Technische Universität München
Informatik VIII
Netzarchitekturen und Netzdienste
Prof. Dr.-Ing. Georg Carle

Alexander Klein, Nils Kammenhuber

Übungen zur Vorlesung Diskrete Simulation

Übungsblatt 3, SS 2010

Abgabe: 30. Juni 2010 (bis 14 Uhr am Lehrstuhl I-8, Finger 03.05 oder in der Vorlesung)
Die Aufgaben können jeweils von zwei Studenten gemeinsam bearbeitet werden.

In der letzten Übung haben Sie einen einfachen Eventsimulator für ein $GI/GI/1/\infty$-Wartesystem implementiert. Der in der Aufgabenstellung vorgegebene Umgang mit Zufallszahlen lässt jedoch aus einer Reihe von Gründen sehr zu wünschen übrig. Diese Gründe waren insbesondere:

1. Es können nur gleichverteilte Zufallszahlen erzeugt werden.
2. Die erzeugten Zufallszahlen sind vom Typ `double`. Aus diesem Grund wurde `double` auch als Datentyp für die Simulationszeit gewählt. Allerdings ist die Auflösung von `double` variabel: sie nimmt in Richtung $+\infty$ logarithmisch ab. Beispielsweise ist `1.0e+18 + 50.0 == 1.0e+18`, wenn man mit `double` rechnet.
3. Der verwendete Zufallszahlengenerator aus `java.util.Random` ist zwar schnell, hat aber keine sonderlich günstigen statistischen Eigenschaften: es ist ein LCG.

Diese drei Hauptkritikpunkte sollen auf diesem Übungsblatt nun ausgemerzt werden.

Aufgabe 1 — Integer-Simulationszeit (30 Punkte)

Ziele: *Wie man sauber mit Simulationszeit umgeht; Vertiefung des einfachen Simulations-Konzeptes.*

Im Code für den Simulator soll aus den o. g. Gründen das Format für die Simulationszeit von `double` auf `long` umgestellt werden. Hierbei geht man sinnvollerweise folgendermaßen vor:

- Man überlegt sich eine *Auflösung* für die Zeit. Das kleinste darstellbare Zeitintervall in dieser Auflösung (sozusagen ein „Simulationszeit-Atom“) nennt man *Clock Tick* oder kurz *Tick*.
- In der Modellbeschreibung (d. h. in der Konfiguration des Simulators) verwendet man stattdessen nach wie vor geläufigere Zeiteinheiten, z. B. Sekunden. Diese müssen also beim Einlesen der Konfiguration in Ticks umgerechnet werden; umgekehrt müssen die Ticks bei der Berechnung von Statistiken oder sonstigen Ausgaben in Sekunden zurückgerechnet werden.
- Darüberhinaus enthält die Modellbeschreibung meistens Angaben über Zufallsverteilungen. Auch diese beziehen sich auf Sekunden, nicht auf Ticks. Meist geht man so vor, dass man während der Simulation mit Hilfe der Zufallsverteilungen Angaben in Sekunden berechnen lässt, welche man dann on-the-fly in Ticks umrechnet.

a) Stellen Sie nun Ihren Simulatorcode von `double`-Sekunden intern auf `long`-Clockticks um. Gehen Sie dabei folgendermaßen vor: Setzen Sie die Frequenz fix auf $1.000.000\,\frac{\text{Ticks}}{\text{s}}$; erlauben Sie optional die Angabe einer alternativen Frequenz (z. B. durch einen entsprechenden Kommandozeilenparameter oder Angabe in einer Konfigurationsdatei). Schreiben Sie zwei kleine Umrechnungsroutinen `ticksToSeconds` und `secondsToTicks`. Stellen Sie die Event-Engine auf `long` um und setzen

Sie die Umrechnungsfunktionen an den entsprechenden Stellen ein.

b) Was ist der Nachteil einer hohen Frequenz?

Aufgabe 2 — Erzeugen anderer Verteilungen (50 Punkte)

Ziele: *Besseres Kennenlernen der Inversionsmethode, musterbasiertes Software-Engineering*

In der Vorlesung haben Sie die Inversionsmethode kennengelernt. Mit ihrer Hilfe lassen sich beliebige Zufallsverteilungen aus einer U(0,1)-Verteilung ableiten; sie wird daher in sehr vielen Programmen angewendet. Der Simulatorcode soll nun entsprechend abgeändert werden, damit er auch die Verwendung anderer Verteilungen ermöglicht.

a) Wir wollen nun das sog. „Strategy Pattern“ aus dem musterbasierten Software-Engineering anwenden (Software Patterns):[1] Erstellen Sie zunächst ein Interface `RandomDistribution`, welches die Funktion `nextDouble()` definiert (analog zu `java.util.Random`). Erstellen Sie dann eine Klasse `UniformDistribution`, welche das Interface `RandomDistribution` implementiert. Dem Konstruktor sollte man Parameter für Minimum und Maximum mitgeben können. Als Quelle für die Zufallszahlen dürfen Sie – trotz aller Bedenken – zwecks Zeitersparnis weiterhin `java.util.Random` verwenden.
(Erklärung: Die Zufallsverteilung ist unsere „Strategie“ im Strategy-Pattern, welche wir dynamisch, also zur Laufzeit, bestimmen können wollen.)

b) Schreiben Sie den Simulator so um, dass er die Zufallszahlen nicht mehr aus `java.util.Random` zieht, sondern aus einem `RandomDistribution`-Objekt. Bedenken Sie, dass Sie zwei getrennte `RandomDistribution`-Objekte benötigen: eines für die Zwischenankunftszeiten, eines für die Bedienzeiten.
Erweitern Sie den Simulator so, dass man (als Kommandozeilenparameter oder per Konfigurationsdatei) für Zwischenankunftszeit und Bedienzeit jeweils eigene Parameter für die Zufallsverteilung angeben kann, und lassen Sie entsprechend parametrisierte `UniformDistribution`-Objekte zur Laufzeit (d. h., zu Beginn der Simulation) erzeugen.

c) Schreiben Sie mit Hilfe der Inversionsmethode eine Klasse `ExponentialDistribution`. Dem Konstruktor soll man als Parameter die gewünschte Rate (λ) übergeben können.
Erweitern Sie den Simulator so, dass man (als Kommandozeilenparameter oder per Konfigurationsdatei) für Zwischenankunftszeit und Bedienzeit die gewünschte Verteilung samt ihrer Parameter angeben kann.

d) Wiederholen Sie zur Überprüfung einige der Beispiele aus der Vorlesung, und vergleichen Sie.

Aufgabe 3 — Verwendung eines besseren Zufallszahlengenerators (20 Punkte)

Ziele: *Gedanken über die Qualität von Zufallszahlengeneratoren*

a) Bislang haben wir aus Bequemlichkeit `java.util.Random` als Zufallszahlengenerator verwendet. Überlegen Sie, ob man seine Verwendung in unserem Fall verantworten kann oder nicht. Machen Sie ggf. eine kurze Recherche im Netz, wie er parametrisiert ist und welche Qualität er hat.

b) Wir wollen nun den Mersenne Twister verwenden. Verwenden Sie hierzu einfach den Code von `http://www.cs.gmu.edu/~sean/research/` und stellen Sie Ihre Verteilungsklassen entsprechend um.

c) Zusatzfrage: Wie hätte man hier das Strategy-Pattern anwenden können, um die Methode der Zufallszahlengenerierung ebenfalls variabel zu gestalten?

[1] Eine gute Beschreibung gibt `http://en.wikipedia.org/w/index.php?title=Strategy_pattern&oldid=369021599`