

VO Einführung Simulation

SS 2020

***Helge Hagenauer
FB Computerwissenschaften***

Teil 1

1 Motivation - Was ist Simulation?

Einige Begriffsdefinitionen zu **Simulation**:

- Vortäuschung (von Krankheiten); Sachverhalte, Vorgänge (mit technischen, naturwissenschaftlichen Mitteln) modellhaft zu Übungs-, Erkenntniszwecken nachbilden, wirklichkeitstreue nachahmen.
(Duden)
- Technische oder naturwissenschaftliche Prozesse, Funktionen in ihren wesentlichen Grundzügen nachbilden.
(Das Digitale Wörterbuch der deutschen Sprache; www.dwds.de)
- Simulation in der Informatik: Nachbildung von Vorgängen auf einer Rechenanlage auf der Basis von Modellen (= im Computer darstellbare Abbilder der realen Welt).
(nach Informatik-Duden)
- Die Simulation oder Simulierung ist eine Vorgehensweise zur Analyse von Systemen, die für die theoretische oder formelmäßige Behandlung zu komplex sind. Dies ist überwiegend bei dynamischem Systemverhalten gegeben. Bei der Simulation werden Experimente an einem Modell durchgeführt, um Erkenntnisse über das reale System zu gewinnen. Im Zusammenhang mit Simulation spricht man von dem zu simulierenden System und von einem Simulator als Implementierung oder Realisierung eines Simulationsmodells. Letzteres stellt eine Abstraktion des zu simulierenden Systems dar (Struktur, Funktion, Verhalten). ...
(de.wikipedia.org)

Beweggründe für Simulation

Allgemein zusammengefasst (nach B.Page, siehe Literatur [2]):

Studying a system from multiple points of view and predicting how, as well as understanding why it will react to different inputs are main motivations for users of modelling tools.

- ➡ Erlangung eines besseren Verständnisses für Systemverhalten und Wirkungsbeziehungen

Since simulation can explore models with any degree of complexity, it can further these goals where other techniques may fail. Whenever a problem's solution requires a model too complex for mathematical optimization, simulation should be the tool of choice.

→ Systeme beliebiger Komplexität behandelbar

Meist dient Simulation der Untersuchung von Abläufen, die in der Realität nicht durchführbar sind wegen:

- Zeitverbrauch (z.B. Klimamodelle, Wachstum von Organismen, ...)
- Kosten (z.B. Crashtest, Schulung an Simulatoren, ...)
- Gefahren (z.B. Airbag, Kernreaktoren, Schulung an Simulatoren, ...)
- ...

Anwendungsbereiche

Simulation kann praktisch in allen Bereichen angewendet werden, z.B.:

- Natur- und Ingenieurwissenschaften
- Wirtschafts- und Gesellschaftswissenschaften
- Informatik
- Psychologie und Medizin
- Ökologie, Umweltverhalten
- Geisteswissenschaften

Einige Anwendungsbeispiele

- Management:
Betriebsabläufe, Personalplanung, Lagerverwaltung, Geschäftsprozesse

- Produktionssysteme:
Planung/Kontrolle (automatischer) Fertigungssysteme
- Informatik, IT:
Rechnerkonfiguration, Betriebssysteme, Netzwerkanalyse/-planung, Datenbankdesign, Sicherheitsaspekte, Systemausfälle/Notfallstrategien
- Transportwesen, Verkehr, Logistik:
Flughafen-Design, Planung von Verlade-/Containerterminalen, Verkehrssysteme planen, Straßennetz (Leitsysteme, Kapazitäten, Steuerung von Ampeln), Auslieferungssystem

2 Literaturhinweise

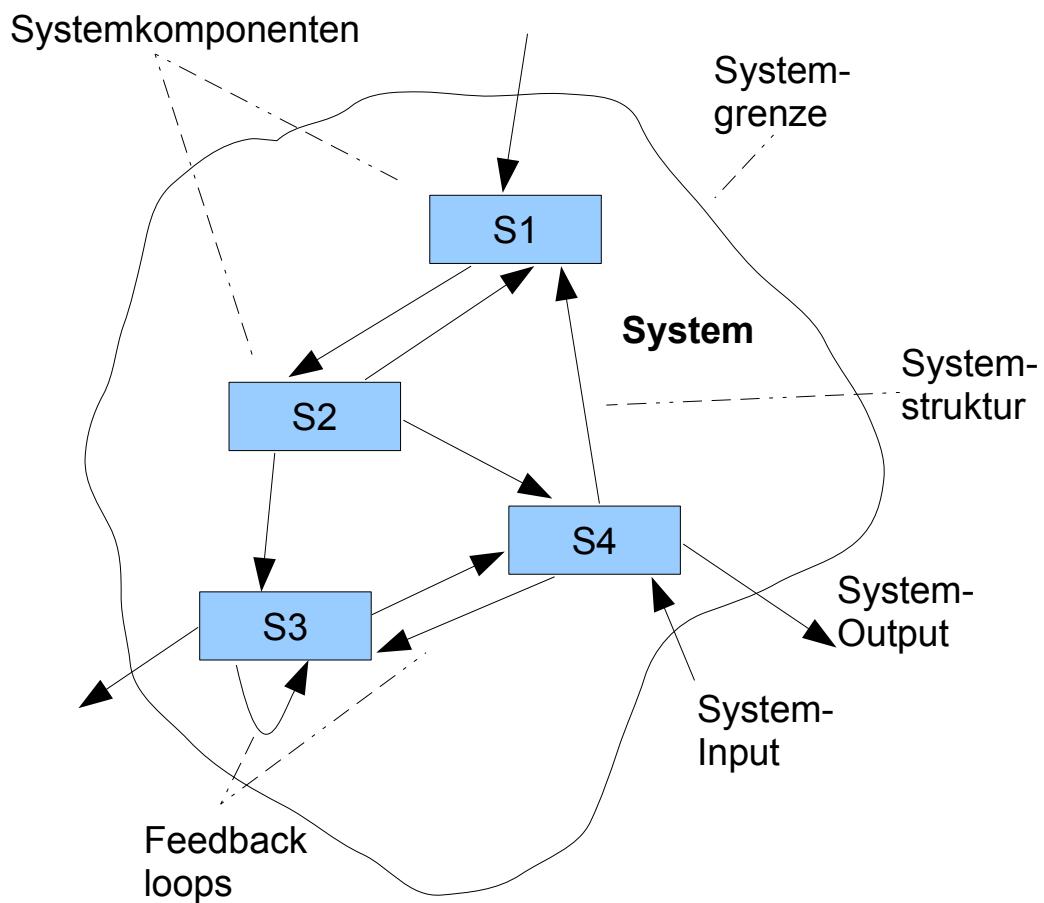
- [1] Banks, J., J.S.Carson, B.L.Nelson, D.M.Nicol.
Discrete-Event System Simulation.
2010. 5th Edition. Pearson .
- [2] Page, B., W.Kreutzer.
The Java Simulation Handbook – Simulating Discrete Event Systems with UML and Java.
2005. Shaker Verlag.
- [3] Law, A.M.
Simulation Modeling & Analysis.
2015. McGraw-Hill Inc.
- [4] Leemis, L.M., S.K.Park.
Discrete-Event Simulation: A First Course.
2006. Pearson Prentice Hall.
- [5] Zeigler, B.P., H.Praehofer, T.G.Kim.
Theory of Modeling and Simulation.
2018. Academic Press.
- [6] ev. weitere Hinweise während der VO

3 Wichtige Begriffe

System

- Ausschnitt der Realität, welcher unter einer besonderen Fragestellung betrachtet wird

- kann materiell (real) oder immateriell (nicht stofflich, geistig, Ideengebilde) sein
- besitzt eine Schnittstelle zur Umwelt (von Fragestellung bestimmt)
- besteht aus identifizierbaren Komponenten, die zu einem bestimmten Zweck interagieren (innerhalb des Systems)



Modell

- materielle oder immaterielle Systeme zur Darstellung anderer Systeme
- experimentelle Manipulation der abgebildeten Strukturen und Zustände soll möglich sein

Ein Modell soll eine Bestimmung haben bzw. für die Bearbeitung bestimmter

Fragestellungen ausgerichtet werden.

Übergang Realsystem → Modell: *Abstraktion, Idealisierung* nötig, d.h. vereinfachte Darstellung des Originals unter Beachtung der Fragestellung.

Weiters ist bei Modellen zu bedenken:

- zu modellierende Eigenschaften sind abhängig von der Fragestellung (d.h. unterschiedliche Modelle zu einem Realsystem möglich)
- ausführliche Tests (statistisch, qualitativ) zur *Validierung* eines Modells notwendig
→ ausreichend Daten über Realsystem sollten vorliegen – Vergleichsmöglichkeit
- wenn Abweichung der gewonnenen Daten vom Original → Hinweis auf fehlerhafte Modellbildung:
 - große Differenz: mögliche Fehler im Modell – falsche Abbildung der Realität
 - kleine Differenz: Kalibrierung vornehmen (Änderung weniger Modellparameter)

Verschiedene Gesichtspunkte bei Modellklassifizierungen möglich, wie z.B. Zeitbezug, Verwendungszweck oder nach Art der Repräsentation:

- physikalisches Modell (z.B. maßstabgetreue Nachbildung eines Flugzeugs oder Schiffs)
- verbales Modell (z.B. Wegbeschreibung zu einer bestimmten Adresse)
- beschreibendes graphisches Modell (z.B. UML-Aktivitätsdiagramm)
- mathematisch-graphisches Modell (z.B. Petri-Netz)
- abstraktes mathematisches Modell (z.B. Menge von Differentialgleichungen)
- abstraktes algorithmisches Modell (z.B. diskrete Ereignissimulation)

Simulation

Jetzt zu Definitionen im Sinn dieser LV:

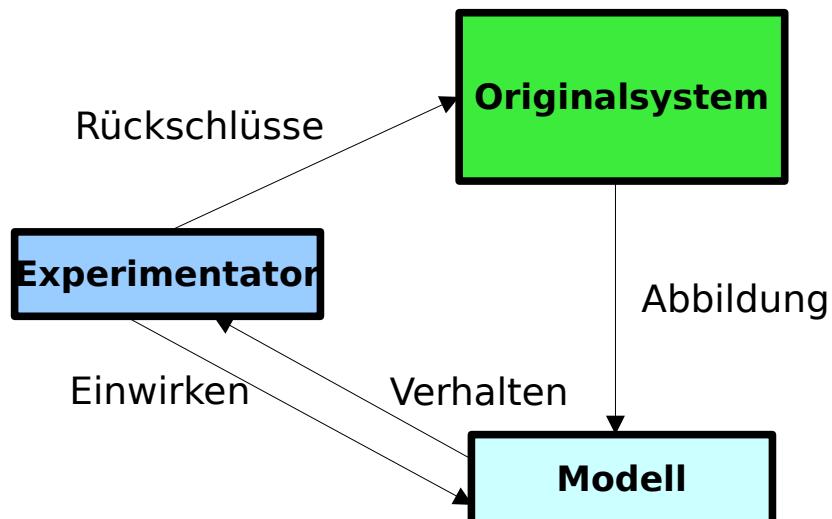
- Simulation is the process of describing a real system and using this model for experimentation, with the goal of understanding the system's behaviour or to explore alternative strategies for its operation.

(Shannon 1975)

- Simulation is the modelling of dynamic processes in real systems, based on real data, and seeking predictions for a real system's behaviour by tracing a system's changes of state over time.

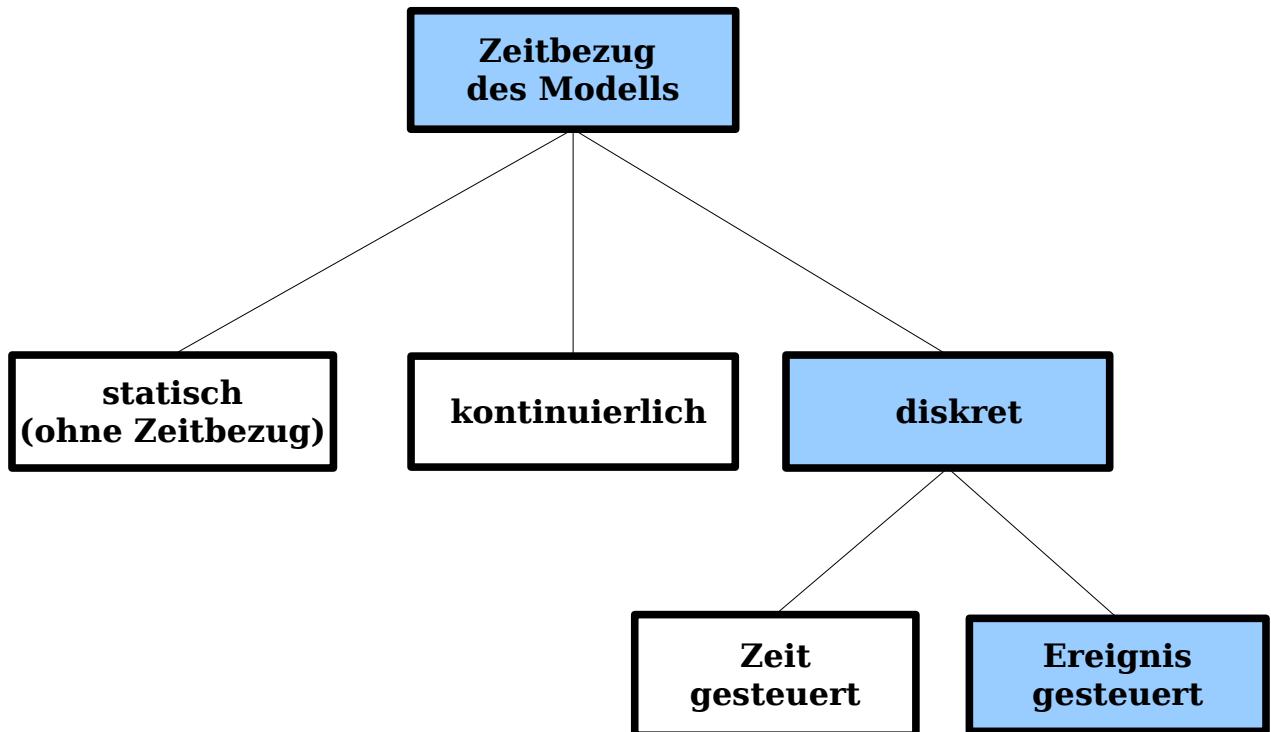
(*The Java Simulation Handbook*. B.Page, W.Kreutzer [2])

Daraus ergeben sich folgende Beziehungen



→ beschäftigen uns mit: **Simulation als Teilgebiet der Informatik**

4 Modellklassifikation nach Zeitbezug



- **statische Modelle:** kein Bezug zur Zeit (z.B. Monte Carlo Simulation)
- **kontinuierliche Simulation:** stetige Zustandsänderung über der Zeit; Modell als System von Differentialgleichungen (mit freier Zeitvariablen);
Simulation heißt hier: lösen der Modellgleichungen (meist numerisch) z.B. für komplexe biologische oder physikalische Prozesse (Wettervorhersage, Strömungslehre, Analyse von Ökosystemen, ...)
- **diskrete Ereignis Simulation (discrete event simulation):** Änderung des Systemzustands nur zu bestimmten (Ereignis-) Zeitpunkten (keine Änderungen dazwischen!)
Unterteilung in
 - **Ereignis gesteuert (event-driven):** Zeitabstände zwischen aufeinanderfolgenden Ereignissen sind unterschiedlich
 - **Zeit gesteuert (time-driven):** gleiche Zeitabstände zwischen

aufeinanderfolgenden Ereignissen

5 Komponenten und Grundkonzept diskreter Simulation

Grundlegende Komponenten von Simulationsmodellen sind:

Entitäten: Komponenten des Simulationsmodells, welche Teile des realen Systems modellieren und miteinander in Wechselbeziehung stehen.

Entitäten sind gekennzeichnet durch: Zustand und Transformationsregeln.

Entität ↔ Objekt (nach OOP): Entität entspricht einem Objekt, dessen Verhalten durch die Simulationszeit bestimmt wird.

Zustand (state): aktuelle Belegung der Attribute einer Entität.

Transformationsregeln (rules): Menge von Regeln (oder Methoden) welche den Zustand einer Entität im Laufe der Simulationszeit verändern (→ Zeitbezug nötig!).

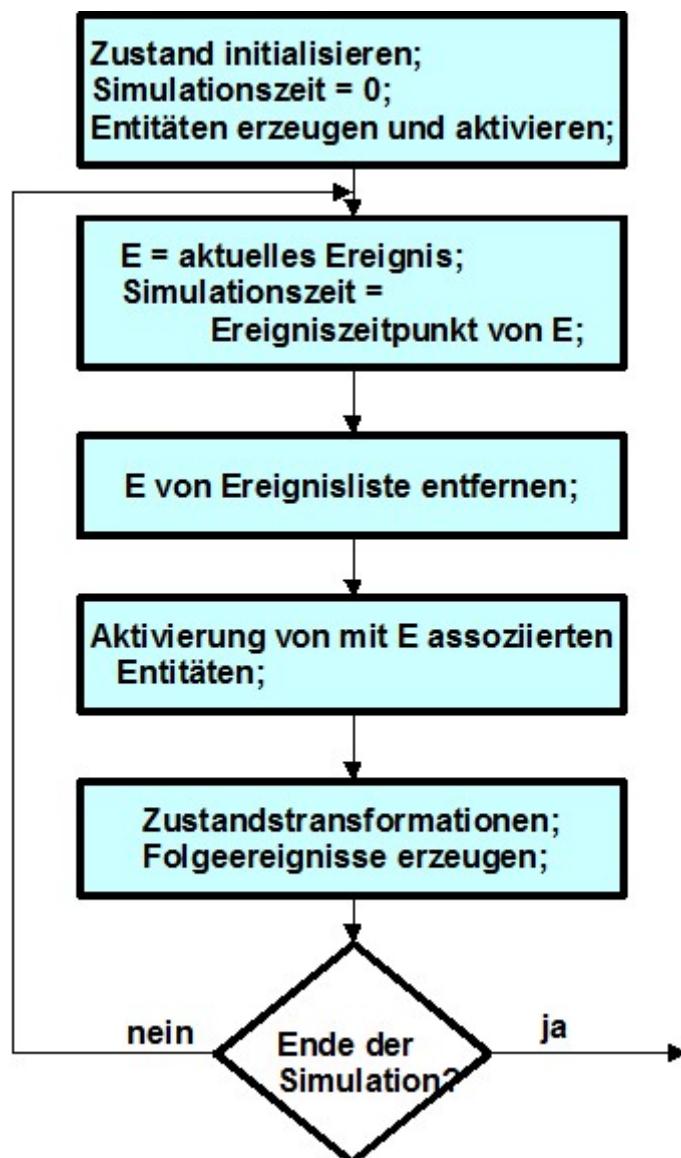
Simulationszeit: fiktive Modellzeit, unabhängig von der realen Zeit und der Ausführungsdauer einer Simulation

Diskrete Ereignis-Simulation bedeutet:

- Simulationszeit springt von Ereignis zu Ereignis.
Ein Ereignis löst maßgebliche (relevante) Zustandsänderungen bei einer oder mehreren Entitäten aus.
- Keine Zustandsänderungen zwischen zwei aufeinanderfolgenden Ereignissen.
- Zustandsänderungen erzeugen i.A. Folgeereignisse.
- Jedes Ereignis

- besitzt einen *Ereignistyp*
- ist assoziiert mit einer Entität (oder mehreren)
- besitzt einen **Eintritts- oder Ereigniszeitpunkt** (= Zeitpunkt der Simulationszeit)

Daraus ergibt sich als Grundkonzept folgende prinzipielle Funktionsweise:



Dabei ist zu beachten:

- aktuelles Ereignis = Ereignis mit kleinstem Ereigniszeitpunkt

- **Ereignisliste:** verwaltet alle noch nicht verarbeiteten (zukünftigen) Ereignisse
- Ereignisse mit gleichem Eintrittszeitpunkt: Verarbeitung in beliebiger Reihenfolge (z.B. FCFS-Strategie, Prioritäten)
- Ende der Simulation: mehrere Möglichkeiten wie z.B. bestimmte Zeit verstrichen, spezielles Ende-Ereignis tritt ein, oder Ereignisliste ist leer

Beziehung Modellzustand – Simulationszeit

Es existieren folgende grundlegende Möglichkeiten um Zustandsänderungen und Simulationszeit zu synchronisieren:

Ereignis: beschreibt/bewirkt die Zustandsänderung von Entitäten zu einem bestimmten Zeitpunkt der Simulationszeit.

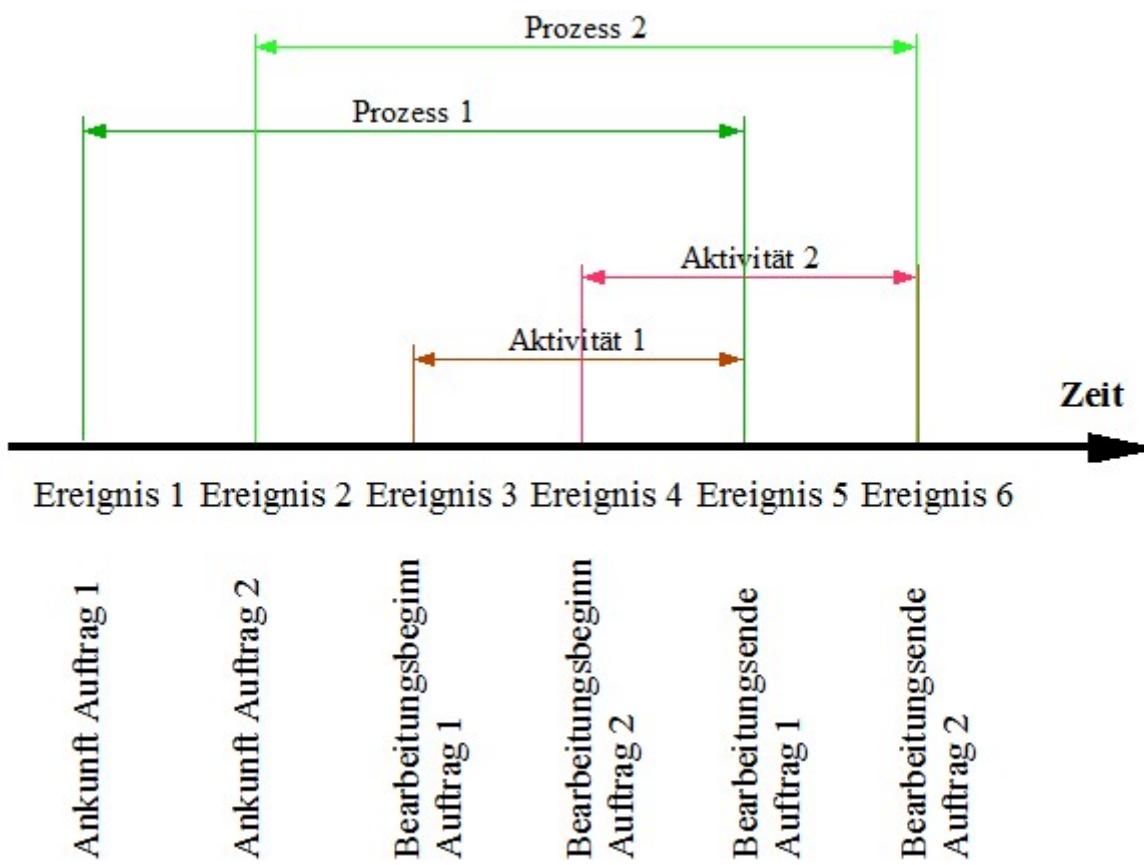
Externe Ereignisse: keine Abhängigkeit von anderen Ereignissen, werden von der Systemumgebung vorgegeben (z.B. eintreffen eines Auftrags).

Interne Ereignisse: entstehen durch Zustandsänderungen (z.B. Bearbeitungsende eines Auftrags).

Aktivität: besteht aus einer Menge von Operationen, welche während eines Intervalls der Simulationszeit ausgeführt werden; Zustandsänderung (Wirkung) wird zu Beginn und am Ende der Aktivität ausgeführt (z.B. Bearbeitung eines Auftrags)

Prozess: besteht aus einer Folgen von Aktivitäten einer Klasse von Entitäten, die den Lebenszyklus dieser beschreiben (z.B. gesamte Abwicklung vom Eintreffen bis zum Bearbeitungsende eines Auftrags)

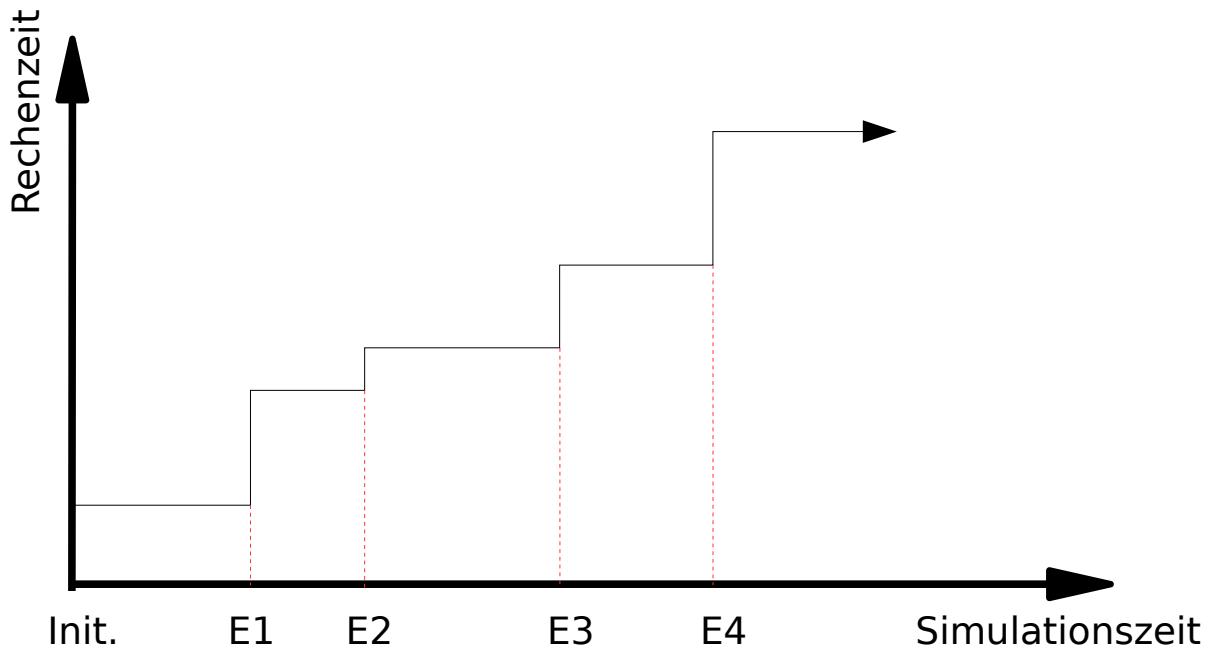
Zusammenhang dargestellt am Bsp. einer Auftragsbearbeitung (nach [2]):



Auch von Interesse ist die *orthogonale* Beziehung zwischen Simulationszeit und Rechenzeit.

- **Simulationszeit** schreitet sprunghaft voran (Abarbeitung von Ereignissen); ist *monoton steigend* (d.h. nicht fallend).
- **Rechenzeit** wird zur Ausführung am Rechner benötigt, wobei gilt:
 - Ereignisse verbrauchen Rechenzeit – aber keine Simulationszeit

- Intervalle zwischen Ereignissen verbrauchen Simulationszeit – aber keine Rechenzeit



Umsetzung in Software

Für die Implementierung wird benötigt:

- Darstellung des Systemzustands: Menge von Variablen die den Modellzustand zum Simulationszeitpunkt t beschreiben
- Simulationszeit: Variable, welche die aktuelle Simulationszeit darstellt
- Ereignisliste: Datenstruktur zur Verwaltung von Ereignissen, geordnet nach der Zeit;
notwendige Operationen darauf: einfügen geordnet nach Ereigniszepunkten, Zugriff auf aktuelles Ereignis, Ereignis löschen, Ereignis verschieben
- Zähler für statistische Auswertung: Variablen zur Speicherung (Zählung) relevanter Daten

Beispiel: Schalter mit Warteschlange

Ein Schalter (Bank, Fahr- oder Eintrittskarten) wird als **Bedienstation mit Warteschlange (single server queueing system)** modelliert. Weitere Anwendungen davon sind z.B. Friseur, Arzt, Drucker oder Router in einem Netz,

Zu beachten ist:

- Kunden kommen zu zufälligen Zeitpunkten zum Schalter
- findet ein ankommender Kunde einen freien Schalter vor, wird er sofort bedient
- ist bei Ankunft des Kunden kein Schalter frei, reiht er sich am Ende der Warteschlange ein
- nach Ende des Bedienvorgangs wird jener Kunde aus der Warteschlange genommen und bedient, der am längsten wartet (falls vorhanden)
→ FIFO-Strategie
- Simulationsbeginn: Zeitpunkt 0 und kein Kunde im System (empty-and-idle)
- Simulationsende: wenn ein bestimmter Zeitpunkt in der Simulationszeit erreicht ist (z.B. Bankschalter schließt nach 4 Stunden)

6 Ereignisorientierter Modellierungsstil

Zur Erstellung eines Simulationsmodells können verschiedene **Modellierungsstile (world views)** verwendet werden. Die wichtigsten und meist verwendeten werden besprochen.

Zuerst der **ereignisorientierte Modellierungsstil**; auch *event scheduling* genannt; historisch gesehen älterer Stil.

Das Prinzip lautet:

- Betrachtung der Gesamtheit aller Zustandsänderungen aller relevanten Entitäten zu einem Ereigniszeitpunkt
- Aktivitäten dazwischen werden nicht direkt abgebildet – erst die entsprechenden Auswirkungen beim nächsten Ereigniszeitpunkt
- Während einer Ereignisabarbeitung vergeht *keine* Simulationszeit (es wird jedoch Rechenzeit benötigt!)

Zur Erstellung eines ereignisorientierten Modells ist zu beachten:

- Identifikation der relevanten Systemobjekte und ihrer Attribute
- Beschreibung *aller* Systemzustände, Ereignisse und Zustandsänderungen der Entitäten inklusive Interaktionen
- Vogelperspektive
- Zusammenfassung aller Zustandsänderungen jener Entitäten, die durch Ereignisse zum selben Zeitpunkt stattfinden, zu *Ereignistypen*

Ereignisorientierter Modellierungsstil ermöglicht eine klare Trennung zwischen Systemstruktur (Entitäten) und Systemverhalten (Ereignisse).

Es ergeben sich 2 grundlegende Arten von Modellkomponenten:

- **Statische Komponenten:** Entitäten, die permanent existierende Objekte des Realsystems oder Klassen von Objekten darstellen; Attribute ermöglichen die Identifikation individueller Entitäten und legen die prinzipiell möglichen Zustandsübergänge fest.

- **Dynamische Komponenten:** alle Ereignisse (repräsentiert durch Ereignistypen) und temporäre Entitäten (Erzeugung und Vernichtung während der Simulation, ausgelöst durch Ereignisse).

Eine Software-Realisierung benötigt daher:

- **Ereignisroutinen/-methoden:** führen Zustandsänderungen durch und bestimmen somit das Modellverhalten durch
 - Änderung der Attributwerte von Objekten
 - Erzeugung und Löschen von temporären Objekten
 - neue, zukünftige Ereignisse der Ereignisliste hinzufügen oder existierende aus dieser löschen
- **Scheduler:** Steuerung der Ablaufkontrolle; sequenzielle Abarbeitung der nach Ereigniszeitpunkten geordneten Ereignisse (next event approach)
→ Simulationszeit springt von Ereigniszeitpunkt zu Ereigniszeitpunkt – nur dabei wird die Simualtionsuhr weiter gesetzt!

Einschub: Überblick DESMO-J

DESMO-J = Discrete-Event Simulation Modelling in Java
(siehe <http://www.desmoj.de>)

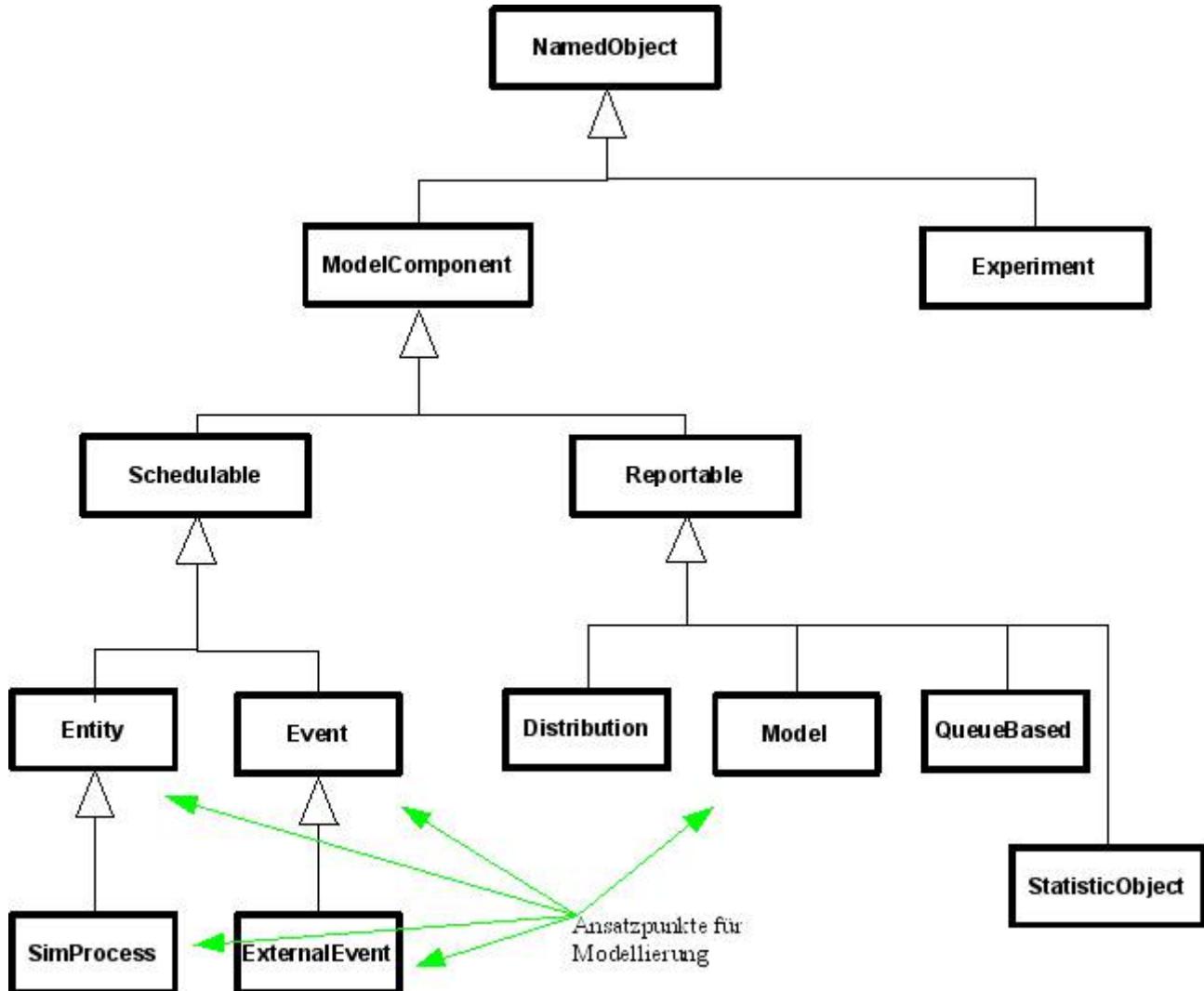
OO-Framework zur Erstellung von diskreten Ereignissimulationen (ereignis- und/oder prozessorientiert).

Wichtige Funktionen sind:

- Erzeugung und Verwaltung von Entitäten, Ereignissen und Prozessen – Modellestellung
- Simulationszeit (-uhr)
- Bereitstellung einer Experimentierumgebung (Ereignislistenmechanismus, Scheduler, ...)
- Verwendung von Warteschlangen und weiterer Hilfsmittel

- Nutzung verschiedener Zufallszahlenverteilungen
- Hilfsmittel zur statistischen Auswertung

Übersicht der Klassenhierarchie



Wichtige Klassen sind:

Experiment: Infrastruktur für einen Simulationslauf (eines Modells)

Model: Verwaltung aller Objekte eines Modells - Darstellung des Modells

Entity: Darstellung modellspezifischer Entitäten

SimProcess: Darstellung von Entitäten mit eigenem „Lifecycle“ (Prozesse)

Event: Darstellung von Ereignissen

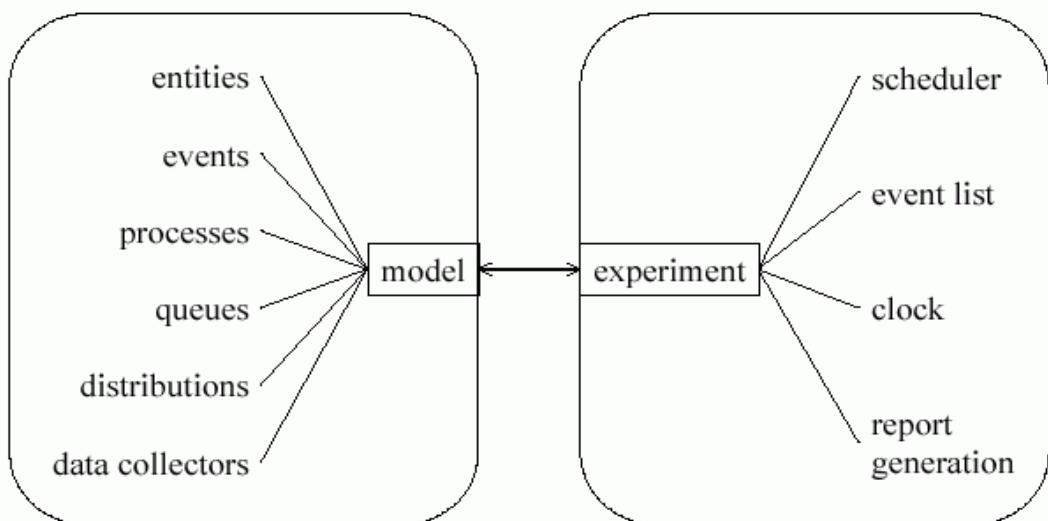
Distribution: verschiedenen Zufallszahlenverteilungen

QueueBased: Basisverhalten und Statistik für Warteschlangenkomponenten

TimeSpan: Zeitspanne der Simulationszeit

TimeInstant: Zeitpunkt der Simulationszeit

Ein Grundprinzip ist die Trennung von Modell und Experiment (Skizze aus DESMO-J Tutorial):



Leitfaden zur Modellerstellung

(nach DESMO-J Tutorial, allgemein anwendbar)

1. Pre-Implementierung

- Modell definieren: was ist nötig, was kann weggelassen werden → Abstraktion

- aktive Entitäten (actors) bestimmen – werden meist zu Ereignissen oder Prozessen
- Aktionen der aktiven Entitäten festlegen

2. Modellklasse implementieren

- Ableitung der zentralen Modellklasse von `Model`
- Definition der notwendigen Infrastruktur: Warteschlangen, Zufallszahlengeneratoren, ...
- `get...()`-Methoden für Infrastrukturelemente
- erste Ereignisse oder Prozesse in die Warteliste eintragen – `doInitialSchedules()`
- `main()`-Methode anpassen: z.B. Simulationsende festlegen

3. Entitäten implementieren

- „Actor“-Klassen von `Event` oder `SimProcess` ableiten
- Konstruktoren anpassen
- Verhalten/Aktivitäten der „Actors“ in `eventRoutine()` oder `lifeCycle()` implementieren

4. Post-Implementierung

- mit Modell experimentieren und Fehler korrigieren (Syntax, Semantik)
- „Error“-Datei prüfen → eventuelle Inkonsistenzen
- Modell validieren: schwierig (bis unmöglich), gute Simulationskenntnisse und Statistiktools nötig

7 Prozessorientierter Modellierungsstil

Später entstanden als ereignisorientierter Modellierungsstil, wurde jedoch zum meist angewendeten Stil.

Wichtige Merkmale:

- Darstellung des *vollständigen Lebenszyklus (lifecycle)* einer Entität als Prozess.
- Simulationszeit schreitet während *aktiver* Phasen der Entitäten voran – d.h. immer dann, wenn eine Zeitspanne vergeht.

Somit ergibt sich auf konzeptioneller Ebene:

Prozesse laufen i.A. zeitlich parallel ab und übergeben die Kontrolle immer dann an den *Scheduler*, wenn die Simulationszeit weiter gesetzt wird.

Danach kehrt die Kontrolle umittelbar oder nachdem andere (parallele) Prozesse ihre Zustandsänderungen durchgeführt haben zurück.

Dabei ist zu beachten:

- Zustandsänderungen geschehen während aktiver Prozessphasen.
- Im Modell dargestellte Aktivitäten oder Aktionen bewirken i.A. das Voranschreiten der Simulationszeit – jedoch Zustandsänderungen wirken augenblicklich, es verstreicht dabei *keine* Simulationszeit (benötigen aber Rechenzeit).

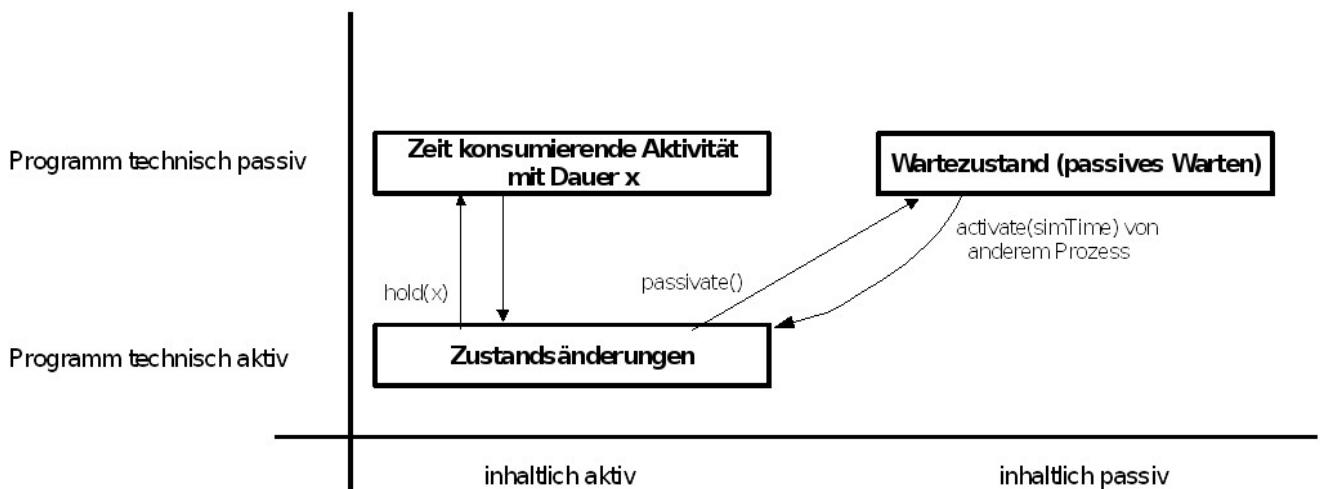
Diese Zustandsänderungen, ausgeführt von einem Prozess, sind z.B.

- Attribute von Entitäten modifizieren
- neue Prozesse (und deren Entitäten) generieren
- Aktivierung anderer Prozesse zu bestimmten Simulationszeitpunkten
- geplante Aktivierungen verschieben oder löschen

- selbst deaktivieren und Kontrolle an Scheduler oder anderen Prozess abgeben
- sich selbst oder andere Prozesse beenden

Somit ergibt sich, dass die Abbildung von zeitverbrauchenden Aktivitäten mittels *inaktiver* Prozessphasen geschieht. Es gibt 2 Arten davon:

- inaktiv für vorgegebenes fixes Intervall der Simulationszeit (mittels `hold()`): Abbildung einer *zeitverbrauchenden Aktivität* (Kontrolle inzwischen an Scheduler);
- inaktiv für unbestimmte Dauer der Simulationszeit (mittels `passivate()`): Abbildung eines *Wartezustands* (Kontrolle auf unbestimmte Zeit abgegeben, Reaktivierung meist durch andere Prozesse)



Der *Scheduler* steuert im Hintergrund die Abarbeitung beliebig vieler Prozesse:

- Ausführung der Prozesse in richtiger Reihenfolge und vorgesehenen Zeitabständen
- (Re-)Aktivierung eines Prozesses entspricht einem Ereignis
 - Verwaltung der zu aktivierenden Prozesse in einer *Ereignisliste*
 - Auswahl des nächsten Aktivierungs- (= Ereignis-)zeitpunkts und

entsprechenden Prozess reaktivieren

- Abarbeitung von Prozessen i.A. nicht von Anfang bis Ende ohne Unterbrechung (im Gegensatz zu Ereignissen)
→ Status bei Deaktivierung ist zu speichern, um spätere Fortsetzung zu ermöglichen
- daher: Implementierung dieses Ansatzes eher aufwändiger!

Zur Erstellung eines prozessorientierten Modells ist zu beachten:

- Identifikation der relevanten Systemobjekte und ihrer Attribute
→ im Modell als Entitäten oder Prozesstypen dargestellt
- relevante Aktivitäten und Attribute dieser Objekte festlegen
- Beschreibung des *Lebenszyklus (lifecycle)* von jedem Prozesstypen

d.h. aus der Sicht des zu modellierenden Objekts sind alle relevanten Aktivitäten inkl. deren Reihenfolge sowie Beziehungen zu anderen Entitäten zu beschreiben

- Froschperspektive

Vergleich ereignis-/prozessorientierter Modellierungsstil

ereignisorientiert	prozessorientiert
ähnliche interne Sicht: sequentielle Abarbeitung der Ereignisliste und weitersetzen der Simulationsuhr an den Ereigniszeitpunkten	
Ereignisliste beinhaltet: Ereignisse mit Ereigniszeitpunkt (und Verweis auf betroffene Entitäten)	Ereignisliste beinhaltet: Prozesse mit ihren (Re-)Aktivierungzeitpunkten
typische Systemänderungen (meist keine Zeit verbrauchend) mittels Ereignissen natürlicher abbildbar	Abbildung aktiver und passiver Phasen von Komponenten näher an der Realität
schlechte Übersichtlichkeit bei komplexen Interaktionen von Komponenten	Übersicht besser durch Verwandschaft zu OO
Orientierung bei Modellbildung an internen Simulationsabläufen → größere semantische Lücke zum Realsystem	Orientierung bei Modellbildung eng am Realsystem

einfachere SW-technische Realisierung	höherer SW-technischer (Verwaltungs-) Aufwand (z.B. Synchronisation, Speicherung von Prozesszuständen)
→ Umsetzung praktisch in jeder gängigen Programmiersprache möglich	Umsetzung benötigt Konzepte wie Threads, Koroutinen, ...

Jedes prozessorientierte Modell kann in ein ereignisorientiertes Modell überführt werden:

Prozessroutinen (z.B. lifeCycle) bestehen meist aus mehreren Ereignissen
 → Aufspaltung in einzelne Ereignisse führt zu ereignisorientierten Modell!

8 Weitere Modellierungsstile

Transaktionsorientierter Modellierungsstil

- abgeleitet aus der Blockdiagrammtechnik der Systemanalyse
- Blöcke: permanent vorhandene, statische Komponenten
- Transaktionen: temporäre, dynamische Elemente
- Transaktionen „durchwandern“ Blöcke, wobei ihr Zustand verändert wird
- Abbildung auf ereignisorientierten Stil leicht möglich

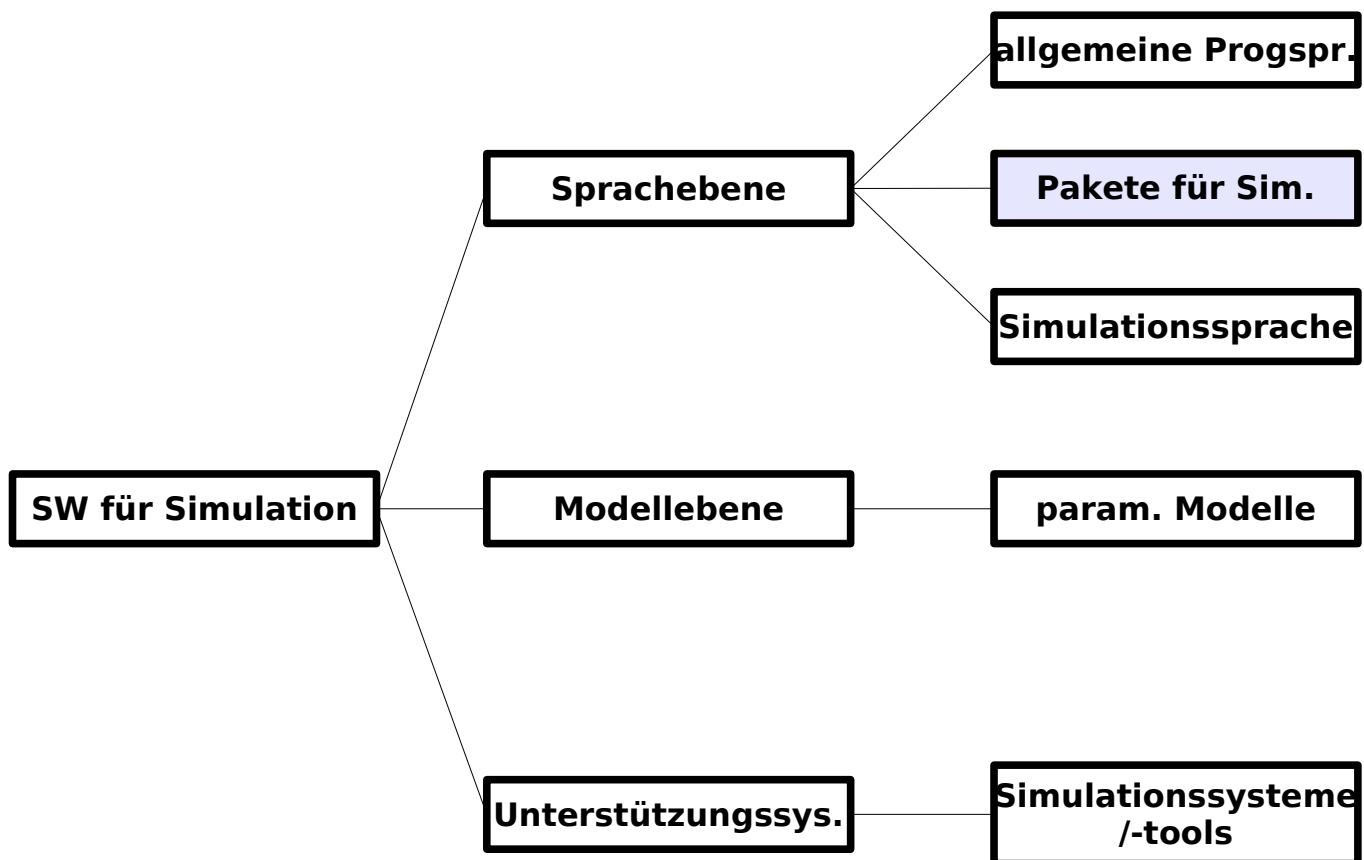
Aktivitätsorientierter Modellierungsstil

- Aktivitäten als Grundelemente, die bei gegebenen Vorbedingungen ausgeführt werden
- rechenintensiv: zu jedem Zeitpunkt sind alle Vorbedingungen aller möglichen Aktivitäten zu prüfen

- keine direkte Abbildung auf ereignisorientierten Stil möglich

9 Software für Simulation

Simulation ist eines der ältesten Anwendungsgebiete der Informatik, daher gab es auch schon früh spezielle Software (*Simulatoren*) dafür. Simulationssoftware kann nach folgenden Aspekten gegliedert werden (nach [2]):



Software auf Sprachebene:

- Simulator in einer (universellen oder speziellen) Programmiersprache realisiert
- Gestaltungsfreiheit wird kaum eingeschränkt
- keine Simulations spezifische Unterstützung (wie z.B. Ablaufsteuerung,

Statistik, ...)

- jedoch meist Konzepte für Pakete, Klassen, ... - damit wird entsprechende Funktionalität bereitgestellt; z.B. *Simulationspakte* mit praktisch beliebiger Erweiterungsmöglichkeit (z.B. DESMO-J)
- Alternative: spezielle *Simulationssprachen*: Konstrukte für Ablaufsteuerung und/oder Modellierung
→ raschere Modellbildung – jedoch geringere Flexibilität und Anwendungsbreite

Software auf Modellebene:

- Simulation für ein spezifisches Realsystem (z.B. Flugsimulator)
- keine Entwicklung in einer Programmiersprache nötig
- kaum Änderungen möglich – nur über Modellparameter
- bei Wiederverwendung ist genau zu prüfen, ob der Simulator das neue System korrekt abbildet

Unterstützungssysteme:

- einheitliche Umgebung für Modellierung und Experimente – *Simulationssysteme* oder *-tools*
- Zyklus Modellbildung – Experiment – Ergebnisanalyse wird unterstützt
- bieten insbesonders: Unterstützung bei der Modellbildung, Experimentierumgebung, Statistiktools
- eventuell auch Teilmodelle zur Wiederverwendung – Modellbanksysteme

Java und Simulation – wie werden Anforderungen erfüllt:

- Zustandsvariablen: primitive Typen, boolean, beliebige Datenstrukturen mittels Klassen

- Simulationsuhr: Kapselung der Simulationszeit, globale Variable
- Temporäre Entitäten/Objekte: mit new erzeugt, vom Garbage Collector entfernt
- Verwaltung zukünftiger Ereignisse: Containerklassen für Ereignisliste
- Warteschlangen: Containerklassen
- Zufallszahlengeneratoren: im Java-API vorhanden
- Statistische Auswertungen: entsprechende Objekte (z.B. mittels Observer Pattern)
- Ergebnisausgabe: Text- oder graphische Ausgabe, Dateien
- unterbrechbare Prozesse: Threads mit entsprechender Funktionalität

VO Einführung Simulation

SS 2020

*Helge Hagenauer
FB Computerwissenschaften*

Teil 2

10 Simulation und Statistik: Überblick

1. Die meisten Modelle enthalten Komponenten, die vom *Zufall* abhängen. Entweder eigentlicher Zufall oder zufälliges Verhalten aus Sicht des Systems/Modells werden mittels Verfahren der Wahrscheinlichkeitsrechnung und Statistik modelliert
→ **stochastische Modelle**
2. Ergebnisse von Simulationen bestehen praktisch immer aus großen Datenmengen (z.B. Wartezeiten aller Kunden, mehrere Simulationsläufe) – mittels Verfahren der Statistik auszuwerten

Dies sind die wichtigsten Aspekte, warum Statistik (u. Wahrscheinlichkeitsrechnung) ein wichtiger Bestandteil von Simulation ist.

Unter dem ersten Punkt (oder davon unabhängig) ist auch noch die Sammlung von Daten und deren Aufbereitung für die Simulation zu nennen.

Einige wichtige Grundbegriffe:

Experiment

Prozess, dessen Ergebnisse nicht mit Bestimmtheit vorhersagbar sind.

Grundmenge (sample space)

Menge aller möglichen Ergebnisse eines Experiments.
Bezeichnung meist: S

Zufallsvariable (random variable)

Funktion, die jedem Element aus S eine reelle Zahl zuordnet.
Bezeichnung meist: X, Y, Z, \dots

Verteilungsfunktion (distribution function)

Die Verteilungsfunktion $F(x)$ einer Zufallsvariablen X ist definiert als

$$F(x) = P(X \leq x)$$

wobei $P(X \leq x)$ ist die Wahrscheinlichkeit von $X \leq x$

Es gelten folgende Eigenschaften:

$$0 \leq F(x) \leq 1 \quad \forall x$$

$F(x)$ ist monoton wachsend

$$\lim_{x \rightarrow \infty} F(x) = 1 \quad \text{und} \quad \lim_{x \rightarrow -\infty} F(x) = 0$$

Diskrete Zufallsvariable

Eine Zufallsvariable X heißt diskret, wenn sie nur abzählbar viele Werte x_1, x_2, \dots annehmen kann. Es gilt

$$p(x_i) = P(X=x_i) \quad \text{Wahrscheinlichkeit, dass } X=x_i$$

$$\sum_{i=1}^{\infty} p(x_i) = 1$$

$$F(x) = \sum_{x_i \leq x} p(x_i) \quad \forall x \quad \text{Verteilungsfunktion von } X$$

Stetige Zufallsvariable

Eine Zufallsvariable X heißt stetig, wenn sie eine nichtabzählbare Anzahl von Werten annehmen kann.

Weiters gibt es eine nicht negative Funktion $f(x)$ (**Dichtefunktion**), sodass für beliebige Mengen B von reellen Zahlen gilt:

$$P(X \in B) = \int_B f(x) dx \quad \text{und} \quad \int_{-\infty}^{\infty} f(x) dx = 1$$

$$F(x) = P(X \in [-\infty, x]) = \int_{-\infty}^x f(y) dy$$

Unabhängige Zufallsvariablen

Seien X und Y diskrete Zufallsvariablen, dann ist

$$p(x, y) = P(X=x, Y=y) \quad \forall x, y$$

$$p_X(x) = \sum_y p(x, y) \quad \text{und} \quad p_Y(y) = \sum_x p(x, y)$$

X und Y heißen **unabhängig**, wenn gilt

$$p(x, y) = p_X(x)p_Y(y)$$

Intuitiv bedeutet dies: ist der Wert einer Zufallsvariablen bekannt und kann nicht auf die Verteilung der anderen Zufallsvariablen geschlossen werden, sind sie unabhängig.

Analog gilt dies für stetige Zufallsvariablen unter Verwendung von f . Verallgemeinerung auf n Zufallsvariablen X_1, X_2, \dots, X_n ist trivial.

Erwartungswert, Median

Der **Erwartungswert (expectation)** $E(X)$ (oder μ) einer Zufallsvariablen X ist definiert als

$$E(X) = \sum_{j=1}^{\infty} x_j p_X(x_j) \quad \text{wenn } X \text{ diskret}$$

$$E(X) = \int_{-\infty}^{\infty} x f_X(x) dx \quad \text{wenn } X \text{ stetig}$$

Der **Median** $x_{0.5}$ der Zufallsvariablen X ist der kleinste Wert von x sodass $F_X(x) \geq 0.5$

Erwartungswert und Median sind „Mittelwerte“. Median ist oft „besser“, wenn X sehr große oder sehr kleine Werte annehmen kann!

Varianz, Standardabweichung

Für die **Varianz** $V(X)$ (oder $Var(X)$ oder σ^2) der Zufallsvariablen X gilt

$$V(X) = E[(X - E(X))^2] = E(X^2) - [E(X)]^2$$

Die **Standardabweichung** σ der Zufallsvariablen X ist definiert als

$$\sigma = \sqrt{V(X)}$$

Beides sind Maße für die Streuung der Werte einer Zufallsvariablen um ihren

Erwartungswert – je größer die Varianz, desto wahrscheinlicher treten Werte mit großer Differenz zum Erwartungswert auf!

11 Zufallszahlen

Simulationsmodelle enthalten meist Komponenten, die aus Sicht des Modells mit Zufallswerten arbeiten – **stochastische Modelle**.

Gründe für „zufällig“ können sein:

- eigentlicher Zufall (z.B. radioaktiver Zerfall)
- zufällig aus Sicht des Modells
 - externe Einflüsse, Ereignisse (z.B. Zwischenankunftszeiten von Kunden)
 - komplexes Verhalten - Details werden nicht modelliert (z.B. Dauer der Bedienzeit am Schalter)

Stochastische Modelle benötigen also Werte (Zahlen), die den „Zufall“ repräsentieren. Prinzipielle Erzeugungsmöglichkeiten sind

- Verwendung eines physikalischen Prozesses mit brauchbaren Zufallseigenschaften: Problem ist die Kopplung mit dem Simulationslauf (Echtzeit, über Dateien, ...) - meist aufwändig, langsam
- Verwendung von Berechnungen – grundsätzlich deterministisch!
Vorteil: schnell, billig, reproduzierbar
Nachteil: nicht wirklich zufällig – benötigen daher entsprechende Gütekriterien

→ **Pseudozufallszahlen**

Jedoch laut allgemeiner Fachmeinung: sorgfältig erstellte Methoden – Zufallszahlengeneratoren – erzeugen Zahlen, die statistische Tests bestehen und somit „Kriterien der Zufälligkeit“ erfüllen.

Die Qualität der Zufallszahlen trägt wesentlich zur Güte von Simulationsergebnissen bei!

Ein **Zufallszahlengenerator** liefert Werte aus einem festgelegten Bereich (z.B. Intervall), wobei jeder mögliche Wert mit einer bestimmten Wahrscheinlichkeit auftreten kann.

Ein Zufallszahlengenerator erzeugt i.A. gleichverteilte Werte im Intervall $(0,1)$.

Gewünschte Eigenschaften eines „guten“ Zufallszahlengenerators sind:

- erzeugte Zufallszahlen sollen scheinbar gleichmäßig in $(0, 1)$ verteilt sein und es soll keine Beziehung (Korrelation) untereinander ersichtlich sein
- schnelle Berechnung und wenig Speicherbedarf
- Reproduzierbarkeit von Zufallszahlenströmen für
 - Debugging, Verifikation
 - Wiederholbarkeit von Experimenten unter geänderten Bedingungen oder Einstellungen
- Erzeugung verschiedener Ströme von Zufallszahlen: jeder Komponente soll ein eigener Zufallszahlengenerator zugeordnet werden – erleichtert Reproduzierbarkeit bzw. Vergleichbarkeit und auch aus theoretischen Gründen (siehe Literatur)!

D.Knuth meinte: „Random numbers should not be generated with a method chosen at random. Some Theory should be used.“

Allgemeiner Ansatz eines Zufallszahlengenerators

Es wird eine Folge natürlicher Zahlen z_1, z_2, \dots aus dem Bereich $\{0, m-1\}$ erzeugt und diese dann mittels Division durch m auf das Intervall $[0, 1]$ abgebildet.

Meist verwendet wird der **linear (gemischt) kongruente Generator**

$$z_{i+1} = (a \cdot z_i + c) \bmod m$$

mit m (Modulus), a (multiplikative Konstante), c (additive Konstante) und z_0

(Startwert, Seed).

Ist $c=0$ dann heißt er auch **multiplikativer (rein) kongruenter Generator**.

Güte des Verfahrens ist abhängig von den Parametern a, c, m ; eventuell auch Einschränkungen bzgl. z_0 ; günstig für interne Darstellung ist $m = 2^n - 1$;

Z.B. ist das Problem der Periodizität nicht zu unterschätzen (insbes. bei „selbst gebauten“ Generatoren): die Wahrscheinlichkeit die Periodenlänge 1 zu erhalten (bel. Generator und Startwert) beträgt für $m=10^{10}$ etwa 1:80000!

Für Zufallszahlengeneratoren ist also wünschenswert:

- Generator soll möglichst große Periodenlänge besitzen, da diese den „nutzbaren“ Teil der Zahlenfolge bestimmt
(maximale Länge durch m begrenzt $\rightarrow m$ groß wählen)
- möglichst alle Werte zwischen 0 und $m-1$ sollen genutzt werden
 \rightarrow garantiert bessere Gleichmäßigkeit der Verteilung

Dazu einige Beispiele (Beweise siehe entsprechende Literatur):

- linear kongruenter Generator erzeugt jeden Wert in $\{0, \dots, m-1\}$ (Startwert beliebig), wenn
 - c und m relativ prim (d.h. keine gemeinsamen Teiler außer 1)
 - $a - 1$ ist Vielfaches von jedem Primfaktor von m
 - $a - 1$ ist Vielfaches von 4 falls 4 Teiler von m
- multiplikativ kongruenter Generator erzeugt jeden Wert in $\{1, \dots, m-1\}$ (Startwert beliebig), wenn

m Primzahl

$a^k - 1$ ist Vielfaches von m für $k = m - 1$, nicht aber für $k < m - 1$

- multiplikativ kongruenter Generator mit $m = 2^n$ hat Periodenlänge $m/4$, wenn
 - $a - 1$ ist Vielfaches von 4
 - z_0 ungerade

Ein warnendes Beispiel: multiplikativ kongruenter Generator mit $m=2^n$ und obigen Eigenschaften liefert immer ungerade $z_i \rightarrow$ niedere Stelle soll nicht für weitere Verarbeitung genutzt werden!

Trotzdem noch viel Freiheit wie etwa Wahl des Multiplikators (jedoch stochastische Eigenschaften beachten) – statistische Tests verwenden (siehe Literatur).

Beispiel eines Generators: multiplikativ kongruenter Generator (aus S.K.Park, K.W.Miller: „Random Number Generators – Good Ones are Hard to Find“, Comm. of the ACM, Vol 31 Nr 10, Oct. 1988)

$$a = 7^5 = 16807 \quad m = 2^{31} - 1 = 2147483647$$

volle Periode, Startwert beliebig aus $\{1, \dots, m\}$;

Achtung auf Overflow – Abhilfe mit folgender Identität (Schrage 1979), alles ganzzahlige Operationen:

$$(a \cdot z) \bmod m = g(z) + m \cdot h(z)$$

mit

$$\begin{aligned} g(x) &= a \cdot (x \bmod q) - r \cdot (x/q) \\ h(x) &= (x/q) - (a \cdot x/m) \quad q = m/a \quad r = m \bmod a \end{aligned}$$

Zufallszahlen beliebiger Verteilungen

Ein allgemein anerkanntes Verfahren zur Erzeugung von Zufallszahlen:

1. ermittle Zufallszahl u_i für gleichverteilte Zufallsvariable U im Intervall $(0,1)$

2. berechne daraus Zufallszahl x_i für Zufallsvariable X mit
Verteilungsfunktion F_X mittels $x_i = F_X^{-1}(u_i)$

F_X^{-1} ist die Umkehrfunktion (Inverse) von F und muss existieren

Zur Erinnerung:

für die Verteilungsfunktion F_X einer Zufallsvariablen X gilt $F_X(x) = P(X \leq x)$,
 F_X monoton wachsend, Dichtefunktion f_X : $F_X(x) = \int_{-\infty}^x f_X(y) dy$

Sei U eine gleichverteilte Zufallsvariable in $(0,1)$: $f_U(u) = 1$, $F_U(u) = u$ für

$u \in (0,1)$;
existiert zu F_X die Umkehrfunktion F_X^{-1} , dann gilt

$$F_X(x) = F_U(F_X(x)) = P[U \leq F_X(x)] = P[F_X^{-1}(U) \leq x]$$

Z.B. im Falle der Exponentialverteilung $F_X(x) = 1 - e^{-x/\mu}$ für $x \geq 0$ ergibt sich
 $x = F_X^{-1}(u) = -\mu \ln(1-u)$

Liegt F_X^{-1} nicht in geschlossener Form vor, dann muss approximiert werden.

Wichtige Verteilungen

Diskrete Verteilungen:

- **Bernoulliverteilung**

$$\begin{aligned} p(0) &= 1-w & p(1) &= w \\ E(X) &= w \\ V(X) &= w(1-w) \end{aligned}$$

Zufallsereignis mit 2 möglichen Ausgängen; Basis für weitere diskrete Verteilungen;

- **Diskrete Gleichverteilung**

$$\begin{aligned} p(x) &= \frac{1}{N} & \text{wenn } 1 \leq x \leq N \\ E(X) &= \frac{1}{2}(N+1) \\ V(X) &= \frac{1}{12}(N^2 - 1) \end{aligned}$$

Gleich wahrscheinliche verschiedene Ereignisse mit wenig weiterer Information.

- **Binomialverteilung**

$$\begin{aligned} p(x) &= \binom{N}{x} w^x (1-w)^{N-x} & \text{wenn } 0 \leq x \leq n \\ E(X) &= Nw \\ V(X) &= Nw(1-w) \end{aligned}$$

Anzahl der Erfolge in N unabhängigen Bernoulliexperimenten (w Erfolgswahrscheinlichkeit im Einzelexperiment); z.B. Anzahl defekter Werkstücke

- **Geometrische Verteilung**

$$p(x) = w(1-w)^{x-1} \quad \text{wenn } x \in [1, 2, \dots]$$

$$E(X) = \frac{1-w}{w}$$

$$V(X) = \frac{1-w}{w^2}$$

Anzahl der Versuche bis zum ersten Erfolg in unabhängigen Bernoulliexperimenten (w Erfolgswahrscheinlichkeit im Einzelexperiment); z.B. Anzahl der untersuchten Werkstücke bis zum ersten schadhaften

- **Poissonverteilung**

$$p(x) = \frac{e^{-\alpha} \alpha^x}{x!} \quad \text{wenn } x \in [0, 1, \dots]$$

$$E(X) = \alpha$$

$$V(X) = \alpha$$

Anzahl unabhängiger Ereignisse in einem fixen Zeitintervall (oder fixem Raum); z.B. Anzahl der Kundenankünfte in einer Stunde, Anzahl der fehlerhaften Stellen auf 50m² Blech

Stetige Verteilungen

- **Gleichverteilung (Rechtecksverteilung)**

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{wenn } a \leq x \leq b \\ 0 & \text{sonst} \end{cases}$$

$$E(X) = \frac{a+b}{2}$$

$$V(X) = \frac{(b-a)^2}{12}$$

zufällige Werte im Bereich a bis b mit wenig weiterer Information

- **Exponentialverteilung**

$$f(x) = \begin{cases} \frac{1}{\mu} e^{-x/\mu} & \text{wenn } x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

$$F(x) = \begin{cases} 1 - e^{-x/\mu} & \text{wenn } x \geq 0 \\ 0 & \text{sonst} \end{cases}$$

$$E(X) = \mu$$

$$V(X) = \mu^2$$

z.B. für Zwischenankunftszeiten, Auftragseingänge ohne Intervallgrenzen; stetiges Gegenstück zur Geometrischen Verteilung

- **Normalverteilung**

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$F(X)$ keine geschlossene Formel

$$E(X) = \mu$$

$$V(X) = \sigma^2$$

z.B. Anzahl der Produktionsfehler in einem Zeitintervall; allgemein für Quantitäten die aus einer großen Anzahl anderer Quantitäten durch Summierung entstehen (zentraler Grenzwertsatz)

12 Eingabeverteilungen

Zufälliges Verhalten von Komponenten ist praktisch in jedem System zu beobachten, wie z.B.

- Computer: Zwischenankunftszeiten von Jobs, Typ eines Jobs, Ausführungsanfordernisse eines Jobs, ...
- Kommunikation: Zwischenankunftszeiten von Nachrichten, Typ und Länge einer Nachricht
- Fertigung: Bearbeitungszeiten, Betriebsdauer einer Maschine bis zum Ausfall, Reparaturzeiten, ...

→ Wahl der „passenden“ **Eingabeverteilung** ist wichtig

(sonst u.U. grobe Ungenauigkeiten oder nicht der Realität entsprechende Abweichungen in den Ergebnissen!)

Anmerkung: es sollte die Eingabeverteilung nicht durch ihren Erwartungswert (oder Median) ersetzt werden - Gefahr der Ergebnisverfälschung ist sehr groß!

Im Prinzip gibt es 2 Arten von Verteilungen

- **theoretische** Verteilungen (z.B. Gleich-, Exponential-,

Normalverteilung)

- **empirische** Verteilungen: werden auf der Basis von sogenannten Realdaten (Beobachtungen, Informationen über das Realsystem) erstellt

Empirische Verteilungen

Liegen brauchbare Realdaten X_1, X_2, \dots, X_n vor, so kann eine *empirische* Verteilung folgendermaßen erstellt werden:

- aufsteigende Sortierung der Realdaten sodass $X_{(1)} \leq X_{(2)} \leq \dots \leq X_{(n)}$
- Verteilungsfunktion ermitteln

$$F(x) = \begin{cases} 0 & x < X_{(1)} \\ \frac{i-1}{n-1} + \frac{x - X_{(i)}}{(X_{(i+1)} - X_{(i)})} & X_{(i)} \leq x < X_{(i+1)} \\ 1 & X_{(n)} \leq x \end{cases}$$

Nachteile davon sind

keine Werte $< X_{(1)}$ und $> X_{(n)}$ erzeugbar und
i.A. ist der Erwartungswert von $F(x)$ nicht identisch mit jenem der Realdaten!

Vorgangsweise ist ähnlich, wenn die Realdaten in Form eines Histogramms vorliegen (keine Einzelwerte).

Diskrete empirische Verteilungen: für jeden möglichen Wert x ist $p(x)$ der Anteil jener X_i , die gleich x sind.

Vorteile theoretischer Verteilungen

- empirische Verteilungen können Irregularitäten aufweisen (besonders bei kleiner Anzahl von Realdaten)
 - theoretische Verteilungen „glätten“
- empirische Verteilungen geben den Beobachtungszeitraum der Realdaten wieder
 - theoretische Verteilungen mischen und erweitern dies durch Beachtung von zusätzlichem Expertenwissen

- übliche Verfahren für empirische Verteilungen erlauben keine Werte außerhalb des Bereichs der Realdaten – „außergewöhnliche“ Ereignisse werden nicht erfasst
→ Verfälschung der Ergebnisse
- theoretische Verteilungen ergeben eine kompakte Darstellung von Datenmengen (keine Verwaltung von Arrays oder ähnlichem nötig)

Findet sich keine passende theoretische Verteilung, dann soll ein empirische Verteilung verwendet werden.

Einbeziehung von Realdaten

Realdaten werden meist durch Beobachtung des Realsystems erhalten. Eventuell ist auch eine Ableitung aus physikalischen oder technologischen Aspekten möglich.

Liegen Realdaten zu einer Eingabe-Zufallsvariablen vor, dann können diese in verschiedener Form verwendet werden:

1. direkte Verwendung genau dieser Werte
→ trace-driven simulation
2. zur Erstellung einer empirischen Verteilung
3. mittels statistischer Methoden werden theoretische Verteilungen an die Realdaten „angepasst“

Dazu ist zu bemerken:

- Nachteil von 1.: nur beobachtete Werte werden verwendet; es gibt selten genug Daten für mehrere verschiedene Simulationsläufe
- 1. ist sinnvoll für Modellvalidierung
- existiert eine vernünftig passende theoretische Verteilung, dann ist i.A. diese einer empirischen Verteilung vorzuziehen

Vorgangsweise bei der Auswahl von Eingabeverteilungen

Eingabeverteilungen können nach folgendem 3-stufigen Verfahren festgelegt werden:

1. Auswahl einer Verteilung (-sfamilie)

Informationen über das System werden benutzt um theoretische Verteilungen auszuwählen (oder auszuschließen): z.B. keine negativen Werte, Werte in begrenztem Intervall

Realdaten können verwendet werden

- * zum Vergleich statistischer Kenngrößen mit jenen der Verteilung (z.B. Min/Max, Mittelwert oder Erwartungswert, Varianz)
- * zur Erstellung von Histogrammen für graphische Darstellung (Intervalllängen → Anzahl der Klassen überlegen: Richtwert \sqrt{n} , n Anzahl der Beobachtungen)
- * für Symmetrievergleiche (z.B. verzerrt nach einer Seite)

2. Parameter schätzen

Um den Wert eines Parameters abschätzen zu können, werden am besten wieder Realdaten X_1, X_2, \dots, X_n verwendet (es können die selben sein wie beim Punkt zuvor).

Als erste Abschätzung können z.b. Mittelwert oder Varianz der Realdaten herangezogen werden.

Z.B. kann die *Maximum-likelihood-Methode* angewendet werden um einen Parameter r zu schätzen:

sei im diskreten Fall $p_r(x)$ die Wahrscheinlichkeitsfunktion, dann wird die likelihood-Funktion $L(r)$ definiert als

$$L(r) = p_r(X_1)p_r(X_2)\dots p_r(X_n)$$

$L(r)$ ist die Wahrscheinlichkeit, dass die gegebenen Daten unter Verwendung von r auftreten

→ daher ist der Wert r^* zu bestimmen, sodass $L(r^*)$ maximal wird – gennant *maximum-likelihood-Schätzer (MLE)*

Stetige Verteilungen: analoge Definition

3. Güte der Charakteristik der ausgewählten Verteilung untersuchen

I.A. gibt keine Verteilung „exakt“ die Realität wieder – daher Suche nach „am besten“ passender Verteilung.

Heuristische Methoden: Frequenzvergleiche mittels Histogrammen (stetig) oder Liniendiagrammen (diskret), graphische Darstellung von Wahrscheinlichkeiten (Vergleich der Verteilungsfunktionen)

Anpassungstests: z.B. χ^2 -Test (testen der Hypothese, dass die Realdaten einer bestimmten Verteilung folgen)

Wenn keine Realdaten zur Verfügung stehen, dann müssen Experten befragt werden (wahrscheinlichster Wert, wahrscheinliches Max/Min, ...)

Beispiel: Ankunftsprozess

Gegeben sind zufällige Zeitpunkte $0 = t_1 \leq t_2 \leq \dots$ für Ereignisse (z.B. Ankunftszeitpunkte von Kunden). Sei $N(t)$ die Anzahl der Ereignisse bis Zeitpunkt t ; dann wird $\{N(t), t \geq 0\}$ ein **Ankunftsprozess** mit **Zwischenankunftszeiten** $A_i = t_i - t_{i-1}$ genannt.

Dies ist ein *Poissonprozess* mit mittlerer Ankunftsrate λ , wenn folgende Punkte gelten:

1. pro Zeitpunkt maximal eine Ankunft (z.B. Kunden kommen einzeln an)
2. $N(t+s) - N(t)$ ist unabhängig von $\{N(u), 0 \leq u \leq t\} \forall s$: die Anzahl der Ankünfte in nicht überlappenden Zeitintervallen sind unabhängige Zufallsvariablen
3. Verteilung von $N(t+s) - N(t)$ ist unabhängig von $t \forall t, s \geq 0$: die Anzahl der Ankünfte in $[t, t+s]$ ist nur von s abhängig, nicht vom Startpunkt t (stationärer Poissonprozess) → Ankünfte sind rein zufällig

Dann gilt: die Anzahl der Ankünfte in einem Intervall der Länge s ist eine Poisson-Zufallsvariable mit Parameter λs :

$$P[N(t+s) - N(t) = n] = \frac{e^{-\lambda s} (\lambda s)^n}{n!} \quad \text{für } n = 0, 1, 2, \dots$$

Weiters gilt für die Zwischenankunftszeiten A_1, A_2, \dots : sie sind unabhängige identische exponential-verteilte Zufallsvariablen mit Erwartungswert $1/\lambda$.

Nichtstationärer Poissonprozess: Ankunftsrate hängt oft von der Zeit ab

(z.B. Stoßzeiten) $\rightarrow \lambda(t)$;
es gilt weiterhin 1. und 2. von oben, zusätzlich wird definiert $\Lambda(t)=E(N(t))$,
differenzierbar, dann ist $\lambda(t)=\frac{d}{dt}\Lambda(t)$.

Achtung: Zwischenankunftszeiten sind nicht identisch verteilt!

Vorgangsweise bei gleichzeitigen Ereignissen (z.B. gleichzeitige Ankunft mehrerer Kunden vor einem Konzert): Punkt 1. von oben ist nicht erfüllt – daher folgende Vorgangsweise möglich:

- $N(t)$ ist nun die Anzahl der Gruppen, die bis t angekommen sind
- $\{N(t), t \geq 0\}$ wird als Poissonprozess modelliert
- ermitteln einer passenden diskreten Verteilung für die Gruppengrößen

13 Ergebnisanalyse

Simulationen erzeugen meist große Mengen von Daten – diese müssen klar und nutzbringend dargestellt werden.

Meist wird wenig Aufmerksamkeit für fundierte Ergebnisanalyse aufgewendet (im Vergleich zur Modellbildung)!

Bei der Ergebnisanalyse ist zu bedenken:

Simulationen bilden Werte von Zufallsvariablen (Zufallszahlen) in Simulationsergebnisse ab – es können große Varianzen auftreten!

D.h. ein einziger (oder zu kurzer) Simulationslauf kann große Abweichungen von den wirklichen Charakteristiken haben.

→ ein (oder ein zu kurzer) Simulationslauf ist nicht ausreichend um seine Ergebnisse als Modell kennzeichnende Charakteristika anzusehen!

Es sind also mehrere Simulationsläufe (oder ein ausreichend langer

Simulationslauf) durchzuführen und die Ergebnisse mit statistischen Methoden zu analysieren.

Jeder Simulationslauf liefert einen Schätzwert für einen beobachteten Ergebnisparameter

→ für ein repräsentatives Ergebnis müssen also genügend viele Simulationsläufe durchgeführt werden (desto mehr Werte vorliegen, desto „besser“ das Ergebnis – allgemeines statistisches Prinzip).

Stochastischer Prozess: Menge von Zufallsvariablen Y_1, Y_2, \dots über der selben Grundmenge und geordnet nach der Zeit.

Z.B.: Wartezeiten der einzelnen Kunden

Problem: i.A. sind die Y_i nicht identisch und unabhängig verteilt – dies ist aber meist Voraussetzung für die Anwendung statistischer Standardmethoden
→ Alternativen überlegen!

Eine mögliche Variante dazu: sei $y_{11}, y_{12}, \dots, y_{1m}$ das Ergebnis des ersten und $y_{n1}, y_{n2}, \dots, y_{nm}$ jenes des n -ten Simulationslaufs
→ dann gilt $y_{1i}, y_{2i}, \dots, y_{ni}$ sind unabhängige Beobachtungen von Y_i – dies kann als Basis für Schätzwert herangezogen werden.

Stationäre stochastische Prozesse

Meist gilt das Interesse einer Simulation so genannten *stationären stochastischen Prozessen (stationären Phasen)* – d.h. die Wahrscheinlichkeit für einen Wert von Y_i hängt nicht von der Zeit (und eventuellen Startbedingungen) ab.

Anders formuliert:

seien $F_i(y|I)$ die *transienten* Verteilungen der Zufallsvariablen Y_i eines stochastischen Prozesses unter der Startbedingung I – d.h. sie variieren i.A. für jedes i (also mit der Zeit) und verschiedenen I .

wenn $F_i(y|I) \rightarrow F(y)$ für $i \rightarrow \infty$, y und I beliebig , dann heißt $F(y)$ eine *stationäre Verteilung*.

Dabei ist zu beachten:

- Stationarität heißt: die Verteilung bleibt die selbe – die Y_i nehmen

nicht alle den selben Wert an (es ist kein konstanter Zustand!)

- theoretisch wird dies erst für $i \rightarrow \infty$ erreicht – praktisch sind jedoch ab einem bestimmten Index die Verteilungen approximativ gleich!
- die Phase bis zur Erreichung der Stationärität wird als „Warm-up“ bezeichnet (Ergebnisse daraus sind meist nicht relevant - Ausnahmen möglich)

Simulation mit Abbruchbedingung

In Bezug auf die Ergebnisanalyse werden 2 Simulationsarten unterschieden – zuerst wird jene mit Abbruchbedingung betrachtet.

Mit Abbruchbedingung bedeutet: es existiert ein „natürliches“ Ereignis E , welches die Simulation beendet (und damit die Länge eines Simulationslaufs bestimmt).

D.h. E gibt meist jenen Zeitpunkt an, nachdem keine nützlichen (Teil-)Ergebnisse mehr entstehen.

Z.B.: Schalter hat von 8:00 bis 12:00 Uhr geöffnet – also E : 4 Stunden Simulationszeit abgelaufen (und keine Kunden mehr vorhanden).

Die Vorgangsweise könnte hier so aussehen (betrachten nur einen Messwert δ):

- Durchführung von n Simulationsläufen mit jeweils der selben Startbedingung und verschiedenen Zufallszahlenströmen (verschiedene Startwerte)
- ist Y_j die Zufallsvariable des j -ten Simulationslaufs des Messwertes δ , dann sind die Y_j identisch und unabhängig verteilt!
 - z.B. mittlere Wartezeit in einem Simulationslauf
- Verwendung *erwartungstreuer (unbiased)* Schätzer für Messwert δ : der erwartete Wert des Schätzers entspricht jenem des Messwerts
 - Mittelwert der Stichprobe

$$\bar{Y}(n) = \frac{1}{n} \sum_{i=1}^n Y_i$$

ist erwartungstreuer Schätzer für δ , d.h. $E[\bar{Y}] = \delta$

- Varianz der Stichprobe

$$S^2(n) = \frac{1}{n-1} \sum_{i=1}^n [Y_i - \bar{Y}(n)]^2$$

ist erwartungstreuer Schätzer für σ^2

Weiters von Interesse ist: wie viele Simulationsläufe sind nötig, um eine vorgegebene Genauigkeit zu erhalten (bzw. wie „gut“ ist der erhaltene Wert)?

Dazu können *Konfidenzintervalle* benutzt werden: es werden Werte U_1, U_2 gesucht, sodass gilt

$$P(U_1 \leq \delta \leq U_2) = 1 - \alpha$$

D.h. mit Wahrscheinlichkeit α liegt der Wert nicht im ermittelten Intervall.

Meist ist das Konfidenzintervall symmetrisch: $U_1 = U - d, U_2 = U + d$

Für eine Zufallsvariable mit unbekannter Varianz kann beispielsweise nachstehendes Verfahren zur Berechnung von Konfidenzintervallen für den Erwartungswert angewendet werden:

- für $Y = Y_1 + \dots + Y_n$ (alle unabhängig, Erwartungswert μ , Varianz σ^2) wird angenommen
 - Y hat angenommene Erwartungswert $n\mu$ und Varianz $n\sigma^2$
 - Y ist normalverteilt wenn Y_1, \dots, Y_n normalverteilt
 - sind Y_1, \dots, Y_n nicht normalverteilt, dann ist Y für große n trotzdem normalverteilt (zentraler Grenzwertsatz)
- für große n gilt $\bar{Y}(n) = \frac{1}{n} Y$ ist normalverteilt mit $\mu, \sigma^2/n$
- da Varianz σ^2 i.A. unbekannt ist, wird die Stichprobenvarianz S^2 als Schätzer verwendet
dafür ist jedoch die t-Verteilung mit $n-1$ Freiheitsgraden zu nutzen

$$\bar{Y}(n) \pm t_{n-1, 1-\alpha/2} \sqrt{\frac{S^2(n)}{n}}$$

mit $t_{n-1,1-\alpha/2}$ als $1-\alpha/2$ kritischer Wert der t-Verteilung mit $n-1$ Freiheitsgraden (aus Tabelle entnehmen)
→ ergibt $100(1-\alpha)$ -prozentiges Konfidenzintervall

Die Länge eines Konfidenzintervalls hängt ab von

- Varianz σ^2 : kleinere Abweichungen in den Beobachtungen führen zu genaueren Abschätzungen
- Beobachtungsgröße n : je größer n , desto kleiner das Konfidenzintervall
- Konfidenzniveau $1-\alpha$: je höher das Konfidenzniveau sein soll, desto größer wird das Konfidenzintervall

Weitere Infos dazu siehe Literatur.

Wichtig: sorgfältige Auswahl der Startbedingung!!

Z.B. soll die mittlere Wartezeit am Schalter in der Stoßzeit (etwa 11:00 bis 12:00 Uhr) ermittelt werden:

- starten mit keinem Kunden entspricht i.A. nicht der Realität
→ falscher Einfluss auf Simulationsergebnisse
- mögliche Abhilfen:
 - gesamte Öffnungszeit simulieren, jedoch für das Ergebnis nur jene Kunden beachten, die zwischen 11:00 und 12:00 Uhr zum Schalter kommen
 - reale Daten sammeln, wie viele Kunden um 11:00 Uhr am Schalter warten (z.B. an verschiedenen Tagen)
→ passende Verteilung ermitteln und Simulation um 11:00 Uhr mit zufällig gewählter Anzahl vorhandener Kunden starten

Simulation ohne Abbruchbedingung

Hier existiert kein „natürliches“ Abbruchereignis.

Aussagen über Stationärität spielen dabei oft eine große Rolle.

Z.B.: für ein Fertigungssystem soll die durchschnittliche Anzahl der erzeugten Einheiten pro Stunde auf lange Sicht ermittelt werden.

Es können erst jene Ergebnisse verwendet werden, die nach der Warm-up-Phase entstehen, d.h. wenn l das Ende davon markiert gilt etwa für den Schätzer

$$\bar{Y}(n, l) = \frac{1}{n-l} \sum_{i=l+1}^n Y_i$$

Dann ergeben sich folgende Möglichkeiten:

- selbes Vorgehen wie im Fall mit Abbruchbedingung – jedoch werden nur jene Werte verwendet, die nach der Warm-up-Phase entstehen
- ein langer Simulationslauf, wobei nach der Warm-up-Phase passende Methoden zur Abschätzung eingesetzt werden

Somit ist folgendes Problem zu lösen:

wie wird das Ende der Warm-up-Phase abgeschätzt oder bestimmt?

Bestimmung der Warm-up-Phase

Wichtig ist eine möglichst genaue Bestimmung der Länge der Warm-up-Phase. Ist diese

- zu lange gewählt:
 - Verschwendung von Daten, Laufzeit länger
 - Varianz wird größer, daher auch die Laufzeit länger wenn die Anforderungen an die Genauigkeit gleich bleiben
- zu kurz gewählt:
 - Ergebnisse sind i.A. verzerrt

Eine „gute“ Wahl der Startbedingung kann die Länge der Warm-up-Phase stark verkürzen.

Es existieren verschiedene Kategorien für Methoden zur Bestimmung der Warm-up-Phase:

- *graphische Methoden*: geeignete Darstellung der relevanten Zeitreihen zur Bestimmung der stationären Phase
- *heuristische Methoden*: beruhen auf einfachen Regeln; meist speziell für bestimmte Anwendungsbereiche
- *statistische Methoden*: statistische Tests zur Bestimmung des Endes der Warm-up-Phase
- unbekannte Startbedingung (z.B. diese soll durch Simulation bestimmt werden): aus Simulationsergebnissen mittels statistischer Tests den Fehler abschätzen, welcher durch falsche Wahl der Startbedingung entstehen kann
- *hybride Methoden*: Kombinationen von obigen Methoden

Beispiele für entsprechende Methoden sind

Regel von Conway

Es liegen die stochastischen Werte y_1, \dots, y_n vor. Für jedes y_i wird getestet, ob es ein neues Minimum oder Maximum der restlichen y_{i+1}, \dots, y_n ist. Wenn ja, dann wird angenommen, es gehört zur Warm-up-Phase.

Crossing the Mean

Es liegen die stochastischen Werte y_1, \dots, y_n vor. Für jedes y_i wird der zugehörige Mittelwert

$$\bar{Y}(i) = \frac{1}{i} \sum_{k=1}^i y_k$$

berechnet. Wenn die entsprechenden Werte y_1, \dots, y_{i-1} diesen eine bestimmte Anzahl mal „überschreiten“ (Richtwert: 3 bis 5), wird der Beginn der stationären Phase angenommen.

Marginaler Varianz-Fehler (Marginal Standard Error)

Es liegen die stochastischen Werte y_1, \dots, y_n vor. Gesucht wird jener Index i (ist dann Endpunkt der Warm-up-Phase), sodass

$$\frac{S^2(n, i)}{n - i} \text{ minimal wird.}$$

Dabei gilt $S^2(n, i) = \frac{1}{n-i-1} \sum_{k=i+1}^n (y_k - \bar{Y}(n, i))^2$

Weiters wird angenommen, dass $i \ll n$ („wesentliche kleiner“)!

D.h. ergibt sich ein Index i , welcher nicht klein genug ist, muss n (die Beobachtungsphase) erhöht werden und darauf nochmals dieser Test angewendet werden.

Batch Mean

Statistisches Verfahren, welches die vorliegenden stochastischen Werte y_1, \dots, y_n in Teile der Größe m unterteilt, also $n=bm$. Berechnet werden

$$\bar{Y}(im, (i-1)m) = \frac{1}{m} \sum_{k=1}^m y_{(i-1)m+k} \quad \text{für } i \in [1, 2, \dots, b] \quad \text{Mittelwert des i-ten Teils}$$

$$S_B^2 = \frac{m}{b-1} \sum_{i=1}^b \left(\bar{Y}(im, (i-1)m) - \frac{1}{b} \sum_{k=1}^b \bar{Y}(km, (k-1)m) \right)^2 \quad \begin{array}{l} \text{Schätzer für die} \\ \text{Varianz} \end{array}$$

Diese so genannten „Batches“ werden in 2 Gruppen (nicht unbedingt gleich groß) aufgeteilt - b' Teile in die erste und $b-b'$ Teile in die zweite. S_{B1}^2 und S_{B2}^2 sind nach obiger Formel die Schätzer für die entsprechenden Varianzen.

Nun kann die Warm-up-Phase mittels dem *F-Test (F-Verteilung)* (Test ob sich die Varianzen wesentlich unterscheiden) bestimmt werden (mit Irrtumswahrscheinlichkeit α):

ist $\frac{S_{B1}^2}{S_{B2}^2} > F_{1-\alpha, b'-1, b-b'-1}$ dann kann die Warm-up-Phase nicht bestimmt werden.

Somit ergibt sich folgende Vorgehensweise: nacheinander 2 Gruppen testen von $b'=1$ bis $b'=n/2$ - der erste Wert, für den

$\frac{S_{B1}^2}{S_{B2}^2} \leq F_{1-\alpha, b'-1, b-b'-1}$ gilt, markiert das Ende der Warm-up-Phase.

14 Validierung, Verifikation

Simulationsmodelle sind oft wichtige Entscheidungshilfen

→ Ergebnisse einer Simulation sollen/müssen möglichst gut dem Realsystem entsprechen.

Existiert kein Realsystem, dann muss das Verhalten des Simulationsmodells „plausibel / stichhaltig“ sein!

Es existieren viele Möglichkeiten wo oder wie Fehler im Simulationsprozess entstehen können, z.B.

- wichtige Komponenten oder Funktionalitäten des Systems werden zu stark vereinfacht oder falsch umgesetzt
- fehlende Daten über das Realsystem können zu ungenauen (oder falschen) Annahmen über Modellparameter führen – auch Experten können irren
- Daten von ungenauen oder ungültigen Approximationen des Realsystems führen zu inkorrekt Modellen
- Fehler bei der Umsetzung des kozessionellen Modells in ein Computermodell (-programm)

Validierung eines Simulationsmodells sollte einer der wichtigsten Aspekte während des Simulationsvorhabens sein (Güte der Modellqualität).

- meist anspruchsvoll, viele Verfahren existieren, aufwändig, ...

Validierung – Verifikation

Validierung:

Vorgang des (rechts)gültig machen (Duden)

ist die Prüfung einer These, eines Plans oder Lösungsansatzes in Bezug auf das zu lösende Problem, die mit der Verifizierung , Falsifizierung oder unklar endet (de.wikipedia.org)

Verifikation:

durch Überprüfen die Richtigkeit bestätigen (Duden)

ist der Nachweis, dass ein vermuteter oder behaupteter Sachverhalt wahr ist (de.wikipedia.org)

Im Kontext mit Simulation erscheinen folgende Definitionen sinnvoll (auch „Test“ wird aus praktischen Gründen noch hinzugenommen):

- **Validierung** eines Modells:
Sicherstellung, dass das Modell unter der gegebenen „Problemdefinition“ (Ziele, Rahmenbedingungen) eine „angemessene“ Repräsentation des Realsystems ist
- **Verifikation** eines Modells:
Sicherstellung, dass ein Modell korrekt dargestellt ist und konsistent von einer Form in eine andere transformiert wird
- **Testen** eines Modells:
ausführen einer Simulation am Computer („Programm ablaufen“) um zu bestätigen, dass es das entsprechende konzeptionelle Modell richtig umsetzt;
(wird oft auch als Teil der Verifikation angesehen)

Validierungsprozess

Alle Aktivitäten im Rahmen eines Simulationsvorhabens, welche zur Modellvalidierung gehören, werden oft als *Validierungsprozess* bezeichnet.

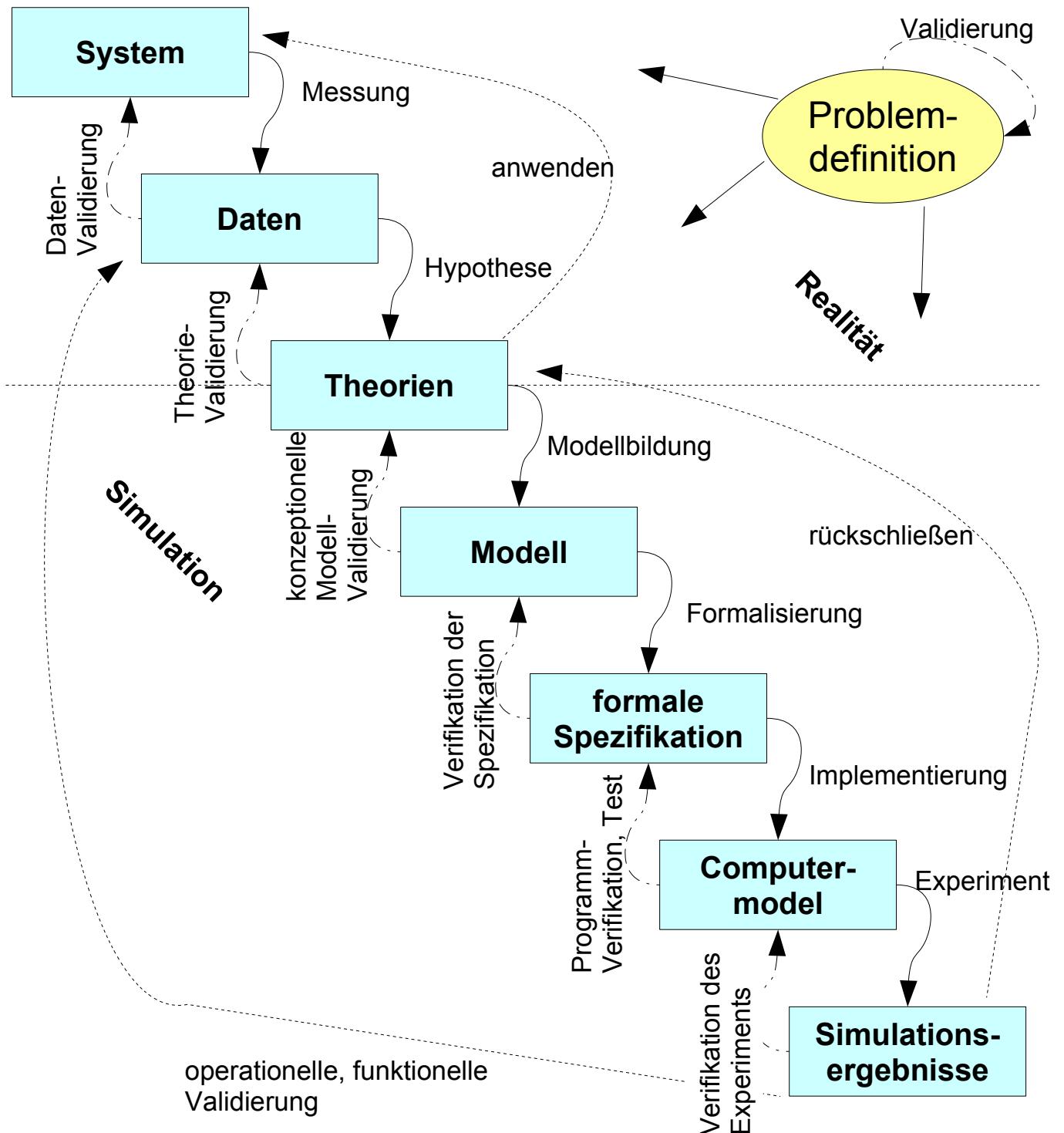
Auch hier sehr wichtig: eine enge Zusammenarbeit verschiedener „Rollen“ (auch wenn die Kommunikation oft schwierig ist):

- **Nutzer** (Konsument) des Simulationsmodells: bestellt Simulation, formuliert Ziele und Rahmenbedingungen; meist gute Kenntnis über Realsystem
- **Designer** des Simulationsmodells: erstellt ein Modell nach der Beschreibung des Nutzers; benötigt Fertigkeit und Kenntnisse der Modellierung (allgemein und für spezielle Bereiche)
- **Implementierer** des Simulationsmodells: erstellen eines Computerprogramms nach den Spezifikationen des Designers

Auch die Validierung wird oft in Phasen eingeteilt, die zumindest enthalten sollen:

- **Konzeptionelle** Validierung: während der Modellierungsphase; das Modell soll eine plausible Repräsentation des Realsystems darstellen
- **Modellverifikation**: während der Implementierung; soll sicherstellen, dass das konzeptionelle Modell richtig in ein Programm umgesetzt wird
- **operationelle** Modellvalidierung: versucht festzustellen, wie nahe das Verhalten des Modells an jenes des Realsystems herankommt

Darstellung des Validierungsprozess (nach B.Page)



Dies bedeutet, dass Validierung während des gesamten Modellierungsvorgangs (oder auch -zirkel) wichtig ist.

Simulationsvorhaben sind stark iterativ, d.h. zu jeder Modellbildungsphase gehört eine entsprechende Validierungsphase – jedoch inkorrekte Ergebnisse können sich auf beliebige frühere Phasen auswirken!

Zu den einzelnen Phasen in der Abbildung:

- Datenvalidierung
Realdaten werden auf ihre „Qualität“ getestet (Daten und Messverfahren untersuchen, ...)
- Theorievalidierung
entwickelte Theorien über das Realsystem werden mit vorliegenden Daten und anderen relevanten Theorien verglichen
- konzeptionelle Modellvalidierung
Vergleich des konzeptionellen Modells mit vorliegenden Daten und anderen relevanten Theorien (Diskussion Nutzer – Designer, ...)
- Verifikation von Spezifikation und Programm
Korrektheit der Transformationen zwischen verschiedenen formalen Modellen und der Implementierung prüfen (Methoden der formalen Verifikation, Test, Review, ...)
- Verifikation des Experiments
vor dem Experimentieren sind Testläufe durchzuführen, um die Plausibilität anhand von vorliegenden Daten zu prüfen (Kalibrierung mittels Modellparameter)

Allgemeine Bemerkungen zur Validierung

- das Ergebnis einer Validierung soll nicht „binär“ gesehen werden – es sind viele Nuancen zwischen korrekt und inkorrekt möglich
- ein Simulationsmodell entsteht unter Beachtung von bestimmten Zielsetzungen – seine Glaubwürdigkeit hat dies zu beachten
- die Glaubwürdigkeit kann nur unter den definierten Zielsetzungen gesehen werden
- erschöpfendes Testen eines Simulationsmodells ist i.A. nicht möglich

Es ist auch zu bedenken, dass Validierung auch unter „ökonomischen“ Gesichtspunkten zu sehen ist – d.h. Wert und Kosten eines Modells steigen nicht linear mit höherem Validierungsaufwand.

15 Simulation kritisch betrachtet

Modellbildung und Simulation als nützliche Instrumente der Systemanalyse wenn

- Simulationsstudie gut geplant und korrekt ausgeführt wird
- Ergebnisse kritisch analysiert werden

Simulation als Unterstützung eines Problemlösungsprozesses

unter vorgegebenen Zielen und Randbedingungen werden Antworten auf konkrete Fragestellungen gesucht.

Simulation kritisch betrachtet – **Pluspunkte**

- Alternativen zu Realexperimenten: wenn diese unmöglich, zu teuer, zu gefährlich, . . .
- Erfassung der Systemkomplexität: isolierte Betrachtung von Komponenten – dann deren Zusammenwirkung
- erhöhtes Systemverständnis (Erkenntnisgewinn): befassen mit dem System und Modellbildung führt zu Erkenntnisgewinn
- Signalwirkung: z.B. Abschätzung der Einwirkung geplanter Eingriffe in ein Realsystem
- Strategiebestimmung: Auswahl alternativer Handlungsvorgänge, Auswirkungen komplexer Vorgänge (Strategien) abschätzen zu können
- Entscheidungshilfen (jedoch keine Entscheidungen selbst!)

Simulation kritisch betrachtet – **Probleme**

- Realitätsferne: Gefahr durch Vereinfachung / Abstraktion
- Modell ist nicht Realsystem: Simulationsdaten nicht als Fakten ansehen!
- mangelnde Transparenz – Ergebnisse sollten möglichst gut

nachvollziehbar sein

- Datenmangel: Fälschung der Ergebnisse durch ungenaue Annahmen
- Fehleranfälligkeit: Risiko von Irrtümern und Fehlentscheidungen im gesamten Modellbildungsprozess; Akkumulation numerischer Fehler
- Computergläubigkeit: Ergebnisse immer kritisch interpretieren

Wann soll Simulation eingesetzt werden?

Es existieren immer wieder Diskussionen, unter welchen Bedingungen Simulation eingesetzt werden soll – und wann nicht.

Ein Überblick (unter Beachtung obiger Argumente), für welche Ziele/Probleme Simulation eingesetzt werden soll (nach Banks et al):

- Simulation ermöglicht das Studieren von und experimentieren mit komplexen Systemen;
gilt auch für den internen Ablauf bzw. komplexer Subsysteme von komplexen Systemen
- veränderte Rahmenbedingungen (informell, organisatorisch, ...) können simuliert werden → Auswirkungen auf Modellverhalten beobachten
- die durch die Entwicklung eines Modell gewonnene Einsicht und das Verständnis können für Verbesserungen im betrachteten Realsystem herangezogen werden
- Änderung von Input-Werten und Beobachtung der Auswirkungen liefert Einsicht, welche Variablen sind wichtig und wie beeinflussen sie sich
- Simulation als pädagogisches Hilfsmittel zur Verstärkung analytischer Lösungsmethodik
- Simulation um mit neuen Designs, Strategien, ... vor der eigentlichen Realisierung zu experimentieren
- Simulation um Ergebnisse von analytischen Lösungen zu verifizieren

- Simulation für Trainingszwecke erlaubt etwas zu Lernen, ohne den eigentlichen Prozess zu stören (Kosten, Gefahren, ... vermeiden)
- Simulation kann mittels Animation visualisiert werden, d.h. Gewinnung eines „realistischeren“ Eindrucks möglich
- viele aktuelle (und zukünftige) Systeme sind so komplex, dass interne Abläufe und Zusammenhänge nur mittels Simulation sinnvoll behandelt werden können

Wann ist Simulation unangebracht?

Auch dazu gibt es einige Diskussionen. Ein Überblick zu möglichen Bedenken (nach Banks et al):

- Simulation ist unnötig, wenn das Problem mit „gesundem Hausverstand“ gelöst werden kann
- Simulation ist unnötig, wenn das Problem analytisch gelöst werden kann
- Simulation ist unnötig, wenn direkte Experimente weniger kosten
- Simulation sollte nicht eingesetzt werden, wenn die Kosten dafür höher sind als die dadurch erwartete Einsparung
- Simulation sollte nicht eingesetzt werden, wenn Ressourcen und Zeit dafür zu gering sind
- liegen keine Daten oder nicht einmal Vermutungen vor, so ist von Simulation abzuraten
- bei übertriebenen Erwartungen des Managements ist von Simulation abzuraten (z.B. zu viel in zu kurzer Zeit)
- bei zu komplexen oder nicht definierbarem Systemverhalten ist von Simulation abzuraten (z.B. menschliches Verhalten)