Diskrete Simulation (Masterkurs)

Überblick über diskrete Simulationssysteme

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem@informatik.htw-dresden.de

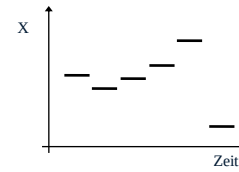
HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Diskrete Simulationssysteme

- Software für die diskrete Simulation
- Historische Entwicklung
- GPSS als bekanntester Vertreter älterer Systeme
- SLX als moderne Simulationssprache
- Bausteinsysteme

Anforderungen an diskrete Simulationssysteme

- Simulationssystem muß algorithmische Beschreibung realer oder geplanter Prozesse mit diskreten Zeitverhalten (ereignisorientiert) erlauben



Schwerpunkte

- Abbildung von parallelen Aktionen und Wechselwirkungen einer großen Menge von Objekten
  - schnelle Umschaltung zwischen den einzelnen Prozessen
  - Effizientes Speichermanagement zur Verwaltung großer Objektmengen
- Modellierung von stochastischen Prozessen
  - Bereitstellung von Zufallszahlengeneratoren
  - Unterstützung der statistische Auswertung der Ergebnisse
- Animation und Visualisierung
  - einfache 2D-Animationen mit Statusanzeigen
  - aufwendige 3D-Animationen

Historische Entwicklung diskreter Simulationsysteme

- erste Simulationen diskreter Systeme mit universellen Programmiersprachen
- Basissprache der ersten Simulatoren war meist FORTRAN.
- FORTRAN war für damalige Verhältnisse sehr effizient und sehr gut für numerische Berechnungen in Wissenschaft und Technik geeignet
- Sehr gute Unterstützung der Zufallszahlengenerierung durch umfangreiche Mathematikpakete (Nachnutzung von Bibliotheken der kontinuierlichen Simulation)

Genereller Aufwand zur Implementierung war dabei jedoch noch höher als bei der kontinuierlichen Simulation

- keine leistungsfähige Funktionen zur Verwaltung dynamischer Objekte,
- stark begrenzte Massenspeicher
- Geringe Rechengeschwindigkeiten stark begrenzt
- Hauptproblem Koroutinenfähigkeit - dadurch Standardprogrammiersprachen nicht direkt einsetzbar
- Lösung durch spezielle, mit Assemblerunterprogrammen erweiterte Subroutinen
- Die Subroutinensammlung bildeten sehr rasch die syntaktische und semantische Basis für spezialisierte Simulationssprachen (siehe Folgeabschnitt).

Die Simulationssprache GPSS

- Die Simulationssprache **GPSS** (General Purpose Simulation System) gehört zu den ersten und bekanntesten Sprachen für die diskrete Simulation.

Hauptmerkmale :

- IBM als (damals) führender EDV-Hersteller setzte bereits 1961 damit den Maßstab
- auch heute noch relativ guten Funktionalität, Flexibilität und Leistungsfähigkeit
- sehr hohe Performance erlaubt auch heute noch sehr große Modelle im Bereich von Bedien- und Warteschlangensystemen
- blockorientierte Sprache :
  - bewegliche Elemente laufen durch ein Netz von einzelnen Blöcken
  - heutige Versionen von GPSS enthalten bis zu 50 verschiedene Blöcke
  - Im Prinzip entspricht jedem Block ein fest programmiertes Unterprogramm.
  - Die Parameter jedes Blockes können damit als Übergabeparameter für das Unterprogramm gesehen werden.
  - mit Gruppen von Blöcken können die statischen Objekte des Modelles nachgebildet werden. Die Grundidee der Blöcke stammt dabei von den anfänglichen Blockdiagrammen ab, mit welcher die Strukturen der Modelle dargestellt wurden.

Sprachkomponenten von GPSS

- Die beweglichen Elemente in GPSS werden als Aktivatoren (engl. Transactions) bezeichnet. Programmtechnisch wird jeder Aktivator als ein Vektor von Werten dargestellt. Neben nutzer- und modellspezfischen Aktivatorparametern enthält jeder Aktivator auch Informationen zur Programmposition und zum Erzeugungs- und Austrittszeitpunkt.
- Die Aktivatoren bewegen sich durch die folgenden statischen Elemente, welche vor dem Start der Simulation definiert werden müssen:
  - **Generator (Generate) -**Generierungsblock, welcher nach einem vorgegebenen Zeitschema Aktivatoren generiert
  - **Einrichtung (Facility) -** Bedienungs- und Bearbeitungsstation, meist mit der Kapazität 1, wenn eine Einrichtung durch einen Aktivator belegt ist, werden keine weiteren Aktivatoren eingelassen
  - **Speicher (Storage) -** Speicherelement zur Aufnahme einer bestimmten Anzahl von Aktivatoren, gut geeignet zur Nachbildung von Lagerplätzen
  - **Warteschlange Queue) -** Warteschlange mit unendlicher Kapazität
  - **Test (Gate) -** Testblock zur Abprüfung von Bedingungen und zur bedingten Verzweigung oder Aufsplittung von Aktivatoren
  - **Terminator -** Block zur Vernichtung (Auskopplung) von Aktivatoren

Grundbefehle von GPSS

Den einzelnen Komponenten sind jeweils eine Reihe von Blöcken zugeordnet:

- **Generator**

  GENERATE A,B,C,D

  mit A,B-Generierungsabstand der Form A+/-B gleichverteilt, C-erster Zeitpunkt, D-maximale Anzahl, (B,C,D sind optional)

  Bsp.: GENERATE 100,10,50,40

  -> erzeugt ab Zeitpunkt 50 mit Abstand 100+/-10 Zeiteinheiten insgesamt 40 Aktivatoren

- **Einrichtung (in der Regl mit einer Kapazität = 1)**

  SEIZE ***Einrichtungsname*** versucht die Einrichtung zu betreten

  ADVANCE 10,2 verzögert zufällig um 10+/- Zeiteinheiten

  RELEASE ***Einrichtungsname*** verläßt die Einrichtung

- **Speicher**

  INITIAL SK=*Speichername,B* legt Speichergröße auf B Einheiten fest

  ENTER *Speichername* Belegung des Speichers mit einer Einheit

  ADVANCE 10 Verzögerung (Lagerzeit)

  LEAVE *Speichername* Aktivator verläßt den Speicher

Grundbefehle von GPSS II

Warteschlange mit unbegrenzter Kapazität

QUEUE *Warteschlangenname* Betreten einer Warteschlange

DEPART *Warteschlangenname* Verlassen der Warteschlange

Tests auf logische Bedingungen zum Verzweigen von Aktivatoren

TRANSFER A,B,C A -Wahrscheinlichkeit p,B-Verzweigung mit 1-p nach Blocknr. B , C- Sprungziel mit Wkt. p

Bsp.: TRANSFER 0.3,M1,M2 Verzweigung mit 30% Wkt. nach Marke M2 und 70% nach M1

LOGIC *Schaltername* setzt einen Schalter (R -aus S-ein I-invers)

GATE *Schaltername* prüft den Schalter und stoppt bei False

- Bsp.: LOGIC R, FLAG1
  GATE LR FLAG1

Terminator dient zum Freigeben (Vernichten von Aktivatoren)

- TERMINATE A vernichtet den eintreffenden Aktivator und verringert optional den Startzähler um A

GPSS-Befehle zur Simulationssteuerung

Block	Einsatz
SIMULATE	Startet die Übersetzung des GPSS-Quellprogramms
START A	Startet die Simulation und setzt den Startzähler auf A, Kombination mit Terminate erfolgt eine definierte Beendigung der Simulation
RESET	setzt alle statistischen Werte zurück, dient zur Unterdrückung der Einschwingvorgänge
REPORT	leitet eine Reihe von Anweisungen zur Druckausgabe ein
END	beendet das Programm

Typische Steuersequenz zur Unterdrückung von Einschwingprozessen

`SIMULATE`	Beginn des Steuerblocks
`START 100`	Startet Simulation mit 100 Aktivatoren (Einschwingen)
`RESET`	setzt die Statistiken zurück
`START 1000`	Startet eigentliche Simulation mit 1000 Aktivatoren
`END`	Ende des gesamten Simulationsprogramms

GPSS-Aktivatoren

GPSS-Aktivatoren können als ein Datenfeld aufgefaßt werden, dessen Attribute zur Modellierung der dynamischen Objekteigenschaften dienen:

Datenfelder :

- Laufende Nummer des Aktivators (zur eindeutigen Referenzierung)
- Erzeugungszeitpunkt (für statistische Zwecke)
- Prioritätsklasse für prioritätsabhängige Verzeigungen
- Nummer des aktuellen belegten Blockes
- Blockabgangszeit (geplanter Ereigniszeitpunkt)
- Freie Parameter für beliebige Anwendungen, unterteilt in verschiedene Datentypen (byte, integer, double), die Anzahl der Parameter kann durch den Anwender festgelegt werden
  - Mit ASSIGN kann der Inhalt modifiziert und durch Testbefehle kann der Inhalt jedes Aktivators getestet und wertabhängig verzweigt werden:

    ASSIGN 2+, 10, PH - addiert 10 zum zweiten Parameter

    TEST GE PH2, 100, AUSSCHUSS // verzeigt bei Par1>100 nach Ausschuss

Ein GPSS-Beispielprogramm

```
START   GENERATE  20        Erzeugung von Aktivatoren aller 20 Zeiteinheitem
        QUEUE     WS1       Eintritt in die Warteschlange vor Maschine 1
        SEIZE     MASCH1    Versuch auch Maschine 1 zu betreten
        DEPART    WS1       bei Erfolg wird Warteschlange verlassen
        ADVANCE   18,5      Verzögerung um 18+/- 5 ZE (Bearbeitung)
        QUEUE     WS2       Eintritt in die Warteschlange vor Maschine 2
        RELEASE   MASCH1    Maschine 1 verlassen
        SEIZE     MASCH2    Versuch Maschine 2 zu betreten
        DEPART    WS2       bei Erfolg wird Warteschlange 2 verlassen
        ADVANCE   19,5      Verzögerung um 19+/- 5 Zeiteinheiten
        RELEASE MASCH2      Maschine 2 verlassen
        TERMINATE 1         Aktivator vernichten, Startzähler um 1 verringern
```

- Hinweis: Eine Besonderheit stellen die verschachtelten Aus- und Eintritts-anweisungen zur Modellierung von Blockierungen dar.

Quelltextbasierte, diskrete Simulationssysteme

- GPSS war lange Zeit der Quasistandard und hat die Entwicklung anderer diskreter Simulationssysteme stark beeinflusst,
- Ältere GPSS-Simulationssysteme sind mit einigen Nachteilen behaftet :
  - Quelltext muß teilweise noch im FORTRAN-Format an festen Positionen notiert werden
  - Rechenoperationen und Syntax unterscheiden sich stark von üblichen prozeduralen Sprachen (vgl. ASSIGN und TEST – Beispiele bei Aktivatorstruktur)
  - Probleme bei der Integration in moderne IT-Infrastrukturen (keine Datenbankschnittstellen, reine Kommandozeilenversionen
- Einziger großer Vorteil: durch die starke Optimierung der Laufzeitperformance sehr schnell im Vergleich zu heutigen Simulationssystemen (Faktor 10 bis 100)
- Dieser Performancevorteil wurde von einem der maßgeblichen Entwickler von GPSS, Jim Hendriksen in einer neuer Simulationssprache SLX auf eine zeitgemäßes Softwareniveau gebracht
- SLX kann auch ältere GPSS-Programme ausführen, ist aber sonst eine eigenständige und relativ moderne Programmiersprache
- Durch den zeitgemäßen Aufbau und das weite Spektrum an Funktionen, auch bis hin zur Ausführung von GPSS-Programmen, wird SLX in den Übungen verwendet.

Das Simulationssystem SLX

SLX im Überblick

- SLX steht für **S**imulation **L**anguage with e**x**tensibilities
- Hersteller ist die Wolverine Software Corporation USA
- von der gleichen Firma stammen auch das leistungsfähigste GPSS-System GPSS/H und das Animationssystem PROOF
- alle Grundkonzepte der sehr effizienten Ereigniskalenderverwaltung von GPSS/H wurden übernommen
- weiterhin beruht SLX auch auf Konzepten aus SIMULA, C und C++
- SLX ist objektbasiert, aber nicht vollständig objektorientiert

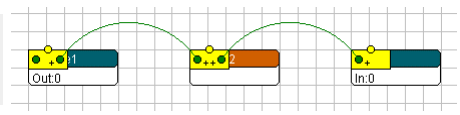
Grundaufbau der Sprache

- alle nicht simulationsbezogenen Funktionen entsprechen C !
- Simulationsbezogene Funktionen verwenden sogenannte Pucks mit frei definierbarer Datenstruktur als dynamische Objekte
- größere Quelltexte können in Module gegliedert werden

=> detaillierte Vorstellung in nachfolgender Vorlesung

Bausteinbasierte Modellierungskonzepte

- Mit der Verfügbarkeit echter 2D-Grafik-Benutzerschnittstellen seit Anfang der 90er Jahre und immer leistungsfähigerer Grafikkarten entstanden verschiedene grafisch-unterstützte Simulationssystem, welche Simulationsbausteine durch grafische Symbole visualisieren



Grundprinzipien

- Jeder Baustein verfügt über eine vordefiniertes (anfangs häufig fest definiertes) Verhalten, welches über einige Parameter genauer bestimmt werden kann.
- Parameterdefinition meist über über Tabellen oder bausteinspezifische Masken ebenfalls über grafisch-interaktive Menüs
- Grafikoberfläche ist gleichzeitig auch Animationsmedium (zu Beginn 2D, heute auch 3D-Animation)
- Simulationsdurchführung über Generierung von Programmcode für ältere Simulationssprachen oder direkte Interpretation der Bausteinparameter

Stärken und Schwächen von Bausteinsystemen

- Modellbausteine erlauben Komposition von Simulationsmodellen ohne oder mit nur geringem Programmieraufwand
- auch die Datenaufbereitung und Simulationsauswertung werden sehr gut unterstützt und laufen meist ohne größeren Anpassungsaufwand
- Modelle mit einem geringen bis mittleren Komplexitätsgrad lassen sich in der Regel einfach modellieren.
- Bei sehr spezifischen Problemen stoßen bausteinbasierte Systeme rasch an Grenzen. Die Lösbarkeit hängt dann vom Grad der Anpassbarkeit der Bausteine ab.
- Nach anfänglichen Systemen mit nicht modifizierbaren Bausteinen verfügen aktuelle Systeme immer über eine Option zur Anpassung der Bausteine :
  - möglich sind Pulldownlisten mit Parametervorgaben, spezielle Skriptsprachen, Einschuboptionen für universelle Programmiersprachen (z.B. C++ oder VBA), oder die Erzeugung neuer Bausteine durch Anwendung objektorientierter Vererbungstechniken
  - der Umfang möglicher Bausteinmodifikationen sollte unter dem Gesichtspunkt des Anwendungsfeldes als eines der wichtigsten Auswahlkriterien für diskrete Simulationssysteme berücksichtigt werden

Die Arten von Simulationssystemen im Vergleich

Gegenwärtig existiert kein in allen Belangen optimales System