Discrete simulatie

D-95-6B
R.S. Sukdeo
Leidschendam, 1995
Stichting Wetenschappelijk Onderzoek Verkeersveiligheid SWOV

Stichting Wetenschappelijk Onderzoek Verkeersveiligheid SWOV

Stichting
Wetenschappelijk
Onderzoek
Verkeersveiligheid
SWOV

Postbus 1090
2260 BB Leidschendam
Duindoorn 32
telefoon 070-3209323
telefax 070-3201261

Inhoud

1. Inleiding

In de praktijk kom je problemen tegen die moeten worden opgelost. Je wilt bijvoorbeeld weten of het zinvol is een bepaalde investering te doen, of je wilt de invloed van een bepaalde beslissing in een technisch systeem of in het beleid van een bedrijf bepalen. Het uitvoeren van experimenten met het systeem of het beleid kan dan tot een oplossing van het probleem leiden. Soms is het niet mogelijk om met het werkelijke systeem te experimenteren, en soms is het niet erg praktisch, zoals het onderzoeken van de invloed van een bepaalde meststof op de opbrengst van een gewas. De periode waarin de mogelijke effecten merkbaar worden, zal lang zijn. Een andere reden om in dit geval niet te experimenteren met het echte systeem is het feit dat de groei van de plant afhankelijk is van omstandigheden die niet altijd gelijk zullen zijn, zoals het weer, zonlicht en regen. In zo'n geval zal men experimenten uitvoeren met behulp van een model van het werkelijke systeem. Dit wordt simulatie genoemd.

Het woord simulatie komt van het Latijnse woord *simulare*, dit betekent gelijkmaken, veinzen of nabootsen. De laatste betekenis geldt bij simulatie.
Volgens Shannon (1975) is simulatie:

> "Het vervaardigen van een model van een werkelijk systeem en experimenteren met dit model met als doel:
> 1. Het gedrag van het systeem te leren begrijpen.
> 2. Het ontwikkelen van een strategie voor het optimaal werken met het systeem.
> 3. Training, van bijvoorbeeld bedieningspersoneel."

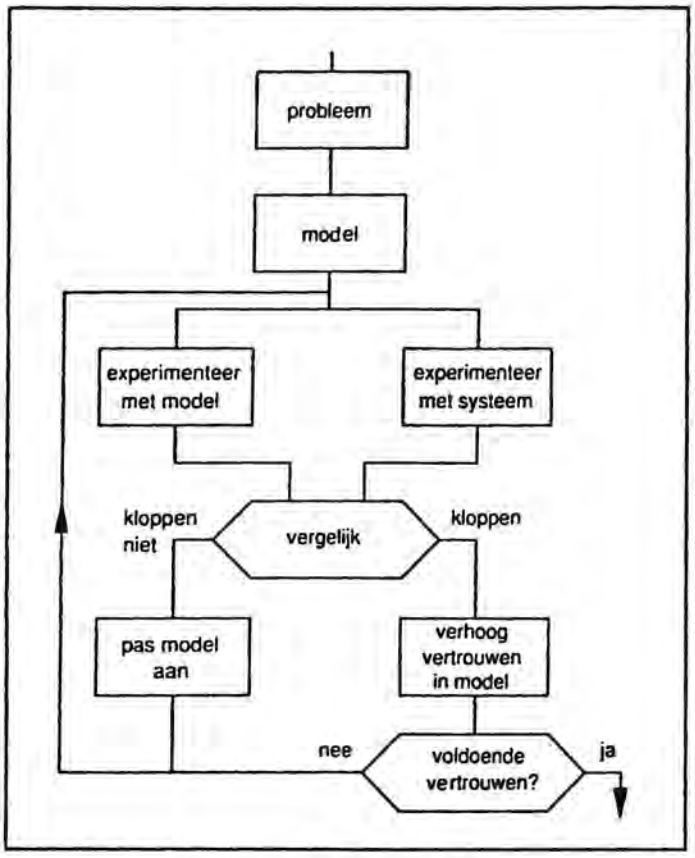
Een systeem is een verzameling elementen of componenten, met een onderlinge relatie, zodat een bepaald doel bereikt kan worden. Het is niet nodig dat alle elementen met alle andere elementen een relatie hebben. Als een van de elementen wordt weggehaald uit het systeem dan zal het overblijvende systeem niet meer op dezelfde manier functioneren.
Voorbeelden van systemen:

Systeem	Componenten	Relaties
produktiehal	werknemers, machines	de werknemers bedienen de machines
tankstation	auto's, pompen, wachtrij	de auto's staan in een wachtrij, de auto's tanken aan een pomp
supermarkt	klanten, kassa's, kassières	de klanten worden door de kassières aan de kassa geholpen

Bij het bestuderen van het systeem is het van belang dat de grenzen van een systeem worden vastgelegd. Die grenzen kunnen meestal zelf gekozen worden, naar het beste inzicht van de onderzoeker, rekening houdend met het doel van de studie.

Het bestaan van een systeem is niet altijd noodzakelijk. Bij het bestuderen van een systeem in ontwerp kan het onderzoek slechts worden gedaan aan de hand van een model. Een model is meestal een vereenvoudigde voorstelling van een bestaand of niet bestaand systeem. Een model in het algemeen zal dan ook niet in staat zijn het gedrag van een systeem onder alle omstandigheden nauwkeurig te beschrijven. Een model kan gedefinieerd worden als een verzameling instructies die, zo goed mogelijk, het gedrag van een werkelijk systeem nabootst.

De modelcyclus:



Bij computermodellen wordt een wiskundige of andere beschrijving van het systeem gemaakt en dit wordt vertaald naar een computerprogramma. Het computerprogramma vervangt het werkelijke systeem. Door middel van het computerprogramma worden de toestandsveranderingen van het gemodelleerde systeem beschreven. Experimenteren met een computerprogramma dat het gedrag van een systeem beschrijft, wordt computersimulatie genoemd. Omdat tegenwoordig vrijwel alle simulatiestudies uitgevoerd worden met behulp van computerprogramma's, wordt de toevoeging computer meestal weggelaten.

Computersimulaties worden in een groot aantal disciplines toegepast, zoals in de lucht- en ruimtevaart en bij het testen van een vliegtuigmodel in een windtunnel. In de bedrijfskunde wordt simulatie toegepast bij onder meer, wachttijdproblemen, produktieprocessen, goederenstromen, voorraadproblemen en investeringsproblemen.

Er zijn een aantal redenen, waarom er wordt geëxperimenteerd met een model in plaats van met het werkelijk systeem. Experimenteren met een werkelijk systeem is:

- Te kostbaar; om te onderzoeken of een nieuw metrosysteem het verkeersprobleem kan oplossen, zal men in het algemeen geen nieuw metrosysteem gaan bouwen.
- Te gevaarlijk; bijvoorbeeld bij een chemisch proces.
- Niet ethisch; het experimenteren met mensen ervaart men in het algemeen als niet-ethisch.
- Onmogelijk; als het systeem in werkelijkheid niet bestaat dan kan het niet anders.
- Training; bijvoorbeeld bij het opleiden van piloten.
- Herhaalbaarheid.

2. Toevalsgetallen

Toevalsgetallen kunnen op veel manieren verkregen worden, bijvoorbeeld door het werpen van een dobbelsteen. Voor simulatietoepassingen hebben we echter veel toevalsgetallen nodig die we bij voorkeur weer moeten kunnen reproduceren zodra zij nodig zijn. Het is mogelijk om met behulp van een computer toevalsgetallen te genereren. Dit blijkt moeilijk te zijn en het kost veel geheugenruimte.
Als een toevalsgetal r_i een functie is van één of meerdere voorgaande toevalsgetallen r_{i-1}, r_{i-2}, enzovoort kan je ook toevalsgetallen genereren. Hierdoor zijn de getallen echter niet echt willekeurig. Ze worden dan ook pseudo-toevalsgetallen genoemd. Ze zijn namelijk voorspelbaar en reproduceerbaar.

De basis van de waarden van random variabelen is een uniforme verdeling op het interval [0,1] mogelijk. Bij simulatie worden vaak trekkingen gedaan uit verdelingen. De basisgetallen worden daarom nog een keer bewerkt, zodat de toevalsgetallen aan een bepaalde verdeling voldoen.

De toevalsgetallen worden gemaakt door een getal x_n te genereren op het interval [0,m], waarbij m een groot getal is. De lengte van de periode (cyclusiengte), voordat de getallen zich gaan herhalen, is mede afhankelijk van m. Vanuit x_n wordt een getal r_n gemaakt door:

$r_n = x_n / m.$

In 1949 is de lineaire-congruentiemethode ontwikkeld door D.H. Lehmer. De reeks gehele getallen $\{x_n\}$ wordt hierbij gegenereerd door middel van de volgende formule:

$x_{n+1} = (ax_n + c) \bmod m \text{ met } n = 0, 1, 2, 3,...$

De constanten a, c, m zijn geheel en positief; x_0 is de startwaarde. Al deze getallen dienen geschikt te worden gekozen.

Getalvoorbeeld:
Neem $a = 3$, $c = 4$, $m = 5$ en $x_0 = 4$
$x_0 = 4$
$x_1 = (3 \cdot 4 + 4) \bmod 5 = 16 \bmod 5 = 1$
$x_2 = (3 \cdot 1 + 4) \bmod 5 = 7 \bmod 5 = 2$
$x_3 = (3 \cdot 2 + 4) \bmod 5 = 10 \bmod 5 = 0$
$x_4 = (3 \cdot 0 + 4) \bmod 5 = 4 \bmod 5 = 4$
$x_5 = (3 \cdot 4 + 4) \bmod 5 = 16 \bmod 5 = 1$
enzovoort.

We zien dat $x_0 = x_4$, dat wil zeggen bij dit voorbeeld krijgen we na x_0 drie verschillende getallen en daarna vindt alleen het repeteren hiervan plaats. Dit is een verschijnsel dat vaak voorkomt bij het genereren van toevalsgetallen. Er is veel onderzoek gedaan naar geschikte waarden voor a, c en m. In dit voorbeeld is $m = 5$ en als gevolg daarvan krijgen we toevalsgetallen bestaande uit één cijfer. In het algemeen geldt: als $m = 10^s$, met als s een positief geheel getal, dan bestaat x_n uit s cijfers.

De lineaire congruentiemethode werkt redelijk efficiënt, is eenvoudig te implementeren, neemt weinig geheugenruimte in beslag en bij een juiste keuze van de constanten a en c, kan de cycluslengte redelijk lang zijn. Deze methode wordt veel toegepast.

In de praktijk wordt vaak $c = 0$ gekozen. Je hebt dan te maken met de methode:

$x_{n+1} = (ax_n) \bmod m$ met $n = 0, 1, 2, 3,...$

Deze methode heet de multiplicatieve congruentiemethode. De getallen a en m zijn weer geheel en positief, maar nu moet ook gelden $a \neq 1$ en $x_0 \neq 0$. Met deze methode kunnen hoogstens m verschillende toevalsgetallen verkregen worden.

Getalvoorbeeld:
Neem $a = 123$, $m = 10^4$ en $x_0 = 3.479$, dan vinden we het volgende schema:

n	x_n	ax_n
0	3.479	427.917
1	7.917	973.791
2	3.791	466.293
3	6.293	774.039
4	4.039	496.797
5	6.797	836.031
6	6.031	741.813
7	1.813	222.999
8	2.999	368.877
9	8.877	1.091.871
10	1.871	230.133

De multiplicatieve congruentiemethode vergt minder berekeningen dan de lineaire congruentiemethode. Deze methode wordt tegenwoordig ook zeer veel gebruikt.
Continue en discrete simulatie wordt in verschillende wetenschapsgebieden toegepast. Hierdoor hebben continue en discrete simulatie zich onafhankelijk van elkaar ontwikkeld.

3. Continue modellen

Continue modellen zijn vaak deterministisch. Dat betekent dat er geen sprake is van toeval. In het model verandert de onafhankelijke variabele (meestal de tijd) continu. Als een toestand van het model en de modelbeschrijving gegeven zijn, kan de toestand op een later tijdstip worden berekend. Wanneer in het model een beperkt aantal discrete toestandsveranderingen plaatsvindt, terwijl in het algemeen het gedrag continu is, spreekt men toch over een continu model. Iedere keer als deze berekeningen uitgewerkt worden, zal het rekenproces dezelfde resultaten geven. Continue modellen worden vooral toegepast in technische toepassingen.

In een continu model verandert de toestand van een systeem continu evenals de toename van de tijd.
Continue modellen bestaan in het algemeen uit gewone vergelijkingen en differentiaalvergelijkingen. Met name de differentiaalvergelijkingen spelen een belangrijke rol, omdat die de toestandsveranderingen beschrijven.

4. Discrete modellen

In een discreet model gaan de toestandsveranderingen, op discrete tijdstippen, schoksgewijs. Discrete modellen bevatten vaak stochastische variabelen. Deze modellen kunnen toegepast worden bij wachttijd-problemen.
In het model worden de toestandsveranderingen gespecificeerd en de tijdstippen waarop die veranderingen optreden vastgelegd. In de praktijk gebeurt dat op drie manieren: gebeurtenisbeschrijving, activiteiten-beschrijving en procesbeschrijving.
In een model kan er sprake zijn van twee soorten componenten: vaste componenten en tijdelijke componenten. In de componenten worden de acties beschreven.
De momenten waarop de activiteiten beginnen en eindigen, zijn de gebeurtenissen of events. Handelingen die tijd kosten worden aangegeven door twee events, namelijk de starttijd en de eindtijd van de activiteit in kwestie. Het tijdsverschil daartussen bepaalt de behandeltijd.

Gebeurtenissen kunnen op drie manieren gegroepeerd worden:
- De activiteitenbeschrijving; de gebeurtenissen worden afzonderlijk beschreven met de voorwaarden waaronder ze moeten worden uitgevoerd.
- De eventbeschrijving; de gebeurtenissen worden gegroepeerd rondom het tijdstip waarop ze optreden.
- De procesbeschrijving; de gebeurtenissen worden gegroepeerd rondom de componenten waarop de gebeurtenissen betrekking hebben.

Activiteitenbeschrijving
Bij een activiteitenbeschrijving worden alle soorten activiteiten (handelingen, acties) die in het systeem kunnen voorkomen beschreven. Er worden voorwaarden vastgelegd, waaraan de activiteiten moeten voldoen. Het simulatiesysteem doorloopt continu alle voorwaarden. Als aan één van de voorwaarden is voldaan, worden de daarbij behorende activiteiten uitgevoerd. Omdat de toestand van het systeem na ieder event verandert, moeten alle voorwaarden na ieder event doorlopen worden. Deze beschrijvingswijze is niet echt populair geworden, doordat computer-programma's er niet efficiënt mee kunnen werken. De programma's zouden efficiënter worden wanneer alleen maar die voorwaarden doorlopen worden, die als gevolg van de uitvoering van de acties veranderd zijn.
Het voordeel is echter, dat in een programma-onderdeel waar de beschrijving van een activiteit aan de orde komt ook alle bijzonderheden van die activiteit zijn te vinden.

Eventbeschrijving
Bij een eventbeschrijving worden de gebeurtenissen gegroepeerd op grond van het tijdstip waarop de gebeurtenissen optreden. Een eventbeschrijving zal in het algemeen - in een simulatietaal of een algemene programmeer-taal - worden geïmplementeerd in een zogenaamde eventroutine. Met alleen eventbeschrijvingen kan de werking van een systeem nog niet worden vastgelegd. Er moet nog aangegeven worden op welke tijdstippen de diverse events plaatsvinden.

Onder eventnotities wordt verstaan het eventtijdstip met de daarbij-behorende eventroutine.
Door vooraf de volledige lijst van alle eventnotities te ontwerpen en bij iedere eventnotitie ook het eerstvolgende event vast te stellen, kan er een volledige beschrijving van het systeem gemaakt worden.

Het simulatiesysteem moet er voor zorgen dat alle eventnotities worden afgewerkt, waardoor weer nieuwe eventnotities aangemaakt kunnen worden. Alle eventnotities die bekend zijn komen op een eventlist te staan, in volgorde van tijdstip van afhandeling. De eventnotitie bevat behalve het moment waarop het event aan de beurt moet komen, ook informatie over welke eventbeschrijving moet worden uitgevoerd. Een eventlist is te vergelijken met een agenda, waarin de activiteiten van een persoon (met de tijdstippen) genoteerd staan.

Om de simulatie uit te voeren moet er nog een mechanisme zijn om de eventnotities achtereenvolgens af te werken. Dat mechanisme wordt klok-mechanisme genoemd. Dit is een routine die er voor zorgt dat de event-notitie die als eerste op lijst staat ook als eerste wordt afgewerkt.
Het afwerken van een eventnotitie houdt in dat de bijbehorende eventroutine wordt uitgevoerd en dat de eventnotitie van de eventlist wordt verwijderd. Vervolgens wordt de eerste eventnotitie weer afgewerkt, enzovoort. Door het uitvoeren van een eventroutine kunnen er één of meer nieuwe eventroutines geplaatst worden.

Het bepalen van toekomstige eventtijdstippen en het vormen van de desbetreffende notities dient in de eventbeschrijving te worden opgenomen.

De procesbeschrijving
Een procesbeschrijving is de beschrijving van de activiteiten die een component uitvoert en van de daarbij behorende interacties met de overige componenten. Op deze manier ontstaat er een model waarin de structuur van het werkelijke systeem duidelijk herkenbaar is. De toestands-veranderingen worden veroorzaakt door het totale aantal activiteiten dat door alle componenten wordt uitgevoerd. Deze aanpak geeft vooral bij systemen waar de componenten verschillende handelingen moeten verrichten, een realistisch model.
Elke activiteit moet eenduidig worden toegewezen aan een component. Voor het model maakt het in principe niet uit welke activiteit door welke component wordt uitgevoerd. Voor de implementatie is het echter wel van belang dat een activiteit wordt toegekend aan de component die variabelen bevat die door de activiteit gewijzigd worden.

4.1. Wachttijdtheorie

In de dagelijkse praktijk worden we allen geconfronteerd met wachtrijen: files op de autowegen, wachtrijen in de wachtkamer bij de dokter, voor het loket bij het postkantoor of bij de kassa van de supermarkt. Wachtrijen zouden niet ontstaan als er genoeg capaciteit was om alles direct te behandelen ten opzichte van het aanbod.
Bij wachtrijen zijn altijd drie categorieën mensen betrokken, namelijk klanten, bedienden en het management van het bedrijf. Het is de taak van

het management om de wachttijden van de klanten af te wegen tegen de kosten van bedieningspersoneel.

Bepaalde situaties uit de praktijk kunnen voorgesteld worden als een wachttijdsituatie. Bekende formules uit de wachttijdtheorie kunnen worden toegepast bij de validatie van modellen van ingewikkelde situaties. Voor ingewikkelde situaties past men dus de formules toe die voor die wachttijdsituatie gelden. Hierdoor bespaart men een kostbare simulatiestudie.

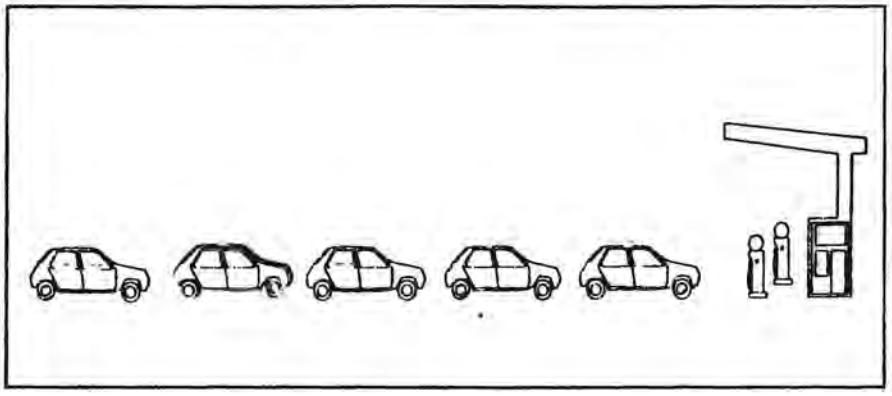
Wachttijdtheorie werd al omstreeks 1910 door de Deen A.K. Erlang toegepast in het telefoonverkeer. Zijn werk werd voortgezet door de Zweed C. Palm vanaf de jaren dertig. Gedurende en na de Tweede Wereldoorlog ontwikkelde de wachttijdtheorie zich snel. Toepassingen van wachttijdtheorie vindt men onder meer bij telefoonverkeer, voorraadbeheer en produktieprocessen.
Iedere activiteit heeft zijn eigen benodigde tijdsduur. Er ontstaat een wachtrij als aan een activiteit wordt gewerkt, terwijl zich één of meerdere andere activiteiten aandienen. Door het wachten ontstaan kosten.
Het wachtsysteem wordt bepaald door:
- Het aankomstproces;
- De wachtrij;
- Het bedieningsproces.

Het aankomstproces vormt de invoer van het wachtsysteem. De personen, machines of produkten die moeten worden bediend, worden klanten genoemd. Het is mogelijk dat een aantal artikelen of personen samen één klant vormen.
Als aankomstproces kiest men vaak het Poisson-proces, aangezien dit het simpelste, niet-deterministische proces is.
De wachtrij kan zijn: *FIFO* (first in, first out), *LIFO* (last in, first out), *SPT* (shortest processing time first), *random selection* (volgens toeval).
De wachtruimte is in sommige gevallen beperkt.
Het bedieningsproces is het proces van het afhandelen van de klanten bij de bedieningsstations, ook wel loketten genoemd. Voor de bedieningstijd kiest men vaak de exponentiële verdeling die gekoppeld is aan het Poisson-proces voor de aankomsten bij de loketten.

Volgens de Kendall-notatie wordt een wachtsysteem aangegeven met *A/B/C*.
Hierin is *A* het aankomstproces, *B* het bedieningsproces en *C* het aantal geplaatste bedieningsstations.
Om het Poisson-proces aan te duiden, gebruikt men de letter *M* (wegens de overeenkomst met een Markov-proces).

Hier volgen een aantal systemen met voorbeelden (bij al deze systemen geldt *FIFO*, 'wie het eerst komt, het eerst maalt'):
- Het meest eenvoudige wachtrijsysteem is een *M/M/1-systeem*. Dit is een systeem met één bediende en één wachtrij, waarbij het aankomstproces en bedienproces een Poisson-proces is.
  Een voorbeeld van een systeem met onbeperkte wachtgelegenheid is het spreekuur van een arts. Bij beperkte wachtgelegenheid, kan men denken aan een tankstation in een drukke straat.

- Het *M/M/n-systeem* is een systeem met één wachtrij met n bedienden, waarbij n groter is dan 1.
  Als er geen wachtgelegenheid is, dan ontstaat er geen wachtrij. Tenminste als we veronderstellen dat een klant wegloopt als hij constateert dat alle aanwezige bedienden bezet zijn. Deze situatie doet zich vaak voor bij telefoonverkeer. Als alle lijnen bezet zijn, dan wordt het binnenkomend gesprek niet geaccepteerd en is er geen sprake van wachtgelegenheid. Ook bij parkeerplaatsen doet zich dit voor. Zelfbediening is een situatie waarbij de wachtgelegenheid beperkt is.



5. Het simuleren van verdelingen

Bij simulatie worden vaak trekkingen gedaan uit verdelingen. Deze verdelingen kunnen discreet en continu zijn. Een discrete stochastische variabele wordt aangegeven met k en een continue variabele met x.

Trekkingen uit discrete verdelingen
Voorbeeld van een discrete uniforme verdeling:
Gegeven de verdeling met kansfunctie $f(k)$.

k	f(k)
1	0,25
2	0,5
3	0,25

Bepaal nu de verdelingsfunctie en intervallen van toevalsgetallen.

k	f(k)	F(k)	intervallen van toevalsgetallen
1	0,25	0,25	0 -< 0,25
2	0,5	0,75	0,25 -< 0,75
3	0,25	1	0,75 -< 1

Stel dat toevalsgetal 0,5432 wordt gekozen, dan is $k = 2$ het resultaat van de trekking.

Trekkingen uit continue verdelingen
Het resultaat van deze trekkingen is $x = F^{-1}(r)$. Hiervoor is nodig dat de inverse van de verdelingsfunctie $F(x)$ wordt bepaald. Voor een aantal functies is de inverse functie bekend, voor andere niet.

De uniforme verdeling
De uniforme verdeling heeft, bij een gegeven interval (a,b), de kansdichtheidsfunctie:

$f(x) = 0$ voor $a > x > b$
$f(x) = 1/(b-a)$ voor $a \leq x \leq b$

en de verdelingsfunctie:

$F(x) = 0$ voor $x < a$
$F(x) = (x-a)/(b-a)$ voor $a \leq x \leq b$
$F(x) = 1$ voor $x > b$

Dan is het toevalsgetal $r = (x-a)/(b-a)$.
Dus $x = (b-a)r + a$ geeft het resultaat van de trekking.

De (negatief)-exponentiële verdeling
De exponentiële verdeling heeft de kansdichtheidsfunctie:

$f(x) = 0$ voor $x < 0$
$f(x) = \lambda e^{-\lambda x}$ voor $x \geq 0$

en de verdelingsfunctie:

$F(x) = 0$ voor $x < 0$
$F(x) = 1-e^{-\lambda x}$ voor $x \geq 0$

Dan is $r = 1-e^{-\lambda x}$ en $x = (-1/\lambda)ln(1-r)$.
r is uniform verdeeld tussen nul en één, dus ook $1-r$.
Dus $x = -(1/\lambda)ln(r)$.

6. Vergelijken met het werkelijke systeem

Validatie is het bewijzen dat het model een goede vervanger is van het werkelijke systeem. Dit is belangrijk, want een model moet ook toepasbaar zijn in andere situaties dan het simulatiemodel. Van een model wordt een implementatie in de vorm van een computerprogramma gemaakt. Die implementatie moet eerst gecontroleerd worden. Het doel hiervan is de resultaten, die uit de experimenten met het model voortkomen, binnen bepaalde grenzen overeen te laten komen met de uitkomsten van dezelfde experimenten met het werkelijke systeem.

De beste manier om een model te valideren is te experimenteren met het werkelijke systeem en dus het uitvoeren van dezelfde experimenten met het model. Vervolgens worden de resultaten met elkaar vergeleken en wordt er beoordeeld of de resultaten in voldoende mate met elkaar overeenstemmen. Wanneer de resultaten niet met elkaar overeenstemmen moet het model aangepast worden.
Op deze wijze is het alleen mogelijk om het model te onderzoeken als dezelfde experimenten ook worden uitgevoerd met het werkelijke systeem. In feite is het doel van simulatie: onderzoekingen doen (door middel van experimenten) met het model die ook geldig zijn voor het werkelijke systeem, zonder dat alle experimenten herhaald moeten worden met het systeem zelf.

De periode die we simuleren moet overeenstemmen met de periode van het reële systeem waarin men geïnteresseerd is. Om een zekere betrouwbaarheid te verkrijgen moeten er meerdere simulatieruns worden uitgevoerd. Voor elke simulatierun bepalen we de resultaten op een gegeven aantal momenten. Vervolgens wordt het gemiddelde genomen van de resultaten van de afzonderlijke runs voor de overeenkomstige momenten.

Als het werkelijke systeem nog niet bestaat, dan zal het ook niet mogelijk zijn om vergelijkingen met het werkelijke systeem te maken. In dat geval moeten er andere methoden worden gebruikt om de betrouwbaarheid van het model na te gaan.

7. Simulatietalen

Veel simulatieprogramma's worden in een algemene programmeertaal geschreven. *FORTRAN* is hiervan de belangrijkste, maar ook *PASCAL, BASIC, C* en andere programmeertalen worden gebruikt. *FORTRAN* is een taal die veel wordt toegepast op technisch-wetenschappelijk gebied, waar simulatie vrijwel altijd wordt toegepast. Het grote voordeel van *FORTRAN* dat het bij vrijwel alle computercentra geïnstalleerd is en er grote hoeveelheden routines beschikbaar en aanroepbaar zijn. Toch kiezen veel mensen voor een speciale simulatietaal. Er is in de loop der jaren een grote hoeveelheid simulatietalen op de markt verschenen.

Toepassing van een simulatietaal heeft een aantal voordelen ten opzichte van het formuleren van simulatieprogramma's in een algemene programmeertaal. Deze zijn:

- De structuur van het model kan beter in overeenstemming worden gebracht met de structuur van het werkelijke systeem. Het model is eenvoudiger te formuleren en een simulatieprogramma is tevens beter geschikt om een model te documenteren.
- De maker van het model kan zich beter concentreren op het model, omdat hij zich niet bezig hoeft te houden met allerlei details.
- De mogelijkheid om gegevens te verzamelen voor bijvoorbeeld een statistiek of grafiek, en het weergeven van die gegevens met de daarvoor bedoelde taalconstructies en routines.
- Een onervaren programmeur zal eerder in staat zijn het model te formuleren in een simulatietaal dan in een algemene programmeertaal.

Literatuur

Kettenis, D.L. (1990). *Simulatie*. Stenfert Kroese, Leiden.

Griep P.A.M. & Flapper S.P.D. (1987). *Discrete simulatie*. Academic Service, Schoonhoven.

Vliegenthart, A.C. (1993). *Leerboek Operations Research*. Academic Service, Schoonhoven.

Winston, W.L. (1987). *Operations Research; Applications and Algorithms*. PWS-Kent, Boston.