Modellierung von parallelen und verteilten Systemen mit domänenspezifischen Sprachen

Andreas Breer, Gerrit Jan Veltink

Programmierung von *nebenläufigen* Systemen ist schwierig.
Programmierung von *parallelen* Systemen ist schwieriger.
Programmierung von *verteilten* Systemen ist noch schwieriger.

Gründe?
- mehr und komplexere Ausführungspfade
- erhebliche Lernkurve, z.B. bei Threads
- nicht (leicht) reproduzierbare Fehler

Lösungen?
- Entwickler länger und besser ausbilden?
- mehr Zeit für Fehlersuche einplanen?

Aber: *Multi-Core*, *GPGPU*, *Big Data*, ...
In der nahen Zukunft werden wir jedoch immer mehr parallele Programmierung brauchen!

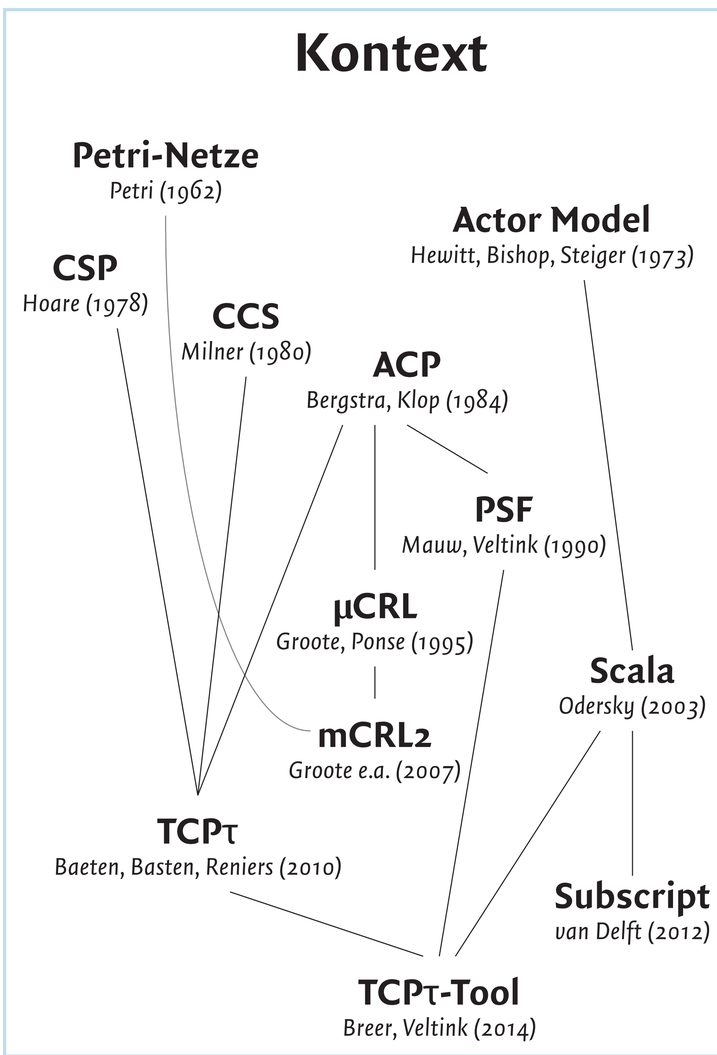
Wir suchen eine Lösung, die sich leicht integrieren lässt und von vielen Programmierern benutzt werden kann!

Alternative Lösungsvorschläge:
Formale Methoden für die Spezifikation von parallelen Systemen.

Werden aber häufig nicht benutzt, weil:
- sie einfach unbekannt sind
- Unterstützung durch Werkzeuge nicht ausreichend ist
- Unterstützung im Management fehlt, weil kein Code generiert wird

Ziel: Integration einer Prozessalgebra (TCPτ) als Domain Specific Language (DSL) in Scala.
- Spezifikation und Verifikation der parallelen Anteile via TCPτ (eingebettet)
- Implementierung und Werkzeugunterstützung via Scala

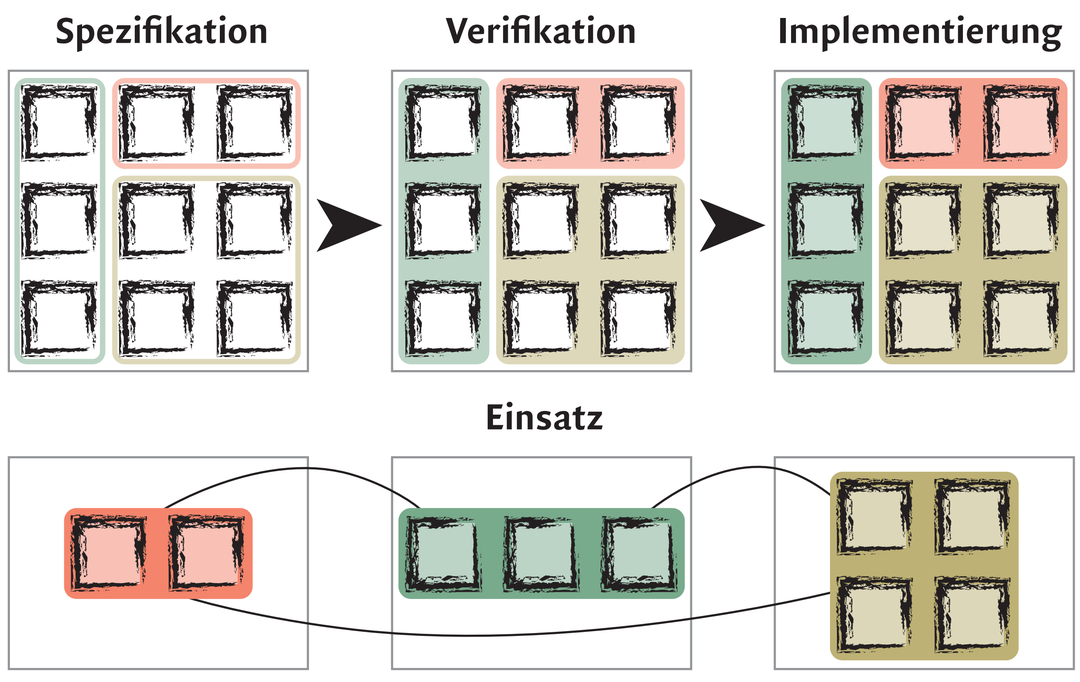
Kontext



Ausführbare Spezifikationen?

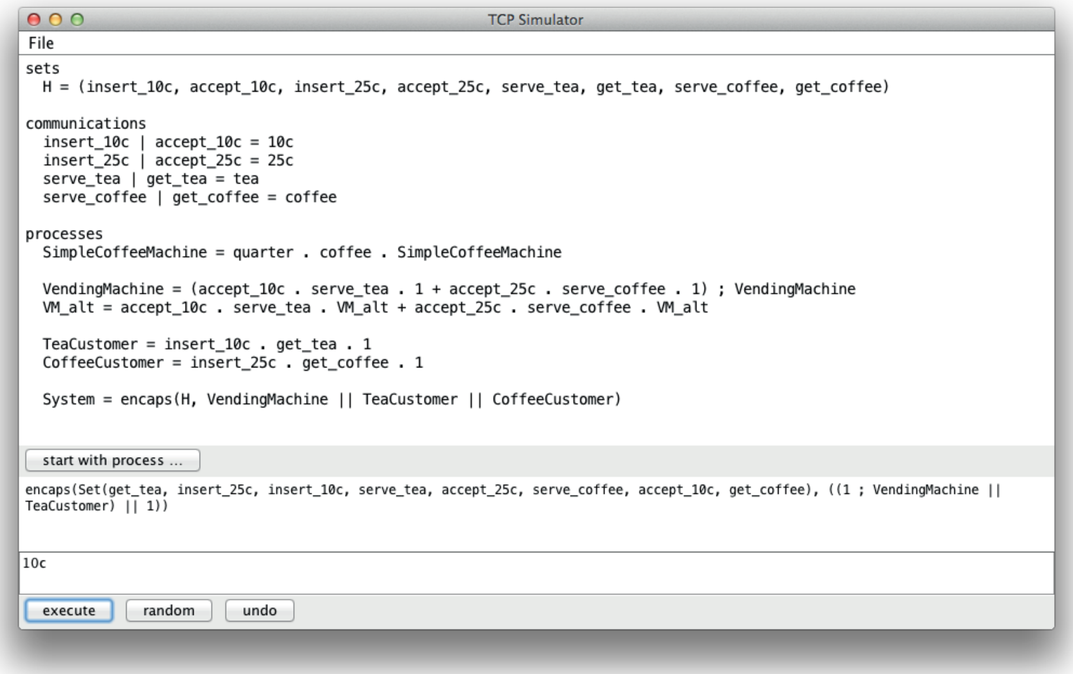
Was wäre, wenn man die parallelen Strukturen einer Anwendung aus einer ausführbaren Spezifikation ableiten könnte?

Ziel: Spezifikation und spätere Implementierung in einem Dokument definieren und mit der gleichen Entwicklungsumgebung bearbeiten.



Simulator

Ergebnis: Einfacher TCPτ-Simulator implementiert und integriert in Scala als externe DSL.



Das Beispiel zeigt eine einfache parallele Spezifikation eines Verkaufsautomaten und zwei Kunden.

Fazit

Scala eignet sich gut als Wirtssprache für die Entwicklung von DSLs.
Die Benutzung des Scala *Pattern Matching* ermöglicht einen übersichtlichen kompakten Quellcode, der die Axiome der Theorie noch deutlich im Code erkennen lässt.
Die feste Reihenfolge der Bindungsstärken von Scala-Operatoren verhindert leider eine „natürliche“ Notation der Prozessalgebra-Terme in Scala und somit die Implementierung einer *internen* DSL.

Ausblick

Eine Implementierung mittels einer *internen* DSL bleibt weiterhin wünschenswert, auch wenn die TCPτ-Syntax sich dann ändern würde. Scala *Macros*, könnten die Anzahl der Änderungen ggf. minimieren.
Angedachte Erweiterungen:
- Parametrisierung der Aktionen (Datentypen)
- weitere Werkzeuge: *Deadlock* und *Bisimulation Checker*
- Erprobung der Sprache und der Implementierung in Fallstudien

Andreas Breer
andreas.breer@technik-emden.de

Gerrit Jan Veltink
gert.veltink@hs-emden-leer.de

Hochschule Emden/Leer
Constantiaplatz 4
26723 Emden

University of Applied Sciences
HOCHSCHULE EMDEN·LEER