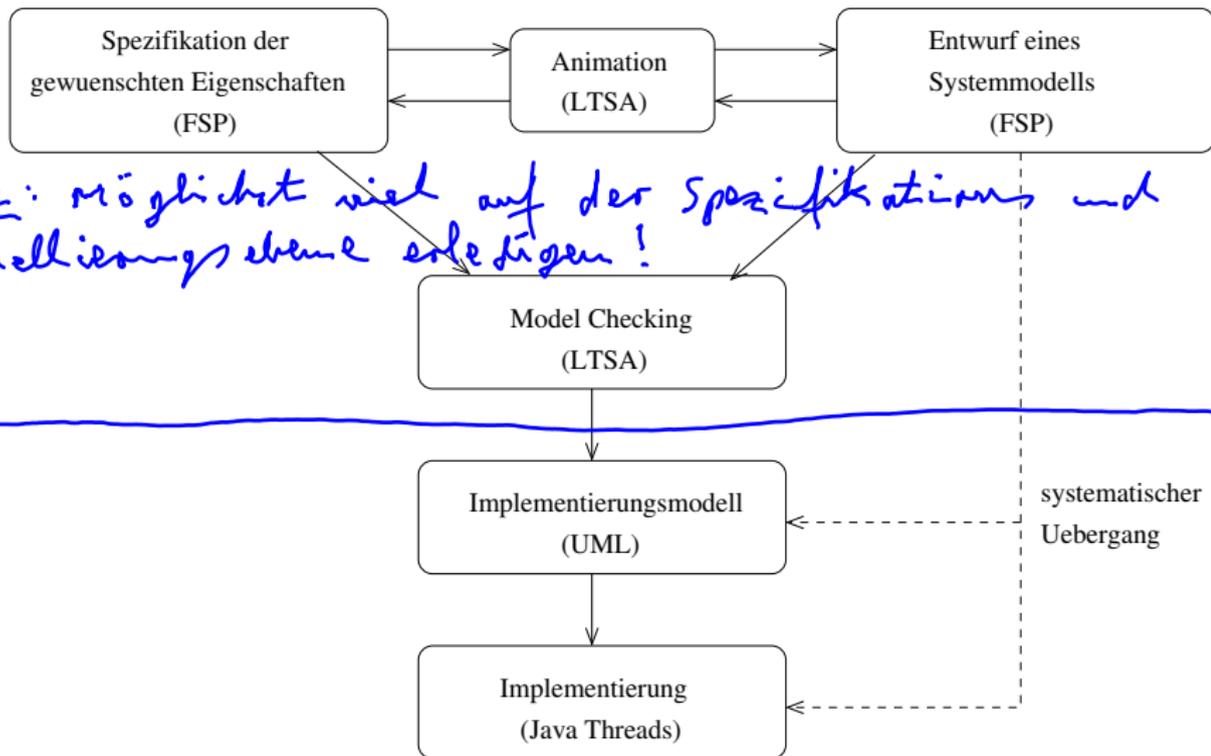


Kapitel 1

Einführung

Prof. Dr. Rolf Hennicker

27.04.2017



1.1 Begriffsbildung

Sequentielles Programm:

Anweisungen werden Schritt für Schritt hintereinander ausgeführt ("single thread of control").

a1; a2; a3; . . . → Zeit

Paralleles Programm (nebenläufig, concurrent):

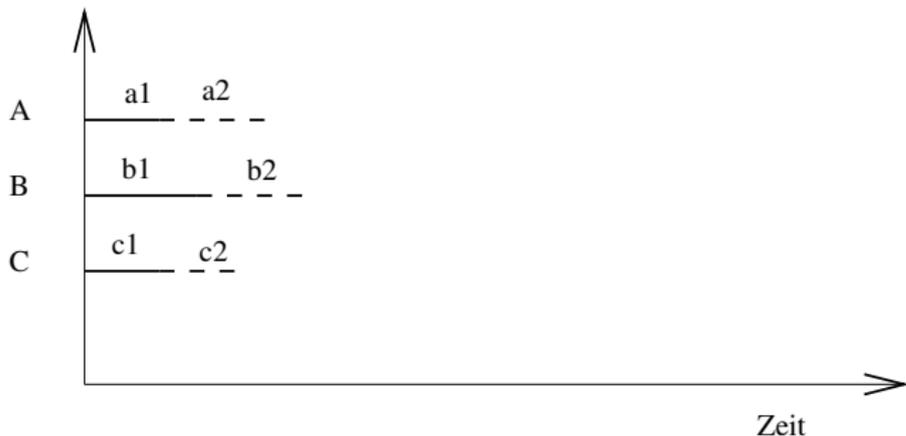
Anweisungen (von Teilen des Programms) werden nebeneinander ausgeführt ("multi threads of control").

Die Abarbeitung der Anweisungen kann entweder

1. echt gleichzeitig (echt parallel) oder
2. zeitlich verzahnt (quasi-parallel)

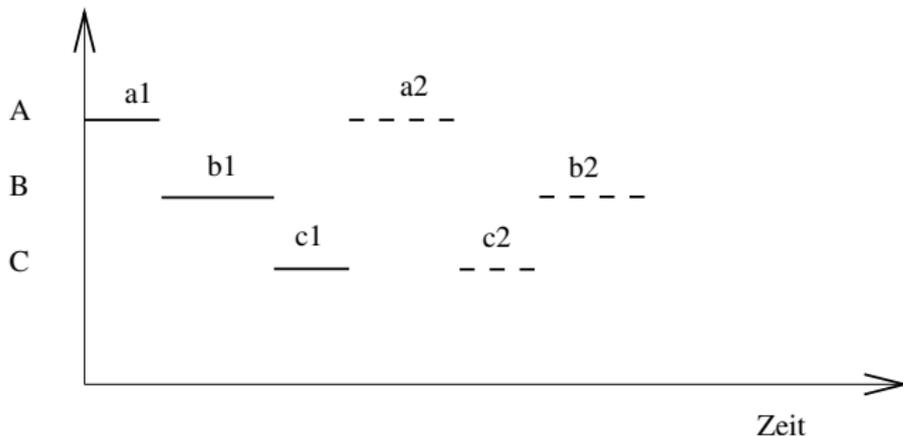
geschehen.

Echt gleichzeitige Abarbeitung



nur bei mehreren Prozessoren möglich

Verzahnte Abarbeitung



Die Reihenfolge wird hier vom Betriebssystem (Scheduler) zur Laufzeit festgelegt und ist vom Programmierer nicht vorhersehbar (Bsp. Mehrbenutzersystem mit 1 Prozessor).

Verteiltes System (“Distributed System“):

Verschiedene Teile des Software-Systems werden auf verschiedenen Rechnern ausgeführt, die miteinander vernetzt sind.

Bemerkung:

Phänomene, die nicht von Netzwerkeigenschaften und Netzwerkprotokollen abhängen, sind auch bei verteilten Systemen wie bei parallelen Programmen (die auf einem Rechner laufen) modellierbar.

1.2 Parallele Programme

Grundidee:

Komplexes System wird in Teile zerlegt, die nebeneinander (parallel) agieren.

Typische Phänomene:

- ▶ Prozess-Synchronisation
 - ▶ wechselseitiger Ausschluss
(z.B. Zugriff auf gemeinsame Betriebsmittel oder gemeinsame Variable)
 - ▶ Kooperation von Prozessen
(z.B. Erzeuger/Verbraucher)
 - ▶ synchrone/asynchrone Nachrichtenübertragung
(z.B. Client/Server)
- ▶ Nichtdeterministisches Verhalten
- ▶ Deadlock bzw. Deadlockfreiheit
- ▶ Sicherheit und Lebendigkeit ("Safety & Liveness")
(Lebendigkeit: z.B. Fortschritt, Fairness)

Vorteile:

- ▶ Adäquate Kontrollstruktur von Programmen bei Anwendungen mit inhärenter Parallelität (z.B. bei eingebetteten Echtzeitsystemen oder Steuerung von Fertigungsprozessen)
- ▶ Besserer Durchsatz (z.B. bei Benutzerinteraktionen)
- ▶ Gewinn von Performanz
- ▶ Multitasking

Probleme:

- ▶ Komplexität paralleler Systeme
- ▶ Abläufe paralleler Aktivitäten häufig schwer zu durchschauen

vieler kombinatorischer Möglichkeiten

Konsequenz:

Fehlerhafte Programme

Benötigt:

Wohlfundierte Methoden und Techniken zum *Entwurf* und zur *Implementierung* paralleler Programme und zur Verifikation von Eigenschaften.

Wir verwenden dazu:

- ▶ im Entwurf:
 - ▶ *Modelle*, d.h. vereinfachte und abstrakte Darstellungen der parallelen Prozesse
 - ▶ *Beschreibungstechniken*:
graphisch: endliche Zustandsmaschinen
textuell: Sprache FSP ("finite state processes")
 - ▶ *Techniken* zur Analyse und zum Nachweis von Sicherheits- und Lebendigkeitseigenschaften (Model Checking)
- ▶ in der Implementierung:
 - ▶ UML: Modellierung der Implementierung
 - ▶ Java: Codierung durch Java Threads, synchronized methods, wait, notify.

Der Übergang

Entwurfsmodell \rightsquigarrow Java-Programm

wird systematisch durchgeführt.