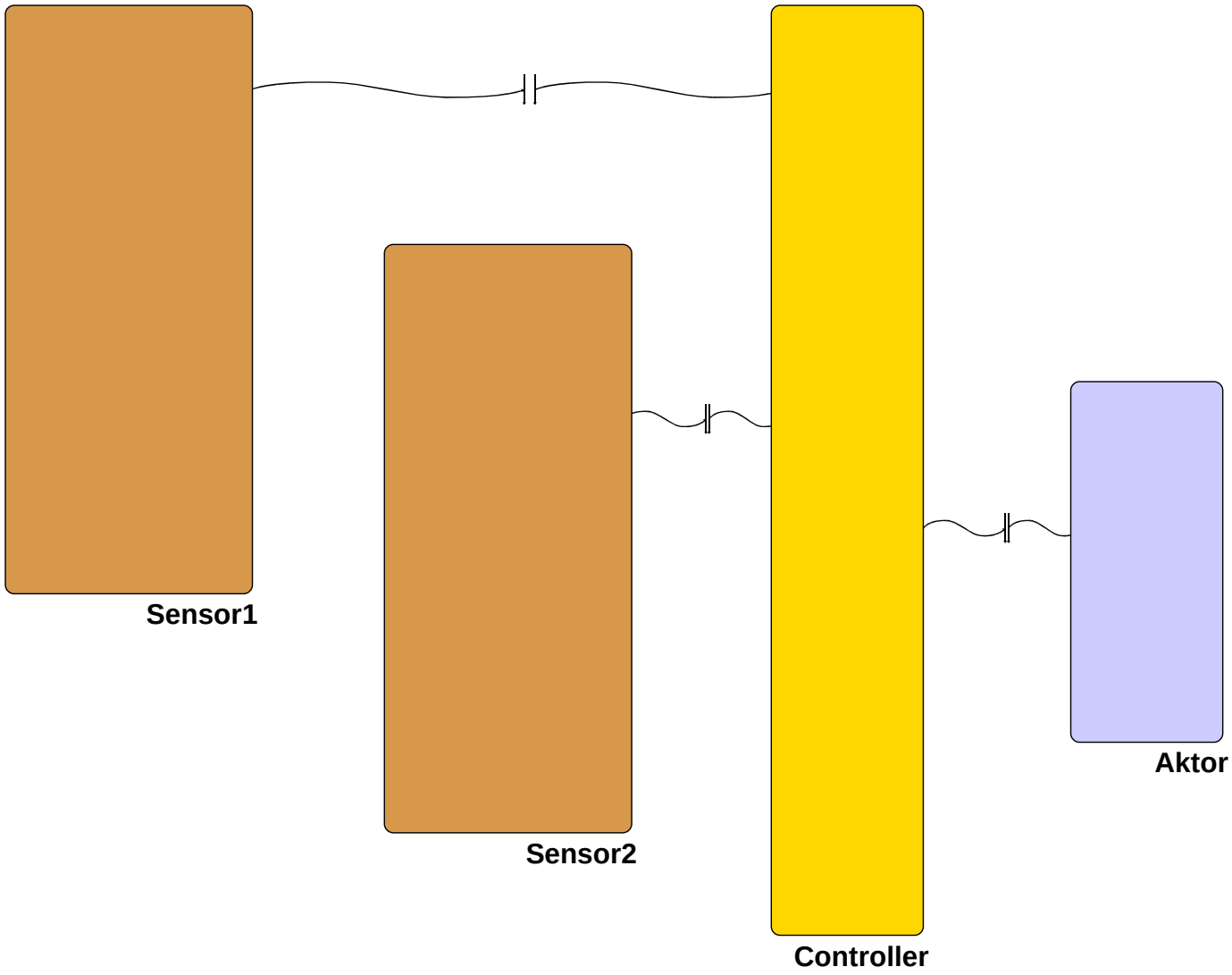
Konsequenzen

- Migration
  - **Ereignissteuerung** → **Zeitsteuerung**
- Legacy Applications
  - falls möglich: Wiederverwendung
    (z.B. virtuelle CAN-Netzwerke auf Basis von FlexRay)
    - nicht sicherheitskritische Subsysteme, z.B. Komfortanwendungen
  - sonst: Migration/Portierung
    - sicherheitskritische Subsyteme, z.B. ABS, ESP
- Portierung
  - arbeitsintensiv
  - Fehlerquelle

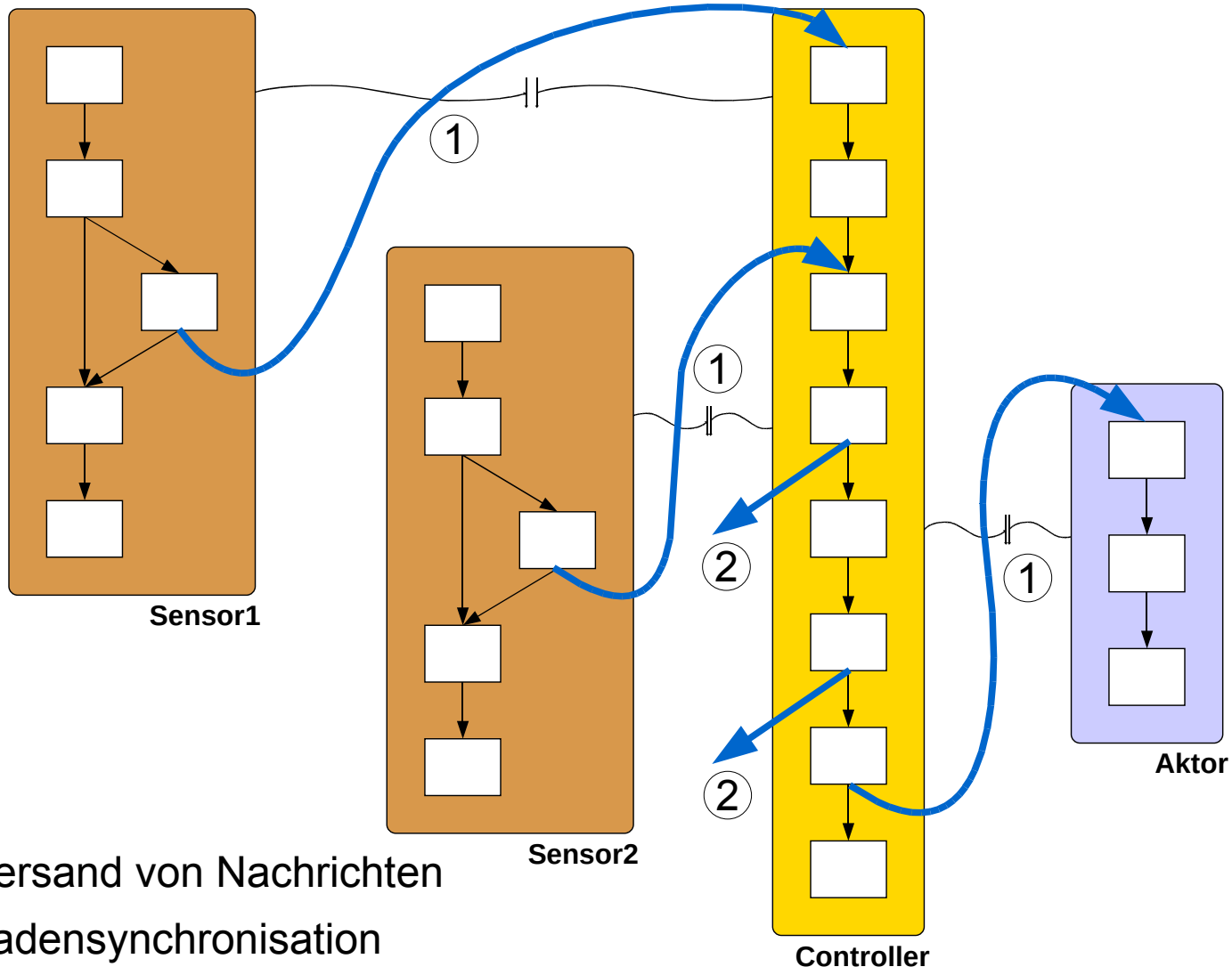
Überblick

- **Das Migrationsproblem**
- Ist Migration sinnvoll?
- Atomic Basic Blocks
- Der Real Time Systems Compiler
- Erzeugung von Atomic Basic Blocks
- Ausblick

Ein triviales Beispiel



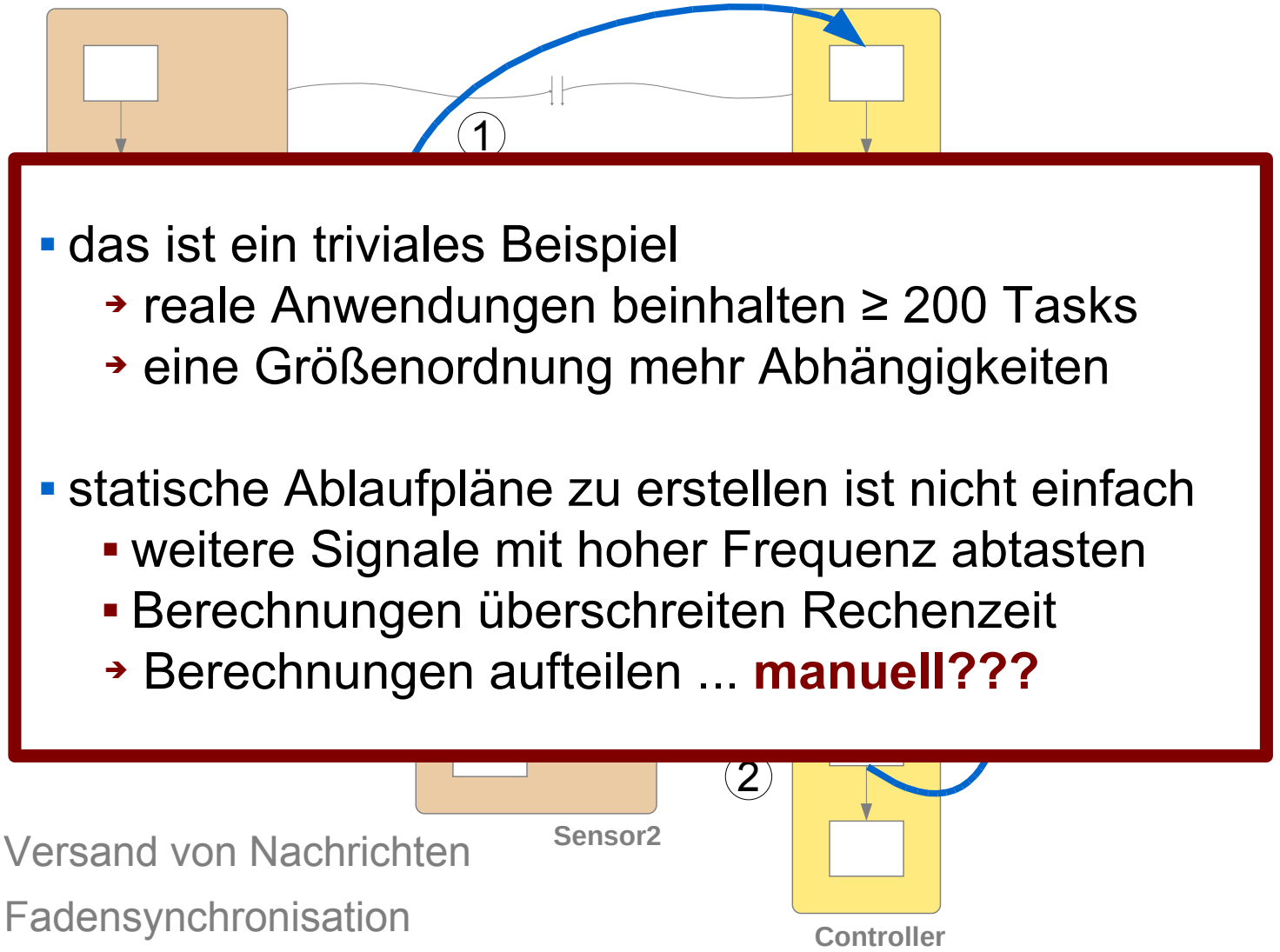
Explizit modellierte Abhängigkeiten



① Versand von Nachrichten

② Fadensynchronisation

Explizit modellierte Abhängigkeiten



- das ist ein triviales Beispiel
  - ➔ reale Anwendungen beinhalten ≥ 200 Tasks
  - ➔ eine Größenordnung mehr Abhängigkeiten
- statische Ablaufpläne zu erstellen ist nicht einfach
  - weitere Signale mit hoher Frequenz abtasten
  - Berechnungen überschreiten Rechenzeit
  - ➔ Berechnungen aufteilen ... **manuell???**

① Versand von Nachrichten

② Fadensynchronisation

Überblick

- Das Migrationsproblem
- **Ist Migration sinnvoll?**
- Atomic Basic Blocks
- Der Real Time Systems Compiler
- Erzeugung von Atomic Basic Blocks
- Ausblick

Ist Migration sinnvoll?

- Lässt sich eine Migration a priori ausschließen?
  - wofür eignet sich Zeit- bzw. Ereignissteuerung
  - gibt es Anwendungsfälle, die ein bestimmtes Paradigma
    - bevorzugen oder sogar
    - ausschließen

➔ nicht-funktionale Eigenschaften

- Gibt es reale Anwendungsszenarien?
  - Will man komplette Systeme migrieren?
  - Gibt es andere Gründe für eine Migration?

nicht-funktionale Eigenschaften

	zeitgesteuert		ereignisgesteuert	
Analysierbarkeit	statische Ablaufpläne	✓	Antwortzeitanalyse	✓
Vorhersagbarkeit	statische Ablaufpläne	✓	dynamischer Ablaufplan, jedoch ist Determinismus ausreichend	✓
Testbarkeit	WCET Analyse + statische Ablaufpläne	✓	WCET Analyse + Antwortzeitanalyse	✓
Erweiterbarkeit	Neuberechnung statischer Ablaufpläne	✓	erneute Antwortzeitanalyse	✓
Fehlertoleranz	Replikdeterminismus	✓	Leader-Follower, Gruppenkommunikation	✗
Ressourcennutzung (nicht-periodische Ereignisse)	Abfragebetrieb	✗	Unterbrechungsgesteuert, {aperiodische, sporadische...} Server	✓

nicht-funktionale Eigenschaften

- liefern kein endgültiges Entscheidungskriterium
  - allenfalls tendieren
    - Fehlertoleranz zu **zeitgesteuerten** Systemen
    - nicht-periodische Ereignisse zu **ereignisgesteuerten** Systemen

- **zeitgesteuertes** System
  - sicherheitskritisch: Absicherung durch Fehlertoleranz
  - **wenige nicht-periodische** Ereignisbehandlungen
  - → Polling

- **ereignisgesteuertes** System
  - viele nicht-periodische Ereignisse
  - **wenige sicherheitskritische** Ereignisbehandlungen
  - → Leader-Follower

Anwendungsszenarien

- Migration einzelner Subsysteme
  - Wiederverwendung in zeit- bzw. ereignisgesteuerten Systemen
- Beispiel: ESP
  - mehrere Sensoren und Aktoren
  - gemeinsame Datenstrukturen notwendig
- **ereignisgesteuerte** Systeme:
  - zwischen versch. Knoten: OSEK COM Messages
  - auf demselben Knoten: OSEK Ressourcen und OSEK Events
- **zeitgesteuerte** Systeme:
  - statisch berechnete Ablaufpläne
  - Berücksichtigung aller Abhängigkeiten

Überblick

- Das Migrationsproblem
- Ist Migration sinnvoll?
- **Atomic Basic Blocks**
- Der Real Time Systems Compiler
- Erzeugung von Atomic Basic Blocks
- Ausblick

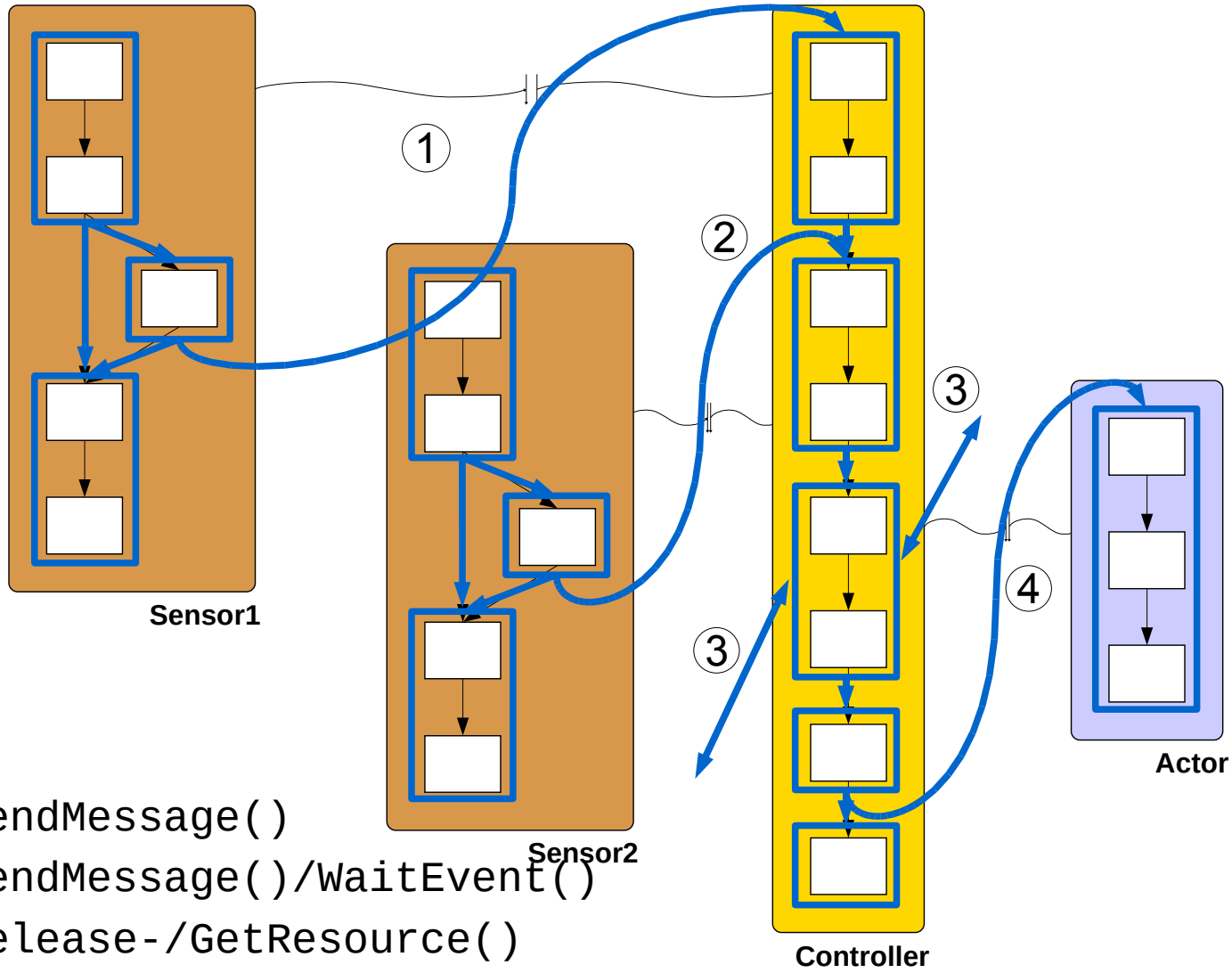
Ansatz

- Entkopplung von Anwendung und Laufzeitsystem
  - Zwischendarstellung
  - unabhängig von der Kontrollflussabstraktion
- Kombination verschiedener
  - Front-Ends und
  - Back-Ends
- ähnlich zum Übersetzerbau
- Zwischendarstellung
  - Kontrollflussgraphen (CFG)
  - Basisblöcke

Atomic Basic Blocks

- Abhängigkeiten zwischen verschiedenen CFGs
  - Datenabhängigkeiten
  - explizit modellierte logische und zeitliche Abhängigkeiten
  - gegenseitiger Ausschluss
- ABB-Graph überlagert einen Wald aus CFGs
- ABBs aggregieren mehrere Basisblöcke
- ABBs werden immer über **genau einen** Basisblock
  - verlassen und
  - betreten
- Grenzen ABB
  - abzweigen/einmünden von anderen CFGs
  - kritische Abschnitte

Atomic Basic Blocks



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

Überblick

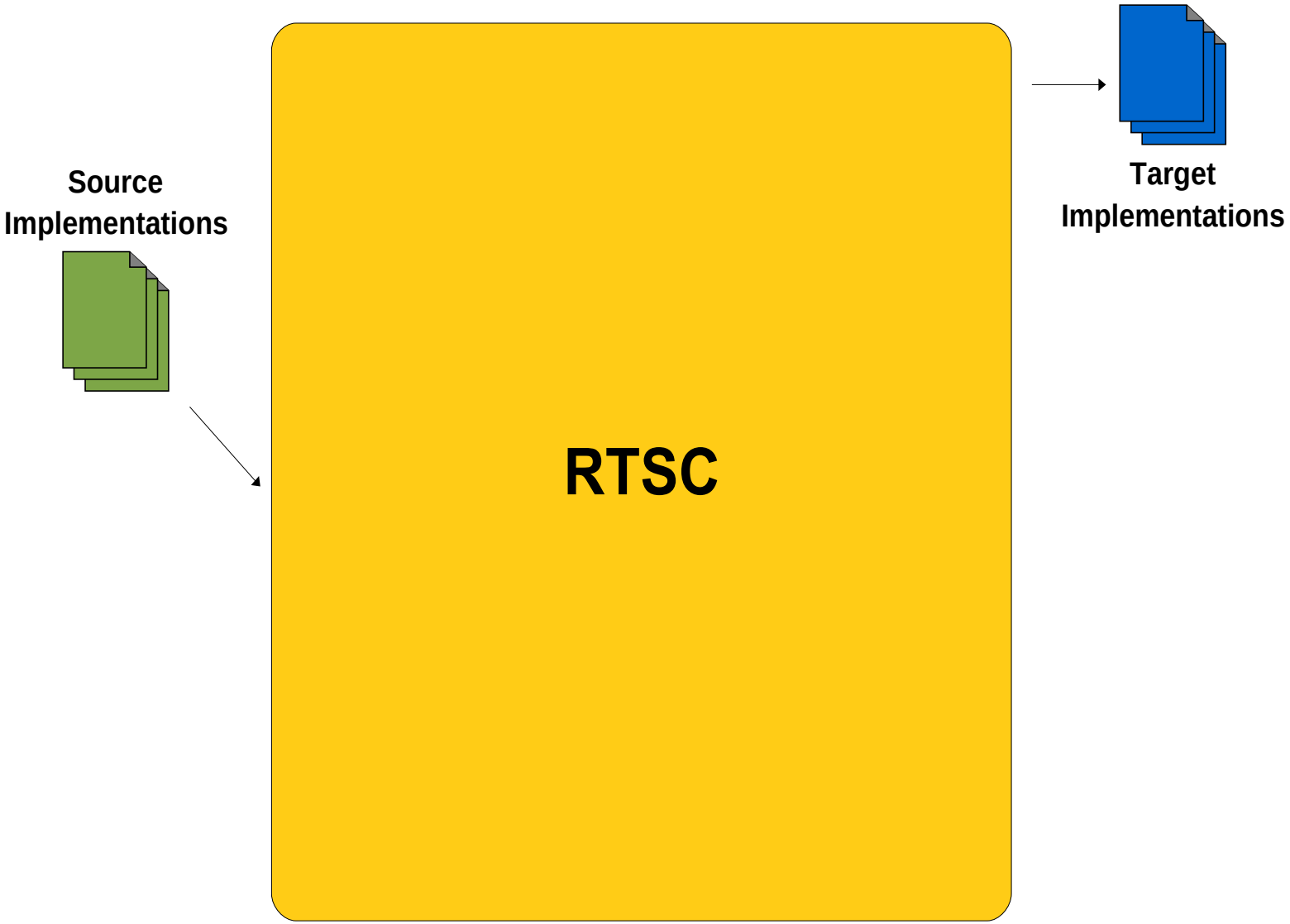
- Das Migrationsproblem
- Ist Migration sinnvoll?
- Atomic Basic Blocks
- **Der Real Time Systems Compiler**
- Erzeugung von Atomic Basic Blocks
- Ausblick

Der Real-Time Systems Compiler

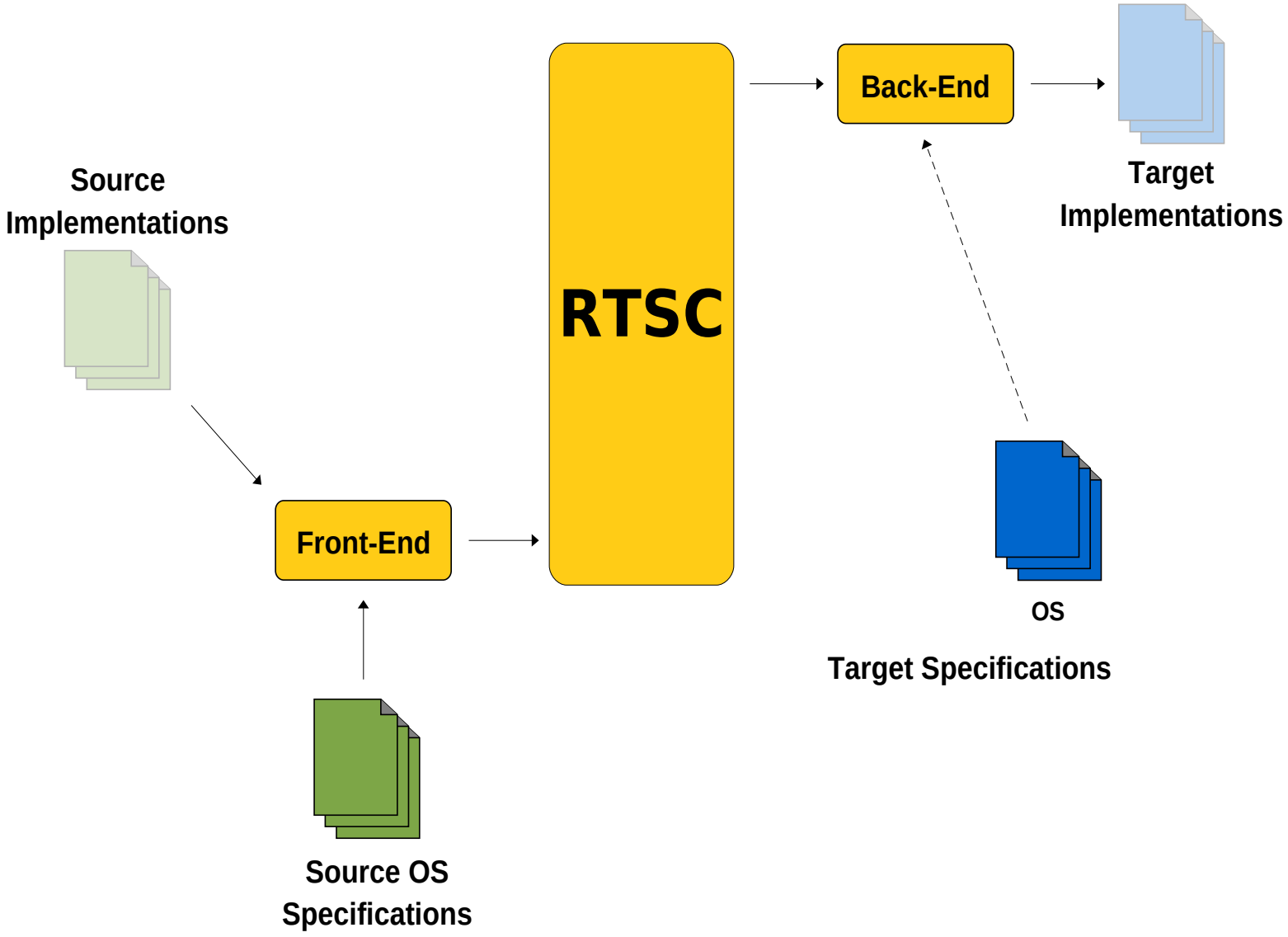
- betriebssystemgewahrer Übersetzer
  - basierend auf der LLVM (Low Level Virtual Machine)

- verwendet ABBs als Zwischendarstellung
  - → ABBs sind basierend auf dem LLVM-Assembler implementiert
    - von der LLVM verwendete Zwischendarstellung
    - typisierter Assembler

Eingabe & Ausgabe



BS-gewahres Front- und Back-End



Ereignisse

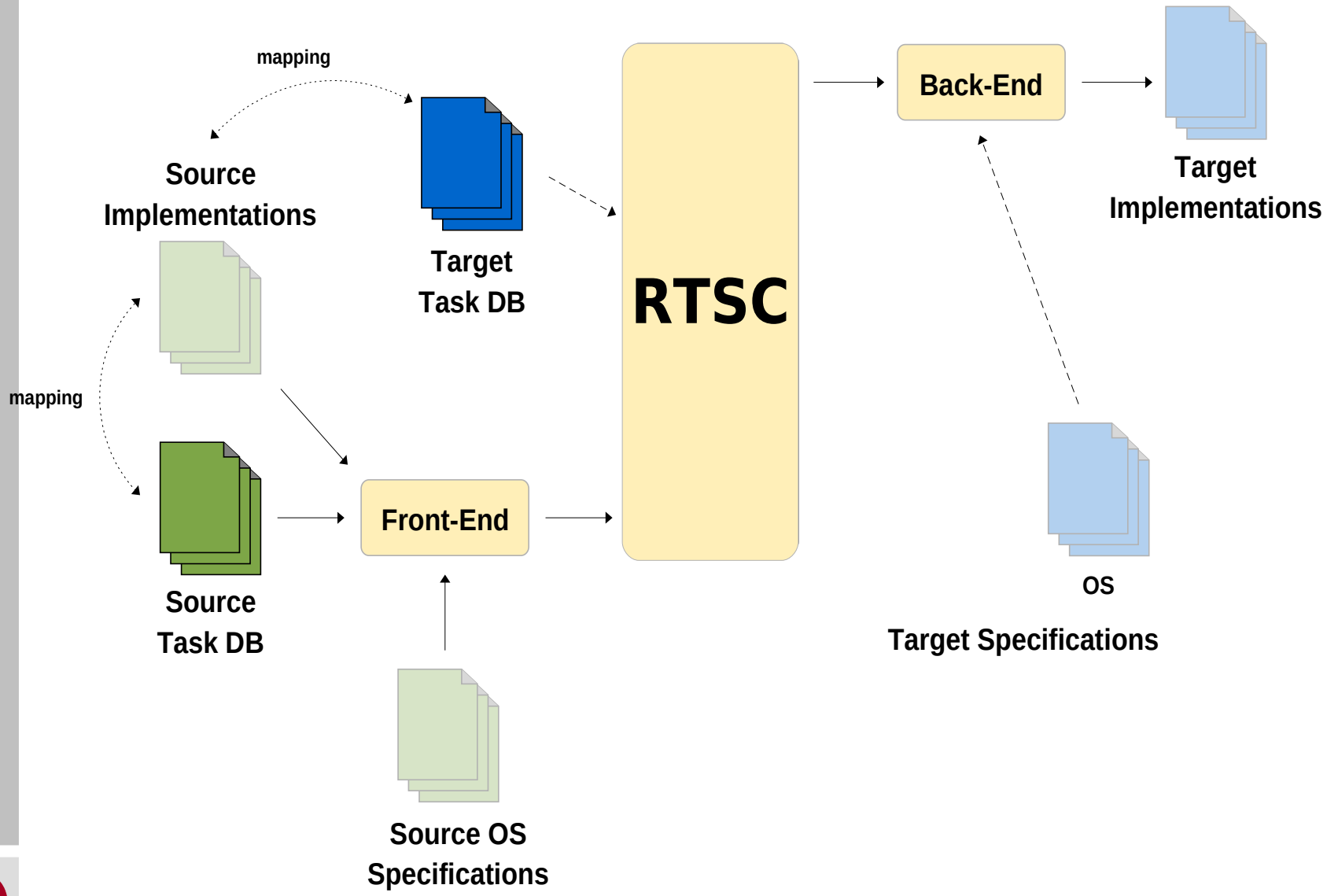
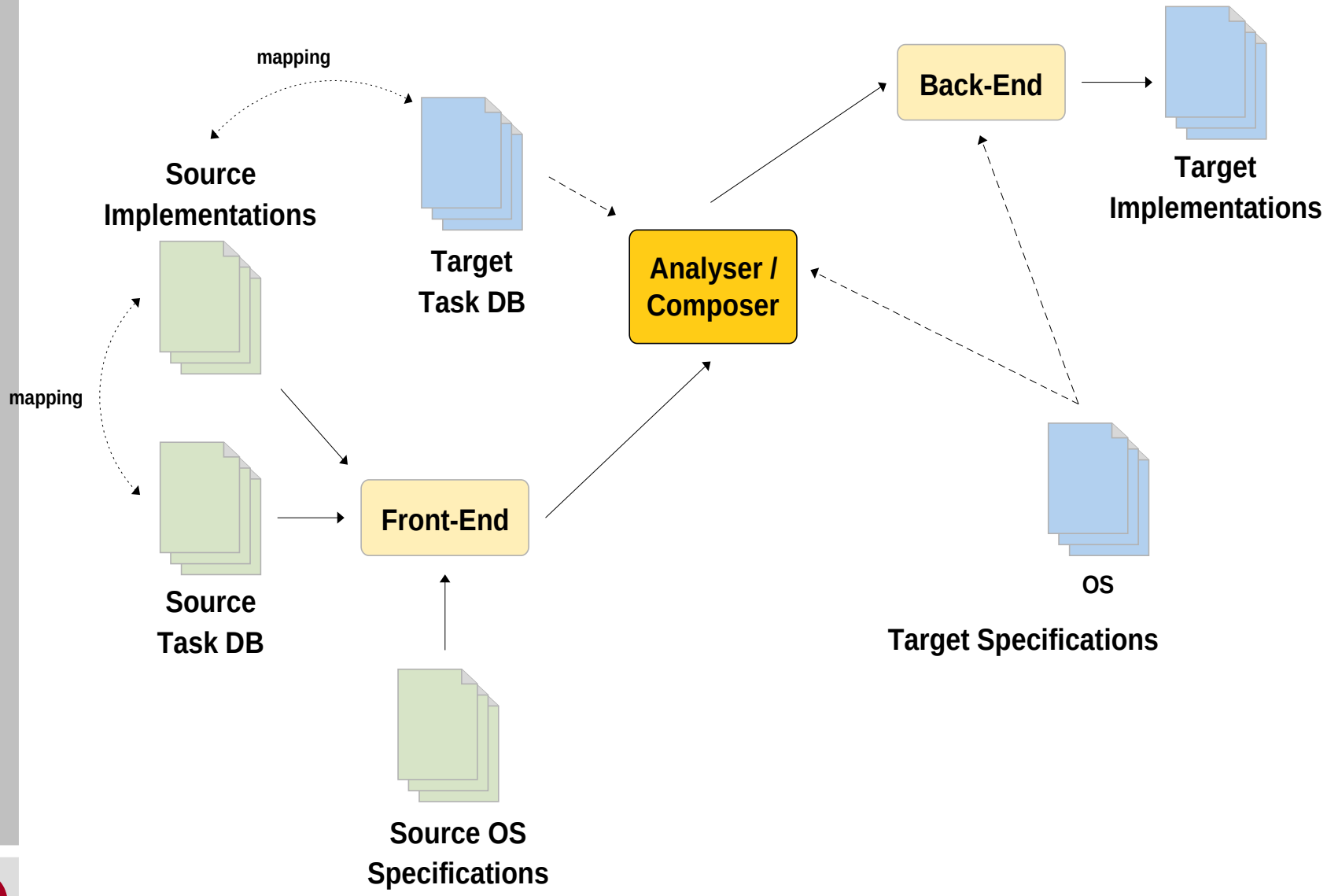
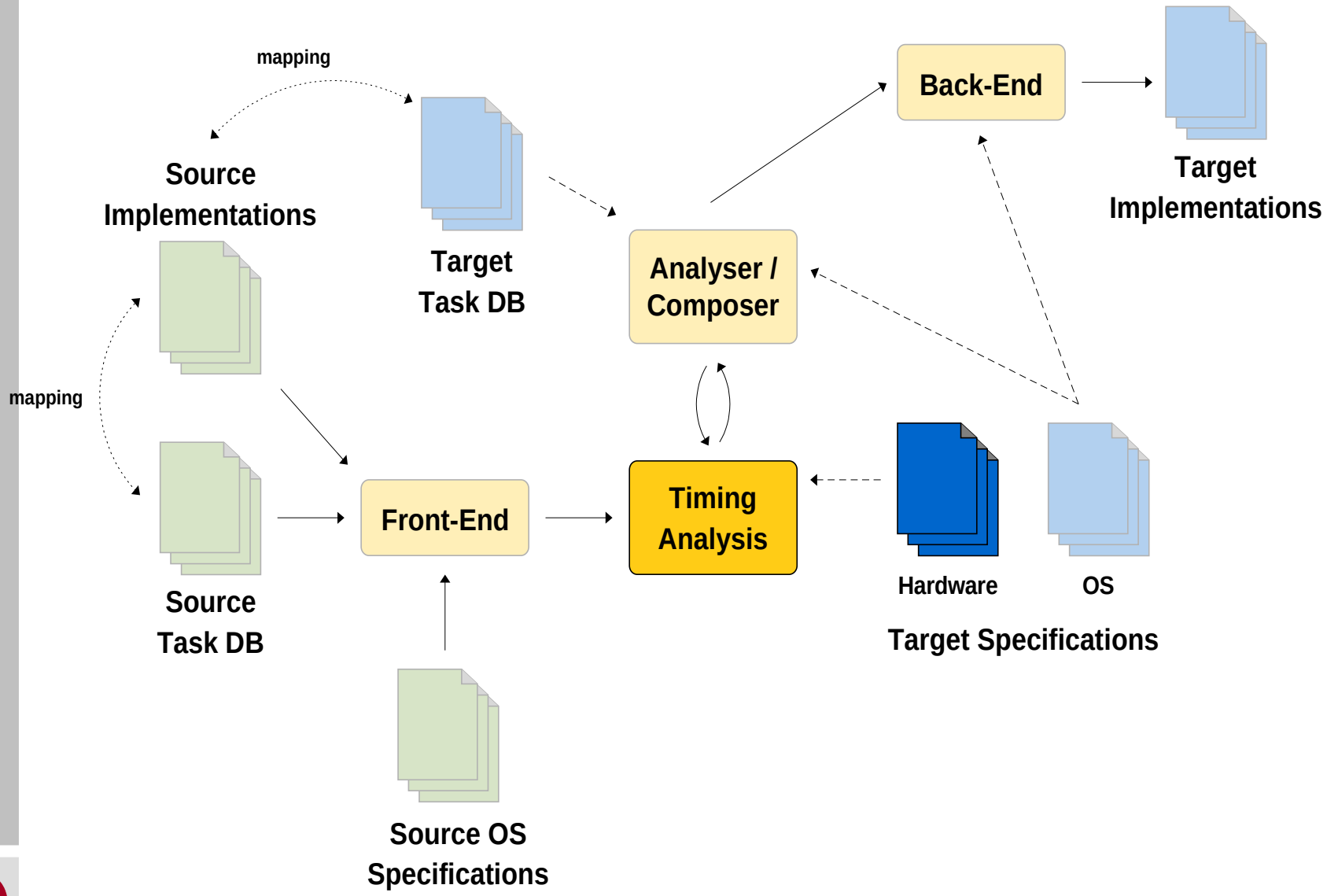

mapping
Source
Implementations
Target
Task DB
RTSC
Back-End
Target
Implementations
mapping
Source
Task DB
Front-End
OS
Target Specifications
Source OS
Specifications

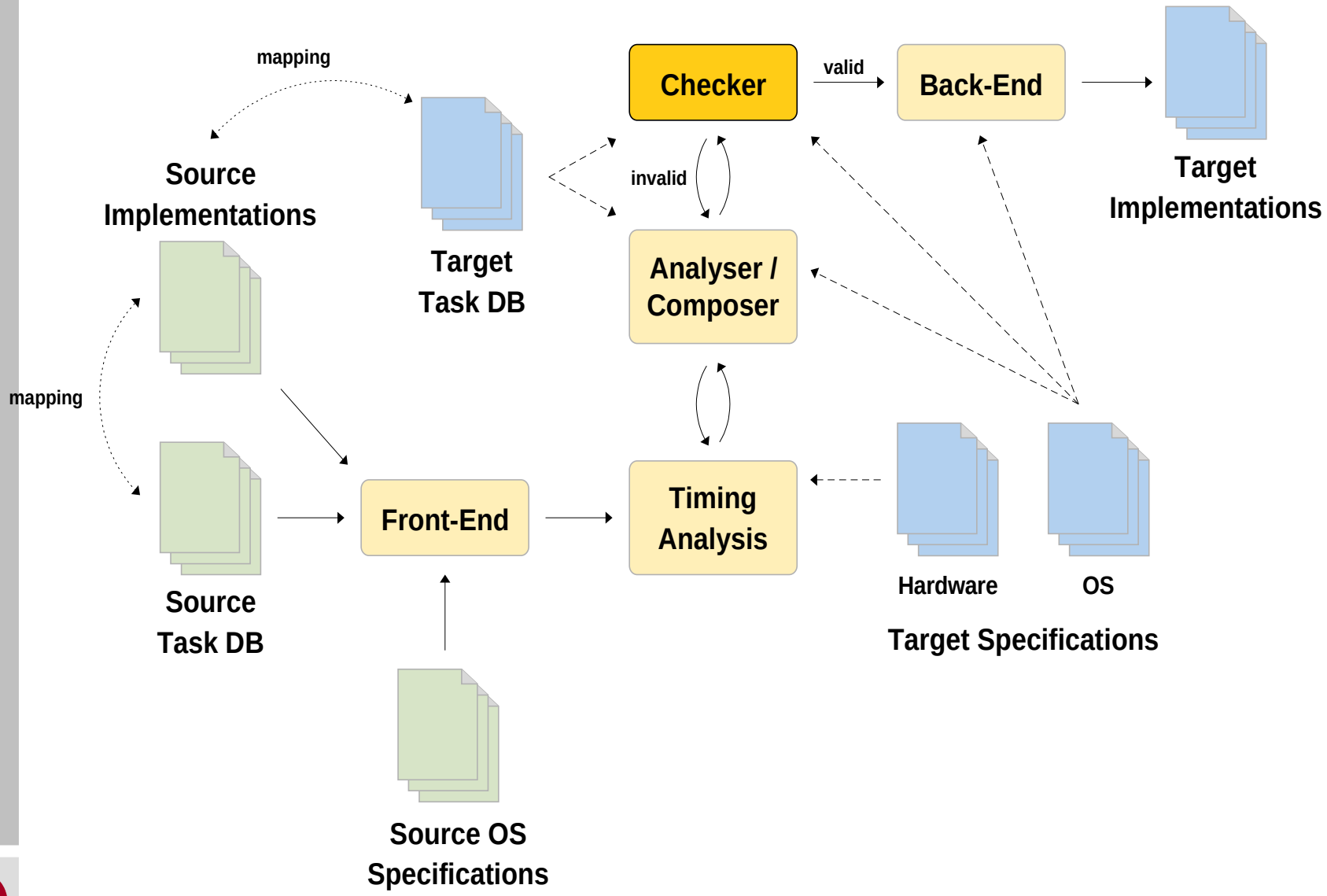
Abbildung von ABBs



WCET Analyse



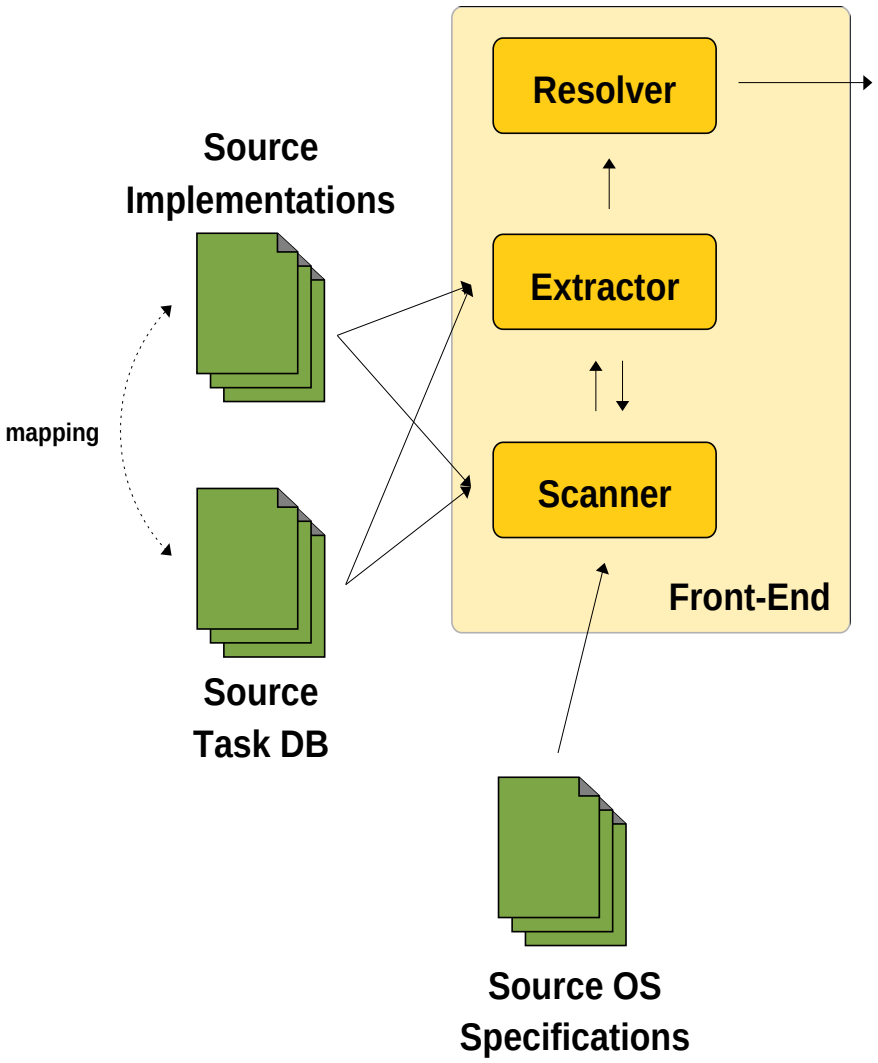
Planbarkeitsanalyse



Überblick

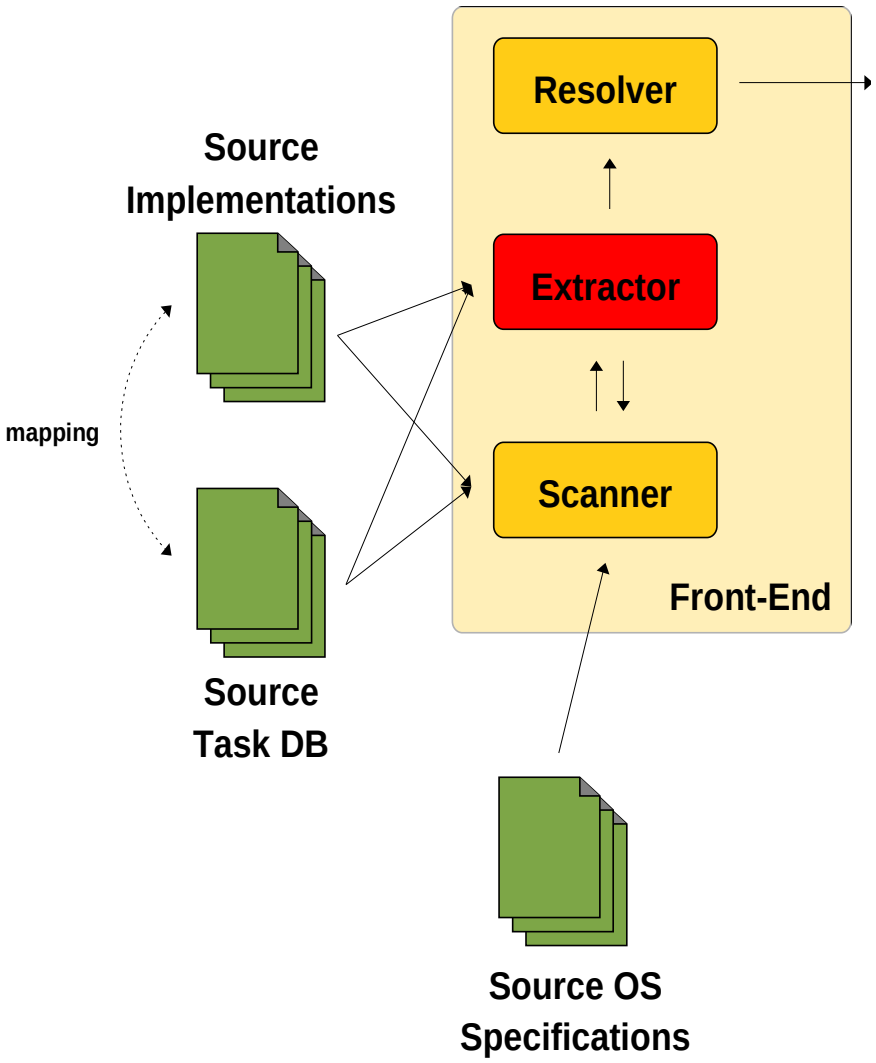
- Das Migrationsproblem
- Ist Migration sinnvoll?
- Atomic Basic Blocks
- Der Real Time Systems Compiler
- **Erzeugung von Atomic Basic Blocks**
- Ausblick

RTSC: Einordnung



- Resolver
  - verknüpft ABB-Endpunkte
  - erzeugt globalen Graph
- Extractor
  - generiert ABBs
- Scanner
  - sucht ABB-Endpunkte
  - systemabhängig

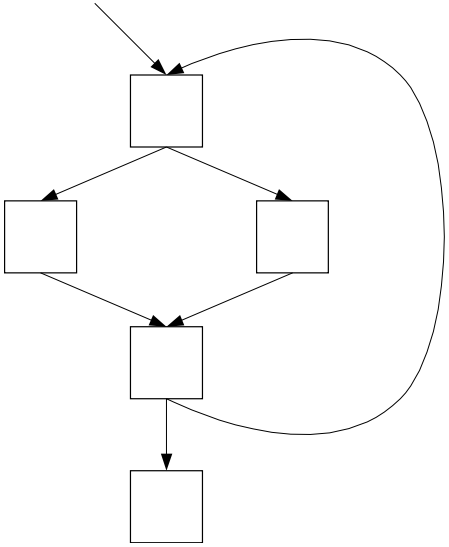
RTSC: Einordnung



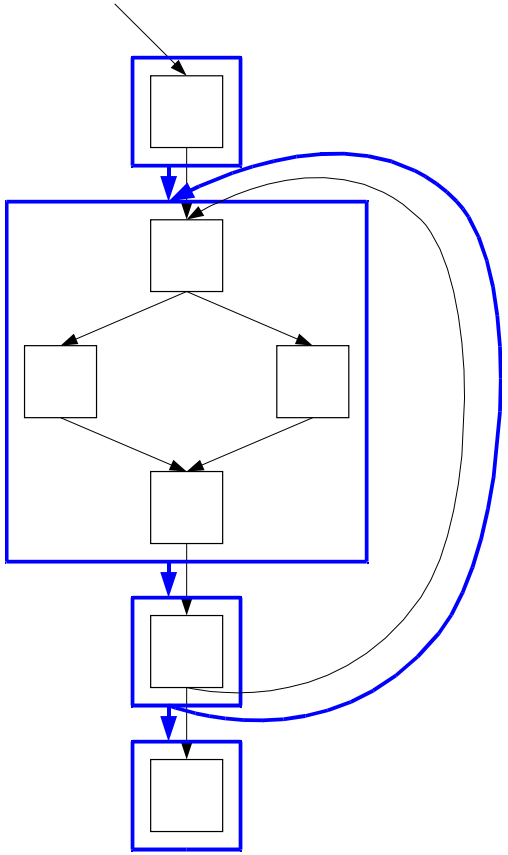
- Resolver
  - verknüpft ABB-Endpunkte
  - erzeugt globalen Graph
- Extractor
  - generiert ABBs
- Scanner
  - sucht ABB-Endpunkte
  - systemabhängig

Eingaben und Ausgaben

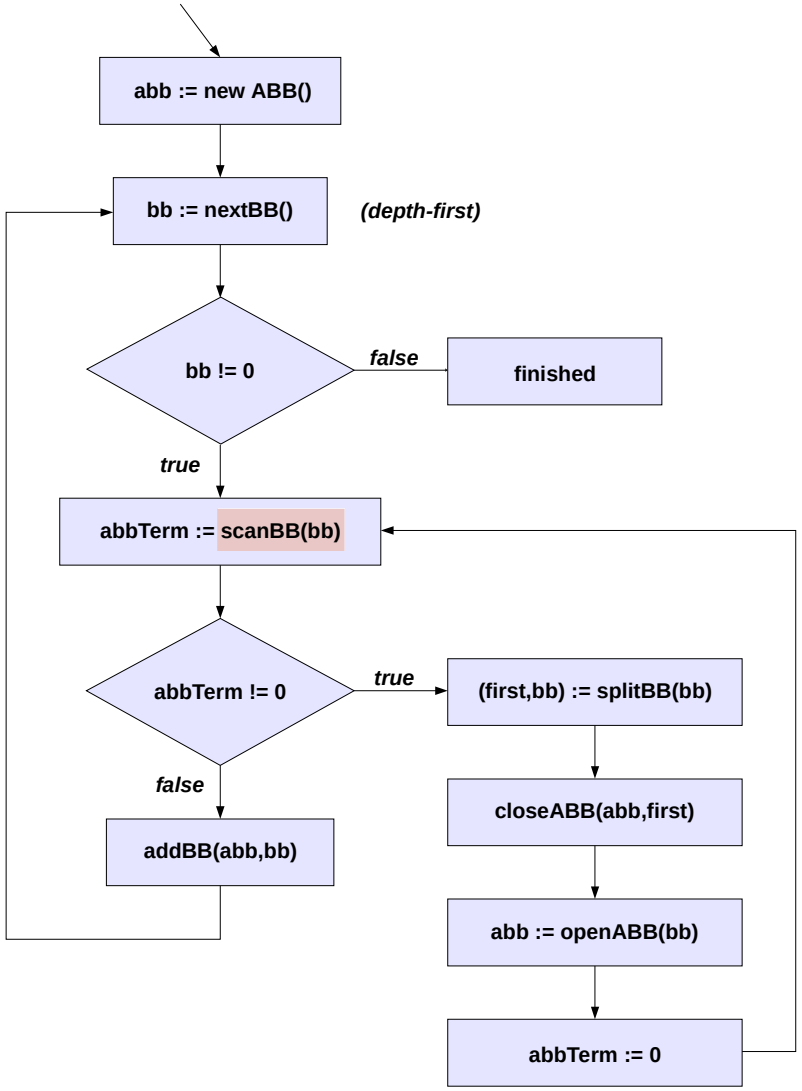
- Eingabe
  - reduzierbarer Kontrollflussgraph
    - (natürliche) Schleifen
    - Verzweigungen (if/switch)
    - kein `goto`

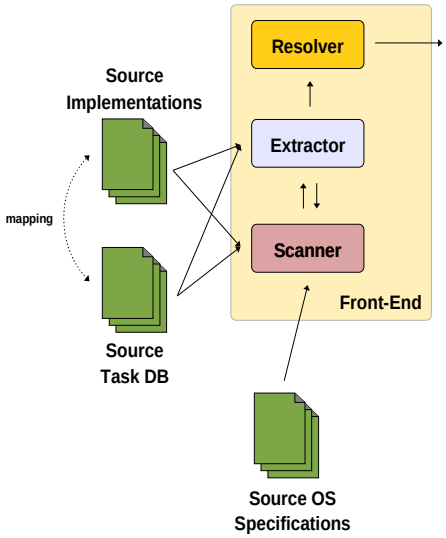


- Ausgabe
  - ABB-Graph
  - überlagert Kontrollflussgraph

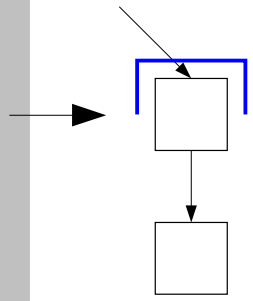


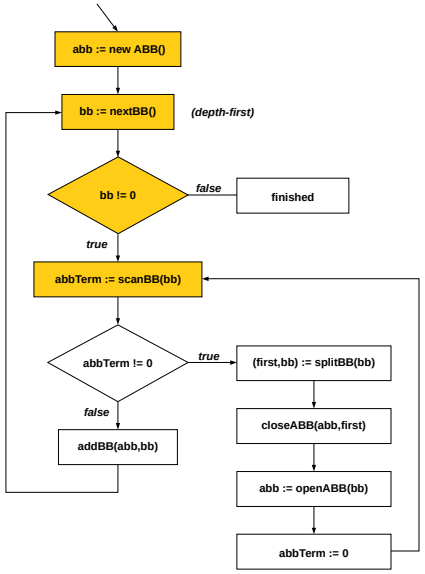
Algorithmus - Überblick





Sequenzen

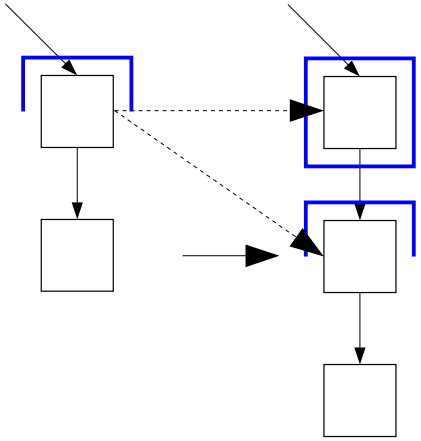




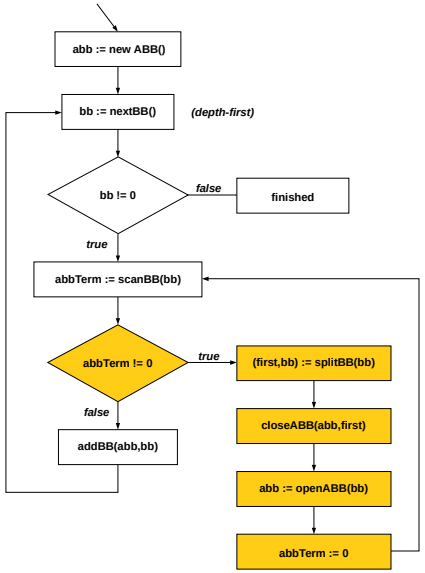
(1)

(1) hole und scanne ersten Basisblock

Sequenzen



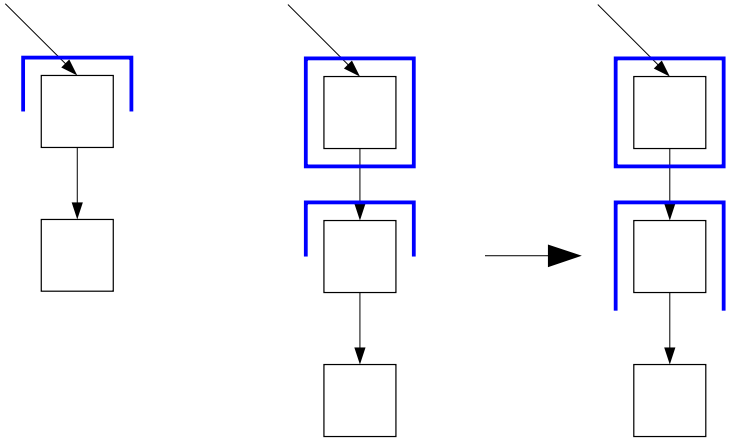
(1) (2)



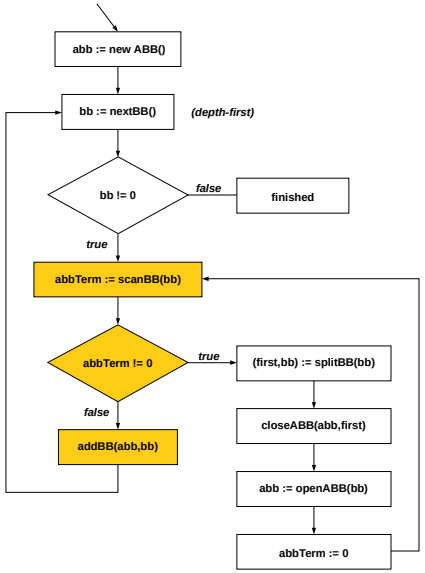
(1) hole und scanne ersten Basisblock

(2) ein ABB-Endpunkt wurde gefunden
- splitte Basisblock
- schließe aktuellen ABB und öffne neuen ABB

Sequenzen



(1) (2) (3)

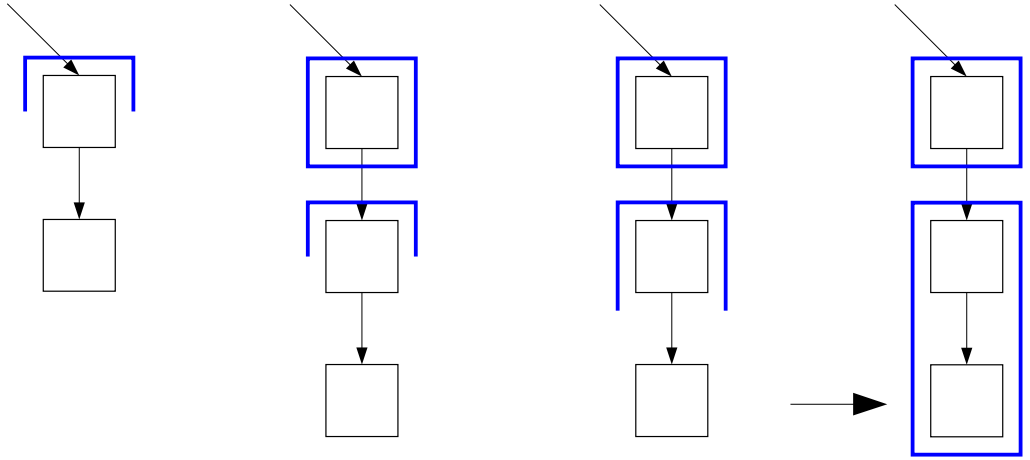


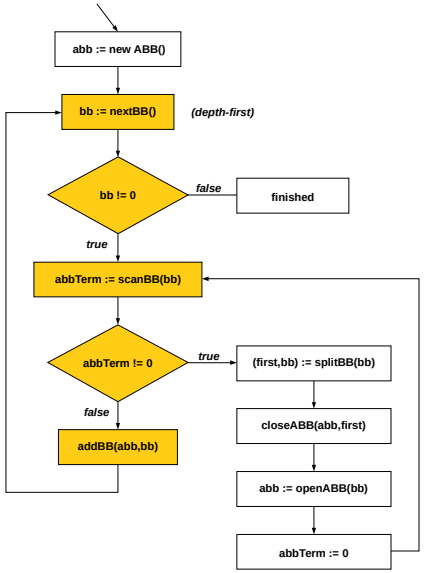
(1) hole und scanne ersten Basisblock

(2) ein ABB-Endpunkt wurde gefunden
- splitte Basisblock
- schließe aktuellen ABB und öffne neuen ABB

(3) scanne zweiten Teil des Basisblocks

Sequenzen





(1) hole und scanne ersten Basisblock

(2) ein ABB-Endpunkt wurde gefunden
- splitte Basisblock
- schließe aktuellen ABB und öffne neuen ABB

(3) scanne zweiten Teil des Basisblocks

(4) nächsten Basisblock holen, scannen und zum ABB hinzufügen

Verzweigungen/Schleifen

- Erkennung → Tiefensuche
  - Nachfolger im CFG
    - Teil eines ABB
    - ABB ist bereits abgeschlossen
- Transformation existierender ABBs
  - splitten von ABBs nach
    - nach verzweigenden Knoten
    - zusammenführenden Knoten
    - und vor Schleifenköpfen
    - Knoten, die eine Schleife verlassen
- verschachtelte Kontrollkonstrukte
  - rekursiv

Überblick

- Das Migrationsproblem
- Ist Migration sinnvoll?
- Atomic Basic Blocks
- Der Real Time Systems Compiler
- Erzeugung von Atomic Basic Blocks
- **Ausblick**

Ausblick

- Status
  - einfach Prototyp eines C-Front-Ends
  - Algorithmus zur Erzeugung von ABB-Graphen
  - Abhängigkeiten durch
    - globale Variablen
    - Versenden/Empfangen von Nachrichten (in Arbeit)
    - Abzweigung von Ereignisbehandlungen
- geplant
  - Front-Ends und Back-Ends für
    - OSEK OS / AUTOSAR OS
    - OSEK ttOS

Ausblick

- Status
  - einfach Prototyp eines C-Front-Ends
  - Algorit
  - Abhä
    - glob
    - Vers
    - Abzv
- gepla
  - Front
    - OSE
    - OSEK ttOS

Vielen Dank für Ihre Aufmerksamkeit!