Implementierung eines kompilierenden Prozessorsimulators für die Architekturbeschreibungssprache TADL

Vortrag zum Diplom

Frank Sakowski
sakowski@ite.inf.tu-dresden.de

Technische Universität
Institut für Technische Informatik

Inhalt

- Vorgeschichte
- Motivation
- Stand der Technik
- Simulatorvergleich
- Simulatorumsetzung
- Ausblick

Vorgeschichte

- `TADL` – Architekturbeschreibungssprache
- `prosim` – interpretierender Prozessorsimulator
- `NDO` – Benutzeroberfläche für `prosim`
- `cheops` – Kommunikationsmodul für `prosim`

Motivation

- Prozessorsimulator – Test und Simulation von Architekturentwürfen
- kompilierender Simulator – Speicher- und Zeiteinsparung in der Ausführungsphase gegenüber interpretierenden Simulator

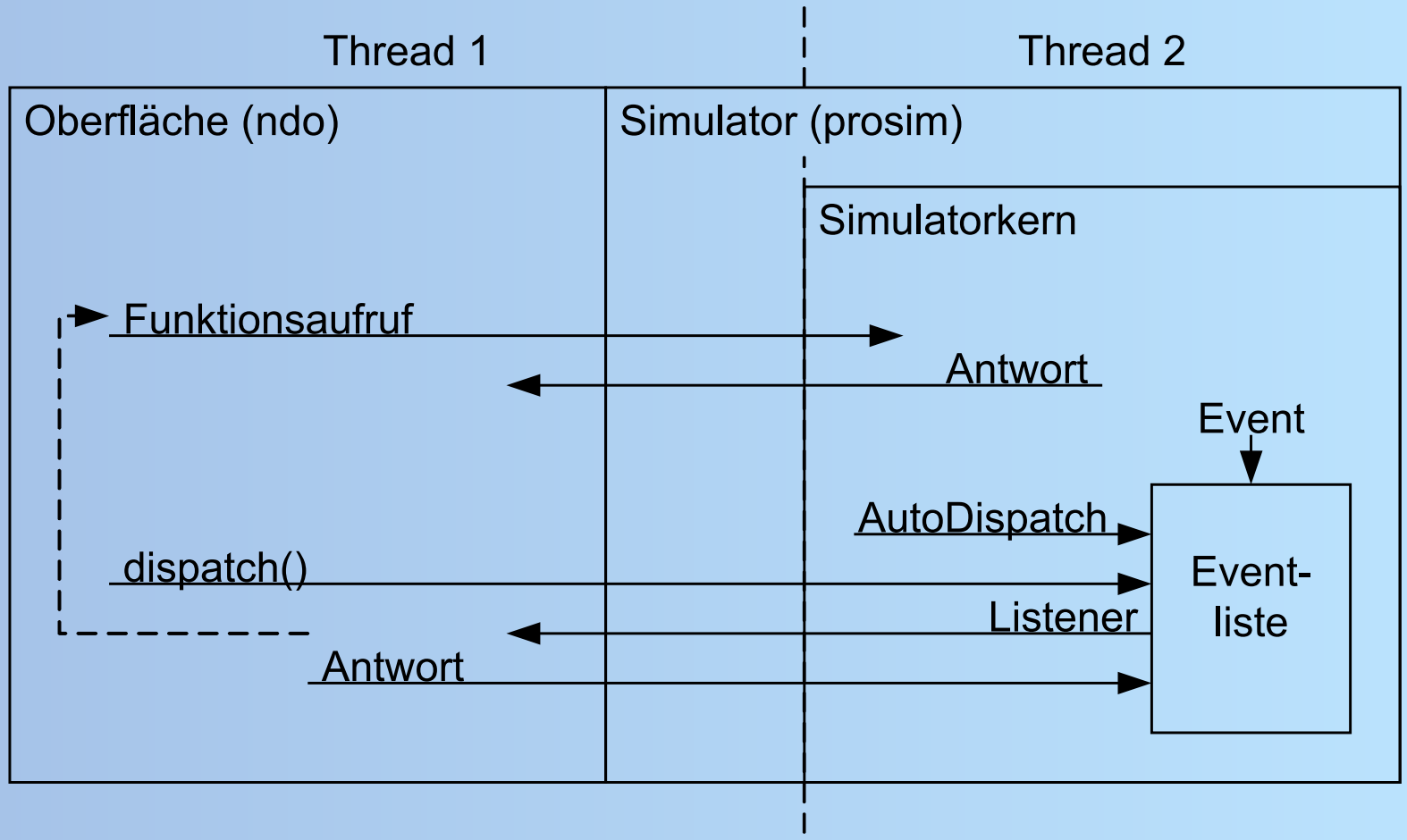
Stand der Technik

- in anderen Projekten werden kompilierende Simulatoren genutzt
- Trend zu noch stärkerer Zeiteinsparung durch Einbindung des Simulationsprogramms in den Kompilierungsvorgang – Vorverarbeitung des Programms, keine reine Architektur-simulation mehr
- nahezu jedes Projekt nutzt eigene Architekturbeschreibungs-sprache

Simulatorvergleich

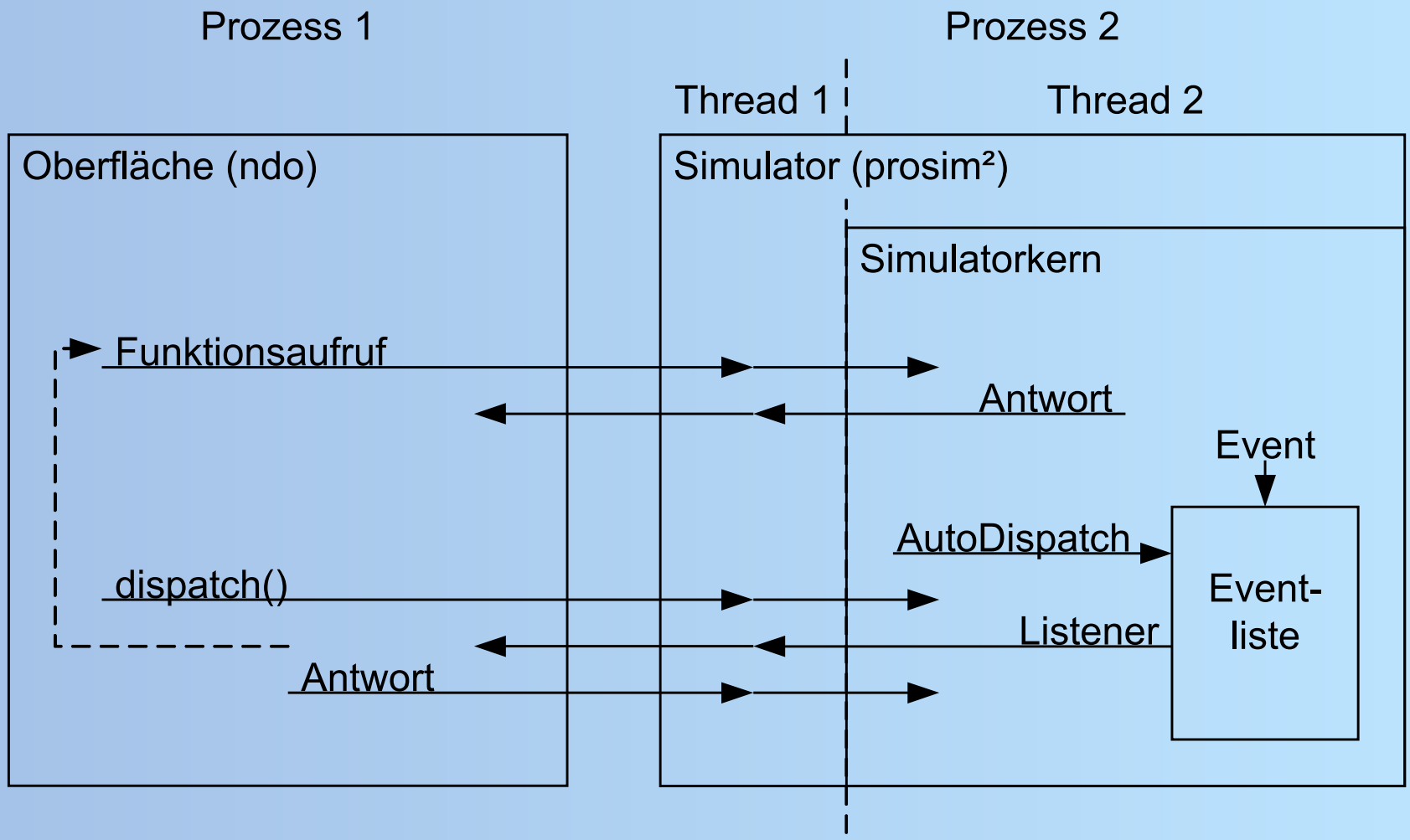
prosim	prosim^2
✦ interpretierend – Generator baut intern baumartige Operationsstruktur der Verhaltensbeschreibung auf	✦ kompilierend – Generator baut kompilierbares Modell der Verhaltensbeschreibung auf
✦ Simulatorkern ist vollständig und universell	✦ Simulatorkern wird mit Auswahl der Architektur generiert und ist spezifisch
✦ direkte Einbindung in NDO	✦ nur Simulatorgenerator direkt in NDO einbindbar

Kommunikation prosim



Frank Sakowski
Technische Universität
Institut für Technische Informatik
sakowski@ite.inf.tu-dresden.de

Kommunikation prosim²



Simulatorumsetzung

- Nutzung der vorhandenen Mechanismen aus `prosim` zur Übersetzung von `TADL`
- Implementierung des Simulators in `C++` und Wiederverwendung vorhandener Strukturen aus `prosim`
- `SystemC` als „Implementierungssprache“ unwarscheinlich, aber als Quelle hardwarenaher Datentypen möglich

Kommunikationsumsetzung

- Kommunikation zwischen den Prozessen mittels `cheops`
- `prosim`^{2} als Server – nahezu vorhanden
- `NDO` als Client – neu zu schreiben
- `UDE` als Client – vorhanden
- Verzicht auf Nutzung der Listener – Einwegekommunikation

Ausblick

- Entwurf und Realisierung des Simulatorgenerators
- Anpassung von `cheops` und `NDO`
- Durchführung von Geschwindigkeitsmessungen und Ermittlung des Speicherplatzbedarfs