2 Stand der Technik

In diesem Kapitel sollen grundlegende Kenntnisse aufgeführt werden, die zum besseren Verständnis der Arbeit beitragen. Zunächst gehe ich auf Technologien in der dezentralen Automatisierung ein. Dieser Abschnitt soll eine Einführung in das Anwendungsgebiet des Simulators sein. Nach der Auffassung mehrerer Autoren ist es für den Modellbauer unerlässlich, neben Kenntnissen der Simulationstechnik auch über grundlegendes Wissen im Anwendungsbereich zu verfügen [WCH94, S. 9]. Am Ende des ersten Abschnittes gehe ich auf das ISILEIT–Projekt ein. Die Ergebnisse dieser Arbeit sollen in diesem Projekt verwendet werden.

Ausgehend vom Begriff der Simulation wird in Abschnitt 2.2 zunächst das prinzipielle Vorgehen innerhalb von Simulationsstudien erläutert. Weiterhin werden die zur Unterstützung von durchzuführenden Studien verwendeten Instrumente aufgegriffen. Ein Überblick über unterschiedliche Simulationsmethoden beschließt den Abschnitt.

Ein objektorientierter Ansatz in der Simulation und die Modellierung der zu simulierenden Systeme durch die *UML* beenden das Kapitel. Hierbei soll keine Referenz zur UML angegeben werden, sondern vielmehr auf spezielle Diagrammtypen verwiesen werden. Detaillierte Abhandlungen zur UML finden sich in zahlreichen Quellen z. B [UML99-ol].

2.1 Materialflusssysteme

Zahlreiche produzierende Unternehmen in Deutschland setzen verstärkt auf automatisierte Produktionseinrichtungen. Sie versuchen damit, neuen Anforderungen, die sich aus einem ausgeprägten Käufermarkt und dem starken internationalen Wettbewerb ergeben, nachzukommen [Geh99]. Der Einsatz automatisierter Fertigungstechniken erfordert allerdings hohe Investitionen, die durch die Produktion abgedeckt werden müssen. Grundvoraussetzung beim Einsatz automatisierter Technologien ist die Flexibilität, zumal Industriegüter immer stärkeren Schwankungen ausgesetzt sind [GF98]. Bedingt durch die Forderungen nach innovativen Produkten vor allem im High-Tech-Bereich und die Nachfrage nach individuellen Produkten sinkt die Losgröße der einzelnen Produktvarianten [Ger97, S.1]. Zum einen steigt somit die Forderung nach automatisierten Fertigungsstrukturen, um z. B. die Lohnkosten senken zu können, zum anderen dürfen die eingesetzten Strukturen nicht statisch sein, damit die geänderten Anforderungen des zu produzierenden Gutes schnell ungesetzt werden können.

Nach [GF98] existiert überwiegend im Bereich des innerbetrieblichen Materialflusses noch ein großes Potenzial zur Anpassung flexibler Transportsysteme an neue Produktionsanlagen.

2.1.1 Streckenmodule für Materialflusssysteme

Die Anforderungen an das Transportsystem ergeben sich aus der geforderten Flexibilität der Produktionsanlage. Gerade schienengebundene Transportsysteme erfordern häufig einen hohen Aufwand zur Anpassung des Materialflusses an neue Produktionsanlagen [GF98].

Um diesem Nachteil entgegen zu wirken, gehen Entwicklungen im Bereich der Materialflusstechnik von einem modularen Aufbau des Transportsystems aus. Ähnlich wie bei einer Modelleisenbahn werden hier vorgefertigte Streckenmodule verwendet, die zu einer Streckentopologie zusammengesetzt werden. Ein Beispiel eines Streckenmoduls wird in Bild 2-1 abgebildet.

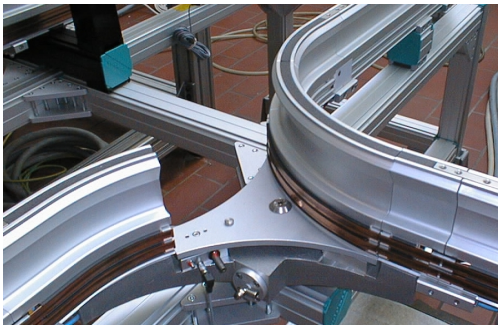


Bild 2-1: Weiche, Teil eines Baukastensystems für modulare Materialflusssysteme.

Dieser Ansatz soll es ermöglichen, die Streckenführung variabel zu halten. Nach [Geh99], [Ger97] sind nur geringe kostensenkende Auswirkungen zu erwarten. Der wesentliche Kostenanteil des Fertigungsleitsystems bezieht sich auf die Software, die zur Steuerung der komplexen Abläufe im System eingesetzt wird. Beim Einsatz zentraler Leitrechner zur Fertigungssteuerung muss nach [Geh99] die Software für jeden Anwendungsfall neu erstellt werden. Entscheidend für den Einsatz eines Transportsystems ist demnach, dass die Modularisierung auf der Ebene der Hardware (Streckenmodule), ebenso wie auf der Softwareebene, stattfindet.

Am Heinz Nixdorf Institut der Universität Paderborn wird im Fachgebiet „Rechnerintegrierte Produktion“ ein System eingesetzt, das die Modularisierung auch auf der Ebene der Steuerungshardware fortsetzt [GF98]. Ziel ist es, intelligente Module zu erhalten, die weitgehend auf Grund ihrer lokalen Information den von ihnen verwalteten Bereich steuern. Um globale Informationen zwischen den einzelnen Modulen austauschen zu können, sind diese durch ein Netzwerk miteinander verbunden.

Da die intelligenten Module den Materialfluss eigenständig steuern können, wird der Aufwand einer Neuprogrammierung der Steuerung beim Umstellen des Streckenverlaufes des Schienensystems erheblich reduziert. Bild 2-2 zeigt ein Steuerungsmodul, das den Ablauf an einer Weiche steuert.

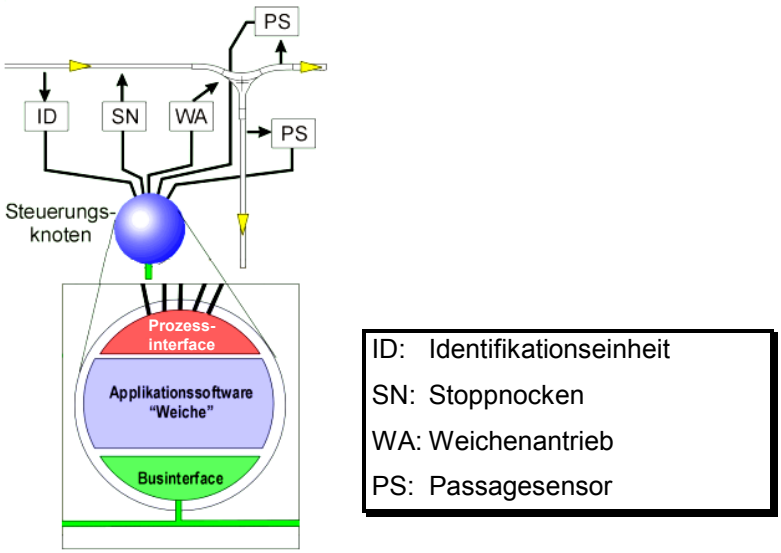


Bild 2-2: Steuerungsknoten einer Weiche, der lokale Information über Sensoren erhält und mit Hilfe von Aktoren Einstellungen vornehmen kann.

Wie auch bei anderen Anlagen der dezentralen intelligenten Automatisierungstechnik sind die Steuerungsmodule des am Institut installierten Fallbeispiels eines dezentral gesteuerten schienengebundenen Transportsystems durch ein Feldbussystem verbunden. Da die Entfernungen der einzelnen Steuerungsmodule (im Folgenden auch *Knoten* genannt) räumlich ausgedehnt sein können und es sich weniger um zeitkritische Prozesse handelt [GF98], wurde zum Aufbau des Netzwerkes das LON (local operating network) gewählt.

Das Controllernetzwerk *LonWorks ™* der Firma *Echelon®* basiert auf dem Neuron-Chip. Im Neuron-Chip sind drei Prozessoren integriert. Zwei Prozessoren bearbeiten das vollständige ISO/OSI 7-Schichtenmodell[1], der dritte Prozessor kann von dem Anwender zur Implementierung seiner eigenen Programme genutzt werden. Aus der Darstellung in Bild 2-2 geht weiterhin hervor, dass der Neuron-Chip Zugriff auf das Kommunikationsnetzwerk hat und eine Schnittstelle zur Überwachung der lokalen Prozesse aufweist.

[1] Das Schichtenmodell wurde im Jahre 1979 von der International Organisation for Standardisation (ISO) entwickelt. Es beschreibt Protokolle auf sieben Schichten, die eine „standardisierte Kommunikationsbasis“ zwischen zwei Rechenanlagen bilden.

Der Anwender kann auf dem Neuron-Chip eigene Programme implementieren. Hierzu dient die Programmiersprache Neuron-C, die als ein C-Dialekt um spezielle Befehle zur Ereignissteuerung erweitert wurde [Geh99].

2.1.2 Das ISILEIT-Projekt

Im Rahmen des DFG-Schwerpunktprogramms *„Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen“* befasst sich ein Projekt mit der integrativen Spezifikation verteilter Leitsysteme in der flexibel automatisierten Fertigung *(ISILEIT)*. Dieses Projekt wird als Zusammenarbeit der Fachgebiete *Rechnerintegrierte Produktion* (Prof. Dr. Gausemeier, HNI), *Entwurf Paralleler Systeme* (Dr. Gläser, HNI) und *Softwaretechnik* (Prof. Dr. Schäfer, Universität Paderborn) bearbeitet. Das Ziel des ISILEIT-Projektes ist, für den integrierten Entwurf, die Analyse und die Validierung verteilter Fertigungssysteme eine durchgängige Methode zu schaffen. In diesem Projekt wird eine Fallstudie eines flexiblen Fertigungssystems eingesetzt. Neben hochautomatisierten CNC-Maschinen und Industrierobotern besteht die Studie aus einem Materialflusssystem, das aus intelligenten Streckenmodulen zusammengesetzt ist. Die AG Softwaretechnik von Prof. Dr. Schäfer hat innerhalb dieses Projektes das Werkzeug *FUJABA* (From Uml to Java and Back Again) entwickelt, das die in [KNN+00] beschriebene integrierte Spezifikationsmethodik unterstützt und daraus ausführbaren Code generieren kann.

Demnach führt dieses Projekt den Schritt zu höherer Flexibilität in verteilten Fertigungssystemen weiter. Gerade im Hinblick auf schienengebundene Transportsysteme ergibt sich eine Entwicklung von einer starren Streckenführung über intelligente Streckenmodule zu einem System, das sowohl in der Hardware als auch in der Programmierung der Steuerungssoftware sehr flexibel einsetzbar ist. Ein weiterer Vorteil ist, dass durch die Spezifikation der Fertigungsanlage und der Codegenerierung die Daten der Streckentopologie und Steuerungsobjekte vorhanden sind. So kann bereits bei der Planung der Anlage die Simulation unterstützend eingesetzt werden, um spätere Fehler der Steuerung bereits frühzeitig erkennen und beheben zu können. Der Einsatz der Simulation kann den Zeitraum von der Aufstellung der Anlage bis zur Inbetriebnahme wesentlich verkürzen [GFS+00, S. 29].

2.2 Simulationsstudien zur Analyse komplexer Systeme

Auf Grund vielfältiger Systemgrößen und Wechselwirkungen einzelner Komponenten, die aus immer komplexeren technischen Systemen resultieren, reichen mathematisch-analytische Verfahren zur Untersuchung dieser Systeme nicht aus.

Die Simulation ermöglicht es, komplexe Systeme zu untersuchen und zu beurteilen [VDI93, S. 2].

Simulationen werden in vielen unterschiedlichen Bereichen eingesetzt. Aufgrund dessen unterliegt der Begriff *Simulation* verschiedenen Interpretationen. So wird die Simulation eingesetzt, um bspw. Piloten zu trainieren. Ziel hierbei ist es, die Geschehnisse in Echtzeitverhalten nachzubilden und auf Benutzereingaben interaktiv zu reagieren.

Ein anderer Einsatzbereich der Simulation beruht darauf, die Realität in einem Modell abzubilden und das Geschehen im Zeitraffer zu betrachten. Nach den VDI-Richtlinien zur *Simulation von Logistik-, Materialfluß- und Produktionssystemen* wird der Begriff *Simulation* definiert als:

> *„[Simulation ist] ein Verfahren zur Nachbildung eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind.“ [VDI96, S.14]*

Der Begriff *Simulation* im weiteren Sinne umschließt ferner die Phasen der Vorbereitung, Durchführung und Auswertung gezielter Experimente mit einem Simulationsmodell [VDI96, S. 14].

Vor allem im Bereich der Produktion, in dem die steigende Produktvielfalt, steigende Flexibilität, sinkende Produktlebensdauer und steigender Kostendruck [vgl. VDI93, S. 2] immer mehr zunehmen, werden Simulationsstudien durchgeführt. In Unternehmen wird die Simulation häufig zur Planung und mittlerweile auch zur Kontrolle der Fertigungssteuerung angewendet. Dies geht unter anderen aus einer Umfrage hervor, in der Benutzer nach ihrer Zufriedenheit mit bestehenden Simulationssytemen befragt wurden [Hlu99]. Eine Frage innerhalb dieser Umfrage zielte auf die Anwendungsbereiche der Simulation ab, die von 66.6 Prozent der Befragten mit „Manufacturing“ beantwortet wurde. Aus den gegebenen Antworten lässt sich ferner erkennen, wie unterschiedlich die Einsatzbereiche der Simulationsstudien sind. An Stelle zwei der Anwendungsbereiche fällt das Gesundheitswesen, gefolgt von der Simulation von Kommunikationssystemen. Aber auch in der chemischen Industrie und in der Verkehrssteuerung finden sich Simulationsstudien.

2.2.1 Simulationsstudien

Das Vorgehen innerhalb einzelner Simulationsstudien wird an vielen Stellen der Literatur aufgegriffen z. B. [Zel92, S. 7], [VDI93, S. 9], [Lie95, S. 221]. Ähnliche Schritte beim Vorgehen innerhalb einer Simualtionsstudie finden sich bei [VDI93], [Lie95]. In beiden Darstellungen werden zyklische Beziehungen her-

vorgehoben, die sich auf Grundlage der durchgeführten Validierung bzw. Verifikation[2] der erstellten Modelle ergeben. Auch bei [Zel92], der den Ablauf einer Simulationsstudie in neun Schritte aufgegliedert, finden sich die Begriffe der Validierung und Verifikation, so dass sich Rückbeziehungen der einzelnen Phasen ergeben können. In [Lie95], [VDI93] werden die Aktivitäten in drei Hauptphasen unterteilt. [Lie95] spricht von den Phasen: Problemdefinition, Modellentwicklung und Entscheidungsunterstützung, hingegen werden in [VDI93] die Phasen Vorbereitung, Durchführung und Auswertung unterschieden werden. Das Vorgehen in [VDI93] bezieht explizit die Modellbildung in die erste Phase der Vorbereitung ein und entspricht somit dem in Bild 2-3 angegebenen Vorgehen. In diesem Zusammenhang möchte ich auf die Abhängigkeiten der Aktivitäten der einzelnen Phasen eingehen, so dass ich die unterschiedliche Zuordnung der Phasen unbeachtet lasse.

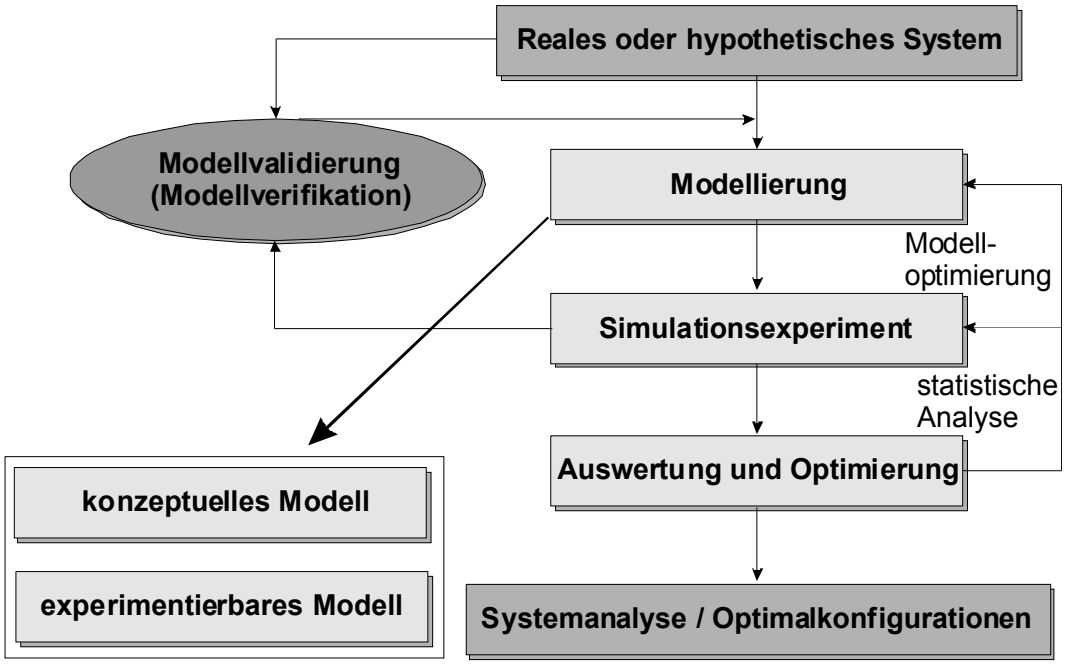


Bild 2-3: Prinzipielles Vorgehen bei Simulationsstudien [nach Wie97-ol]

Aus Bild 2-3 geht das prinzipielle Vorgehen innerhalb von Simulationsstudien hervor. Ausgangspunkt ist ein reales existierendes System bzw. ein noch nicht existierendes hypothetisches System. In einem erstem Schritt wird durch die Analyse, Beobachtung oder eine Vermutung zum erwarteten Verhalten des zu simulierenden Systems ein konzeptuelles Modell erstellt [Wie97-ol]. Es kann abstrakt in natursprachlicher Form oder mit formalen Mitteln beschrieben werden.

[2] Der Begriff Verifikation wird in diesem Zusammenhang als Gewährleistung verstanden, dass ein erstelltes Computermodell den beabsichtigten Vorgaben entspricht. Diese Gewährleistung muss nicht mit einem formalen Beweis verbunden sein, der in der Informatik häufig mit der Verifikation verbunden ist.

In [VDI93, S. 12] wird dieses Modell als symbolisches Modell interpretiert, welches noch nicht experimentierfähig ist. Ein experimentierfähiges Modell ist nach [VDI93] ein Software-Modell, das auf einen Rechner ein symbolisches Modell implementiert.

Auf Grundlage des experimentierfähigen Modells können Simulationsexperimente durchgeführt werden, um das Verhalten des zu untersuchenden Systems zu simulieren. Die Ausführung der Modelle für einen vorgegebenen Zeitrahmen wird hierbei als *Simulationslauf* bezeichnet. Während des Simulationslaufes werden verschiedene Statistiken über den Systemzustand geführt, die in einer anschließenden Phase ausgewertet werden. Diese Auswertungen können zu einer Modelloptimierung bzw. zur Verifikation des implementierten Modells herangezogen werden. Es ist zu beachten, dass ein implementiertes verifiziertes Modell noch nicht valide ist. Die Validität eines Simulationsmodells ergibt sich aus einer hinreichenden Übereinstimmung des Modells mit dem ursprünglichen System.

Das Ergebnis einer Simulationsstudie ist häufig eine Optimierung eines bestehenden Systems. Allerdings dienen Simulationsstudien auch einer Systemanalyse. In der Literatur wird oftmals darauf hingewiesen, dass eine Simulationsstudie nur dann Erfolg haben kann, wenn die zu ermittelnden Ergebnisse eindeutig spezifiziert sind [vgl. Lie95, S. 223].

2.2.2 Instrumente der Simulation

Die unterschiedlichen Aufgaben einer Simulationsstudie sollen durch entsprechende Instrumente erleichtert werden. Diese Instrumente werden durch die Be griffe *Simulationssystem*, *Simulationswerkzeug* oder *Simulator* synonym bezeichnet [VDI 96, S. 15]. Ziel des Simulators ist es, den Anwender bei der Modellerstellung des realen Systems mit seinen dynamischen Prozessen zu unterstützen. Weiterhin soll das erstellte Modell ausführbar sein. Nach [VDI96, S. 16] kann der Simulator eine Programmier- bzw. Simulationssprache[3] sein, wobei alle simulationsspezifischen Funktionen als Paket bereitgestellt werden. Ferner kann der Simulator eine komplette Modellerstellungs- und Experimentierumgebung mit entsprechender Oberfläche oder eine Simulatorentwicklungsumgebung anbieten.

Die Zahl der kommerziell verfügbaren Softwareprodukte, deren Anwendungsbereiche sehr weit gestreut sind, steigt ständig an. Vor allem im Anwendungsbereich der Produktion ist die Palette der Simulationswerkzeuge entsprechend vielfältig. Diese Entwicklung ergibt sich aus dem großen Anwendungsbedarf der Simulationsstudien in diesem Bereich [siehe Hlu99]. Die hierbei verwendete Simulationssoftware sind *Simul8* (Visual Thinking International), *WITNESS* (Lanner Group,

[3] Die Simulationssprache ist eine höhere Programmiersprache, die den Problemstellungen der Simulation angepasst ist [VDI96, S. 15].

Inc.), *Siman/Cinema*, *Simfactory II.5* und *MicroSaint* (Micro Analysis & Design Inc.). Weitere bekannte Simulationspakete bilden *Simlpe++* (Tecnomatix Technologies Inc.) und *Factor/Aim* (Symix/Pritsker Division).

In seinem Vortrag über die Grundlagen der Modellierung und Simulation geht [Wie97-ol] auf die Vor- und Nachteile der unterschiedlichen Werkzeuge zur Simulation ein. Er unterscheidet Programmiersprachen, Simulationssprachen und die Simulationssoftware. Die kommerziellen Simulationssoftwarepakete erlauben eine Modellierung mit minimalen Programmierkenntnissen, sind in ihrem Einsatzbereichen aber oft eingeschränkt. Bezogen auf die Flexibilität räumt [Wie97-ol] den Simulationssprachen Vorteile ein.

2.2.3 Simulationsmethoden

Neben der Unterscheidung der Werkzeuge zur Unterstützung der Simulationsstudien ist zu entscheiden, welche Simulationsmethode eingesetzt wird. Laut [VDI96, S. 14] bestimmt die Simulationsmethode die Art des Voranschaltens der Simulationszeit und die Durchführung der Zustandsänderungen im modellierten System. In Bild 2-4 werden die unterschiedlichen methodischen Ansätze der Simulation angegeben.

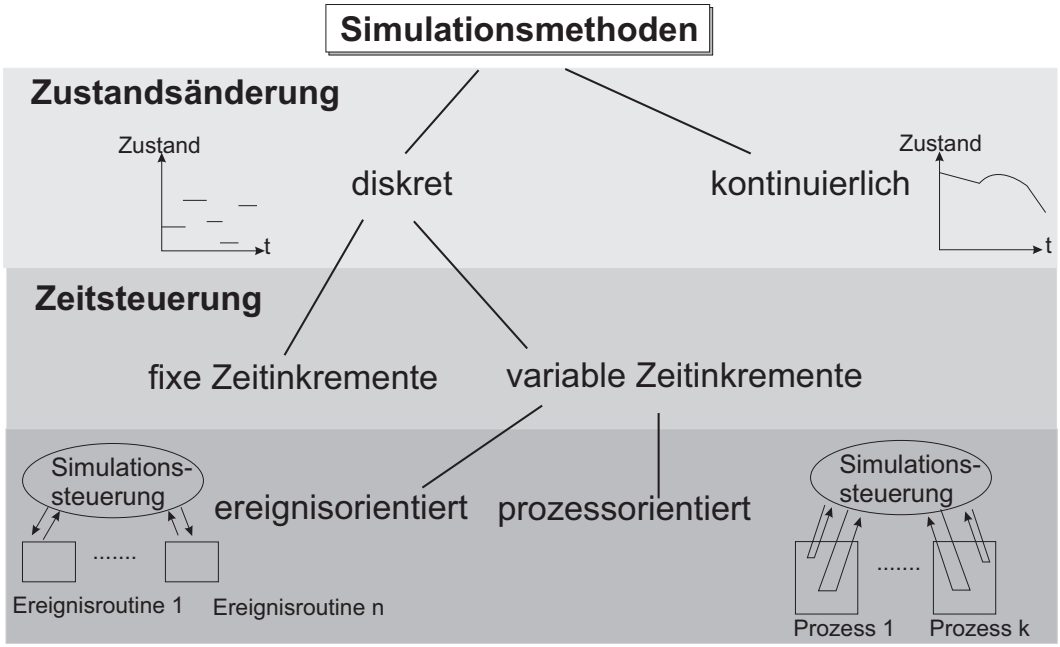


Bild 2-4: Grobe Übersicht unterschiedlicher Simulationsmethoden

Bei der Art der Zustandsänderungen wird nach den diskreten und den kontinuierlichen Methoden getrennt. Die beiden Ansätze zielen auf den Modus, nach dem die variablen Größen ihre Werte verändern. Ändern sich die Werte sprunghaft, so

liegt ein diskreter Übergang vor. Befinden sich z. B. zum Zeitpunkt t=0 fünf Aufträge in der Warteschlange vor einer Maschine, so ändert sich diese Anzahl sprunghaft, wenn neue Aufträge eingehen bzw. ein Auftrag bearbeitet wurde und an die folgende Station weitergeleitet wird. Die Variable zum Speichern der Anzahl ändert ihre Werte diskret. Im Gegensatz zu diskreten Systemen ändern sich in kontinuierlichen Systemen die Zustände nicht sprunghaft, sondern stetig in der Zeit. Als Beispiel dient der kontinuierliche Beschleunigungsvorgang eines Fahrzeuges [vgl. Lie95, S. 9]. Nach [Lie95] ist darauf zu achten, dass nicht jedes System, in dem Beschleunigungsvorgänge enthalten sind, durch eine kontinuierliche Simulation modelliert werden muss. Vielmehr ist die Fragestellung zu berücksichtigen, die mit dem Zweck der Simulation einhergeht.

In einem kontinuierlichen System werden zur Darstellung des Systemzustandes Differential- oder Differenzengleichungen genutzt [WCT94, S. 19]. Die Zeitfortschaltung arbeitet zumeist mit gleichbleibenden Zuwächsen, d. h. die virtuelle Simulationszeit wird immer um ein festes Zeitintervall erhöht. Anwendungsbereiche der kontinuierlichen Simulation finden sich häufig in den Naturwissenschaften, z. B. zur Untersuchung von Schaltkreisen bzw. der Reaktion von Pendeln. Auch in der Städteplanung und in der Volkswirtschaftslehre wird diese Simulationsmethode eingesetzt [Lie95, S. 10].

In der Betriebswirtschaftslehre wird überwiegend die diskrete Simulation verwendet [Lie95, S. 10]. Auch in den Softwarepaketen zur Simulation von Produktionsanlagen werden vorwiegend die diskreten Ansätze eingesetzt, wenn auch Möglichkeiten zur kontinuierlichen Simulation vorgesehen sind. Nach [Lie95] lässt sich eine genaue Trennung zwischen diskreten und kontinuierlichen Systemen in vielen Fällen nicht vornehmen. Vielfach wird ein kontinuierliches Verhalten durch eine diskrete Simulationsmethode angenähert.

Die diskreten Ansätze der Simulation werden weiterhin unterteilt nach der Art, in der die Simulationszeit fortgeschaltet wird. Es werden Methoden mit fixen und mit variablen Zeitinkrementen differenziert. Bei einem Simulationsmodell, in dem *fixe Zeitinkremente* eingesetzt werden, wird die Simulationszeit immer um ein konstantes Intervall erhöht. Vorteilhaft an dieser Methode ist die relativ einfache Implementierung des Verfahrens. Ein gravierender Nachteil ergibt sich jedoch aus der zuvor festzulegenden Größe des Zeitintervalls. Bei [Lie95, S. 91] werden mehrere Konsequenzen aufgeführt, die sich aus einem festen Zeitabschnitt ergeben. Ein Nachteil resultiert aus den *ties*, die eine Situation bezeichnen, in der in einer festgelegten Periode zwei oder mehrere Ereignisse[4] auftreten, die miteinander in Beziehung stehen. Da die gewählte Schrittweite der Zeitfortschaltung die kleinste Einheit ist, nach der die Ereignisse aufgelöst werden können, können die-

[4] Ereignis: Atomare Begebenheit, die eine Zustandsänderung bewirkt und keine Zeit verbraucht [VDI96, S. 6]

se Konstellationen nur im nächsten Abschnitt behoben werden. Die Zeitinkremente sollten bei diesem Ansatz demnach nicht zu groß gewählt werden, da es sonst zu viele Ereignisse in einer Periode geben kann. Wird das Inkrement allerdings zu klein gewählt, können infolgedessen Berechnungen ausgeführt werden, wobei in den meisten jedoch keine Veränderung des Systemzustandes stattfindet. In diesem Fall wird die Rechenzeit des Simulationslaufes unnötig verlängert.

Eine weitere Methode der diskreten Simulation wird in [Lie95, S. 92] als Ansatz mit *variablen Zeitinkrementen* bezeichnet. Im Benutzerhandbuch [DCS99-ol] wird die Bezeichnung *„kontinuierliche Zeit – diskrete Ereignisse“* gewählt. Das Ziel dieser Methode ist es, die Erhöhung der Simulationszeit von den Ereignissen im System abhängig zu machen und nicht von einem fixen Zeitintervall. Nach [Lie95] wird die Zeit durch einen globalen Fortschaltungsmechanismus nicht mehr synchron, sondern asynchron aktualisiert. Es ist somit möglich, dass Ereignisse zu einem beliebigen Zeitpunkt eintreten können. Die Berechnungen des Systemzustandes erfolgen nur, wenn eine Veränderung stattgefunden haben kann. Die Nachzüge, die sich aus einem festen Intervall ergeben, treten demnach nicht mehr auf.

Während die fixen Zeitinkremente dem Simulationsprogramm die Zeit vorschreiben, ist es bei der Methode der variablen Zeitinkremente dem Programm selbst überlassen, die Zeitsteuerung zu übernehmen. Diese ergibt sich hier aus der Koordination der Veränderungen des Systemzustandes. Auch an dieser Stelle werden wieder mehrere Verfahren unterschieden. [Lie95, S. 92] bezeichnet die Ansätze als *„event scheduling approach“*, *„activity scanning approach“*, *„process interaction approach“*. In [VDI96] entspricht diese Einteilung der *ereignisorientierten*, *aktivitätsorientierten* bzw. *prozessorientierten* Methode.

Die unterschiedlichen Ansätze werden an vielen Stellen der Literatur behandelt, an dieser Stelle wird nur auf die ereignisorientierte bzw. die prozessorientierte Methode eingegangen. Nach [Lie95, S. 111] haben die aktivitätsorientierten Ansätze zumeist nur nachrangige Bedeutung, wenn auch in England eine Fortentwicklung der „*three-phase approach*“ häufig angewandt wird [Pid95]. [Lie95, S. 109] bemerkt zu diesem Ansatz, dass kaum noch eine Unterscheidung zwischen dem ereignisorientierten bzw. diesem aktivitätsorientierten Ansatz getroffen werden kann.

Der ereignisorientierte Ansatz

Im Folgenden wird der ereignisorientierte Ansatz erläutert. Ich berufe mich hierbei auf die Ausführung von [Lie95, S. 94ff]. Der Kerngedanke dieser Methode ist, bei der Zeitfortschreibung in einem Simulationsprogramm ausschließlich Ereignistermine zu betrachten. Die Systemzustände werden von Ereignis zu Ereignis aktualisiert. Da dieser Ansatz wesentlich auf der Abarbeitung der Ereignisse beruht, besteht die Aufgabe bei der Modellbildung zunächst darin, alle Ereignisse im

System zu erkennen und die gegenseitigen Abhängigkeiten zu berücksichtigen. Es werden auslösende und blockierende Abhängigkeiten unterschieden. Auslösende Beziehungen bestehen, wenn Ereignis A ein Ereignis B unmittelbar nach sich zieht. Blockierende Abhängigkeiten liegen vor, wenn ein Ereignis B erst dann eintreten kann, sofern zuvor Ereignis A stattgefunden hat. Zwischen einzelnen Ereignissen besteht somit ein zeitlicher Zusammenhang. Die Zeitfortschaltung ergibt sich nun aus der Bearbeitung einer Menge solcher Ereignisse, die nach ihren Zeitpunkten bearbeitet wird. Diese Menge wird in der Literatur als *„future event set (FES)“* bezeichnet.

In Bild 2-5 wird ein Beispiel gegeben, das die Arbeitsweise verdeutlichen soll [vgl. Lie95, S. 100]. Das System besteht aus sechs möglichen Ereignissen. Zum aktuellen Zeitpunkt 11.59 sind die Ereignisse drei, vier und sechs terminiert, d. h. diese Ereignisse treten zu den in der FES festgelegten Zeitpunkten auf. Da das Ereignis drei als nächstes auftreten wird, wird die Simulationszeit auf diesen Zeitpunkt eingestellt und die Ereignisse eins und fünf, die abhängig von Ereignis drei sind, terminiert. Nach der Aktualisierung der FES wird die Simulationszeit auf 11.59 belassen und das Ereignis eins ausgeführt.

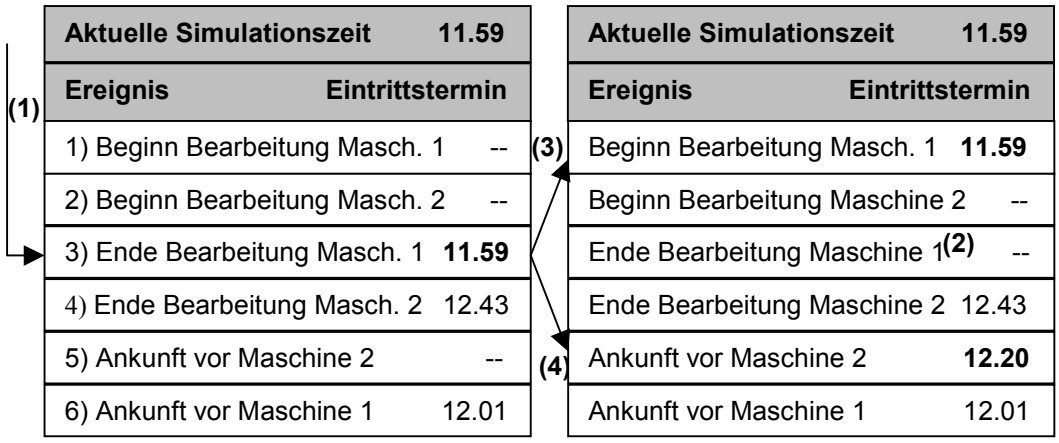


Bild 2-5: Zeitsteuerung der ereignisorientierten Simulationsmethode. Das Fertigungsende auf Maschine 1 erzeugt die Terminierung der Ereignisse „Beginn der Bearbeitung auf Maschine 1“ und „Ankunft vor Maschine 2“ [Lie95, S. 100].

Der Vorteil des ereignisorientierten Ansatzes ist die relativ einfache Implementierung des Mechanismus zur Fortschaltung der Simulationszeit. Sind alle Ereignisse und ihre Abhängigkeiten bekannt, bietet dieser Ansatz eine einfache Möglichkeit, das System zu simulieren. Der Nachteil der Methode liegt u. a. in der zu erfüllenden Voraussetzung, dass alle Ereignisse bekannt und ihre Abhängigkeiten eindeutig festgelegt sein müssen. In sehr komplexen Systemen mit vielen Abhängigkei-

ten der Ereignisse untereinander können leicht Nebeneffekte eintreten, die meist nur sehr schwer aufgedeckt werden können [Lie95, S. 101, FA96, S. 103].

Der prozessorientierte Ansatz

Während die ereignisorientierte Methode durch die Beschreibung aller Ereignisse und Zustandsübergänge zu sehr unübersichtlichen Modellen führen kann, geht der prozessorientierte Ansatz von einer *„...anderen Sicht der Welt aus“* [Lie95, S. 107]. Die prozessorientierte Sichtweise geht davon aus, dass die Objekte im System in einem Zyklus das System durchwandern. Nach [Lie95] wird diese Tatsache in den ereignis- und aktivitätsorientierten Ansätzen ignoriert. Beide Verfahren beruhen auf einer Aneinanderreihung verschiedener Ereignisse bzw. Aktivitäten[5]. Ein Prozess im Sinne der Simulationstechnik umfasst alle Aktivitäten, die für ihn relevant sind.

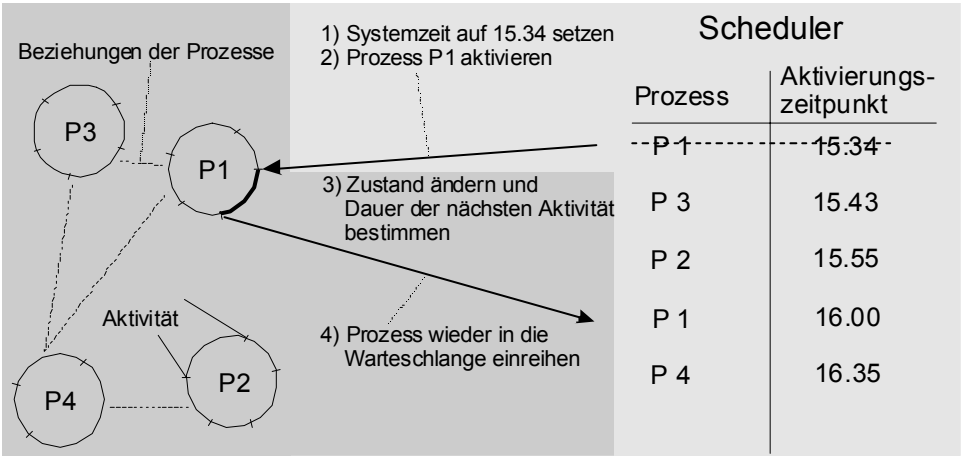


Bild 2-6: Zeitliche Koordination der Prozessverläufe durch einen Scheduler. Nach der Aktivierung erhält der jeweilige Prozess die Kontrolle über den weiteren Simulationsverlauf, bis er sie wieder an den Scheduler zurückgibt.

Während nach Meinung vieler Autoren [z. B. Lie95, Fa96] die andere Weltsicht des prozessorientierten Ansatzes oft eine intuitivere Vorgehensweise bei der Modellierung erlaubt, entsteht durch die Verwaltung mehrerer Prozesse ein Koordinierungsproblem. Prozesse müssen so koordiniert werden, dass sie aktiviert werden, sobald für sie relevante Änderungen stattfinden bzw. müssen angehalten werden, wenn sie nicht mehr von Ereignissen betroffen sind. Somit sind die zu verwaltenden Prozesse dennoch von Ereignissen abhängig. Dieser Aspekt zeigt sich in der Betrachtung des Abarbeitungszyklus zur zeitlichen Steuerung im prozessorientierten Ansatz. In [FA96, S. 111] wird das prozessorientierte *„schedu-*

[5] Aktivität: Zeitverbrauchender Vorgang, der von einem Anfangs- und einem End-Ereignis begrenzt wird und zu einem Zustandsübergang führt [VDI96, S. 2]

ling-Verfahren" als dem in der ereignisorientierten Steuerungstechnik eingesetzten Ansatz ähnlich angesehen. Das „scheduling-Verfahren" beruht auf einer Liste, in der mehrere Prozesse mit ihrem zugehörigen notwendigen Aktivierungszeitpunkt geführt werden (Bild 2-6). Aus dieser Liste wird der Prozess mit dem frühesten Zeitpunkt entnommen und aktiviert. Nachdem dieser Prozess seine Veränderungen vorgenommen hat, übergibt er die Steuerung wieder dem Scheduler, der nun den nächsten Prozess aktiviert.

2.3 Der objektorientierte Ansatz in der Simulation

Wie in Abschnitt 2.2.1 beschrieben ist eine Hauptaufgabe bei der Durchführung einer Simulationsstudie das Erstellen eines Modells des zu simulierenden Systems. In diesem Abschnitt wird aufgezeigt, wie mit Hilfe der Objektorientierung übersichtliche und erweiterbare Modelle erzeugt und notiert werden können. Zunächst wird darauf eingegangen, wie die objektbasierte Sichtweise eine systemorientierte Sicht verfeinert. Danach werden Faktoren behandelt, die eine objektorientierte Modellierung unterstützen. Abschließend wird ein Überblick über die Entwicklung objektorientierter Methoden geben. Ziel ist es, auf Diagrammtypen der UML hinzuarbeiten, die im weiteren Verlauf der Arbeit verwendet werden.

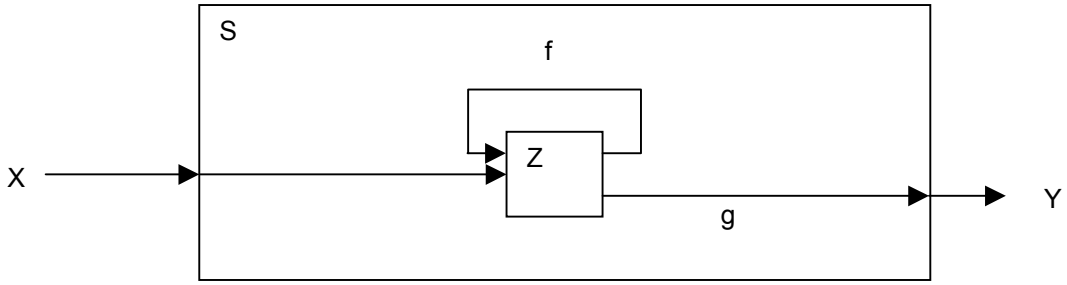
Nach [VDI96, S. 17] ist ein System eine abgegrenzte *„... Anordnung von Komponenten, die miteinander in Beziehung stehen."* Nach dieser Definition ist ein System gekennzeichnet durch (zur weiteren Definition siehe [VDI96]):

- Systemgrenze, Systemein- und –ausgangsgrößen
- Subsysteme, Systemelemente
- Aufbaustruktur
- Ablaufstruktur
- Ablauflogik
- Zustandsübergänge und –größen

[Fa96, S. 33] beschreibt ein System durch ein Tupel $S=(Z, z_0, X, Y, f, g, time)$ vgl. Bild 2-7.

Die objektorientierte Modellierung ist nach [Fa96, S. 38] keine alternative Methode zur systemorientierten Sichtweise. Sie verfeinert und konkretisiert vielmehr die systemorientierte Sichtweise. Die Verfeinerung wird in zwei Schritten vollzogen. Zunächst wird die systemorientierte Sicht durch die Ergänzung von Objekten und Klassen zu einer *objektbasierten Sicht* erweitert. Diese Sicht erlaubt die Typisierung von Objekten als Systemelemente. Der Zustand der Objekte wird durch Attribute beschrieben, ihr Verhalten durch Methoden. Die Methoden werden unterteilt in innere Aktionen und Interaktionen. Als innere Aktionen werden Methoden bezeichnet, die ein Objekt auf sich selbst aufruft. Interaktionen entstehen durch

Kommunikation der Objekte untereinander. Die Kommunikation kann z. B. durch Aufruf einer Methode in einem zweiten Objekt erfolgen. Durch die Interaktionen werden Abhängigkeiten der Objekte untereinander beschrieben. Da das objektbasierte Modell ein System ist, hat es eine wohldefinierte Abgrenzung zur Umgebung [vgl. FA96, S. 39-43]. Das von [FA96] beschriebene objektbasierende Modell ist durch die Einführung der Objekte und Klassen eine erste Verfeinerung der systemorientierten Sicht. Weitere Aspekte der objektorientierten Modellierung, wie die Vererbung und Polymorphie, erlauben eine bessere Strukturierung der objektbasierten Sicht. Diese Punkte werden in Abschnitt 2.3.1 kurz aufgegriffen.



Z	Menge der möglichen Zustände
z_0	Anfangszustand
X	Menge der möglichen Eingänge
Y	Menge der möglichen Ausgaben
time	Zeitmenge als (T, <=, t_0) mit möglichen Zeitpunkten T, einer Ordnungsrelation <= auf T und einem minimalen Element t_0
f	Z x X x T → Z als Zustandsüberführungsfunktion und
g	Z x X → Y als Ausgabefunktion

Bild 2-7: Allgemeine Wirkungsstruktur in einem System [FA96, S. 33]

Neben der Beschreibung des Systems ist die Wahl der Zeitfortschaltungssteuerung bei der Erstellung einer Simulation wichtig. Auch aus Bild 2-7 geht die Abhängigkeit des betrachteten Systems von der Zeit hervor. Bezogen auf den objektorientierten Ansatz wurden hier sowohl die ereignisorientierte [PP99] als auch die Verfeinerung der aktivitätsorientierten [Pid95] sowie die prozessorientierte Methode [FA96] betrachtet. [FA96, S. 104] vertreten die Ansicht, dass die prozessorientierte Simulation durch den Einsatz einer objektorientierten Sprache sehr flexibel eingesetzt werden kann. Diese Auffassung wird auch in [DCS99-ol] geteilt: *„The process-oriented approach best fits with the object-oriented paradigm...“*.

2.3.1 Objektorientierte Modellentwicklung

Ein objektbasiertes Modell unterstützt das Konzept der Datenabstraktion, welches die innere Struktur eines Objektes verbirgt und nur auf das Objekt anwendbare

Operationen nach außen sichtbar macht [Oes98]. Ein objektorientiertes Modell bietet weitere Eigenschaften, die den Einsatz einer objektorientierten Modellierung gerade in komplexen Systemen gegenüber anderen Ansätzen prädestiniert [FA96, RD98]. Durch Verwenden der Konzepte der Objektorientierung wie Vererbung, Polymorphie und dynamische Typbindung beim Design der Simulationsmodelle und die Implementierung in einer objektorientierten Programmiersprache wird es dem Entwickler erlaubt, die Komponenten des Systems und ihre Beziehungen untereinander direkt in seinem Modell abzubilden. Die Entwicklungszeit des Modells wird verringert, die Wartbarkeit verbessert, es wird leichter modifizierbar, die Software und das Design sind wiederverwendbar und weiterzuentwickeln, die Risiken der Entwicklung komplexer Systeme werden reduziert. Diese genannten Eigenschaften sprechen für den objektorientierten Ansatz in der Simulation. Es ist aber darauf zu achten, dass der Ansatz alleine diese Eigenschaften nicht garantieren kann. *„Dennoch ist Objektorientierung kein Allheilmittel ... auch mit diesem Ansatz ist es weiterhin möglich, ganz lausige Ergebnisse zu erzielen.“* [Oes98, S. 31]. Vielmehr ist in der objektorientierten Modellierung, also der Anwendung der Objekttechnologie[6] bei der Modellbildung weitere Umfeldfaktoren zu beachten [Bur99, S. 6]:

- das Vorgehen
- die Methode
- das Werkzeug

Die drei Umfeldfaktoren sollen nachfolgend kurz erläutert werden. Die Ausführungen beziehen sich überwiegend auf [Bur99].

Vorgehen

In der objektorientierten Modellierung wird zwischen zwei unterschiedlichen Bereichen des Vorgehens unterteilt [siehe auch NS99]:

- methodenspezifischer Modellierungsprozess
- projektbezogenes Vorgehensmodell

Wird zur Modellierung eine spezielle Methode ausgewählt, so unterteilt sich diese zumeist in unterschiedliche Phasen. Innerhalb dieser Phasen werden einzelne Aktivitäten ausgeführt. Die Ergebnisse der Aktivitäten müssen entsprechend festgehalten werden. In diesem Zusammenhang orientiert sich der methodenspezifische Modellierungsprozess an den *„... Besonderheiten der Methode in Notation und Zielsetzung. ...“* [Bur99, S. 7].

[6] Die Objekttechnologie ist ein Paradigma und ein Wissenschaftsgebiet, das sich heute zur ausgereiftesten Softwareentwicklungstechnologie entwickelt hat. Darüber hinaus greift sie langsam auch auf viele andere Wissenschaftsbereiche außerhalb der Informatik über und bietet dort ein Mittel zur Repräsentation von vielfältigen Wissensstrukturen in unzähligen Anwendungsfeldern [Bur99, S. 445].

Der zweite Bereich der Vorgehensmodelle umfasst das projektbezogene Vorgehen. Neben dem methodenspezifischen Vorgehen sind *„ (...) dabei auch verschiedene Tätigkeitsbereiche wie die Qualitätssicherung, das Konfigurationsmanagement oder das Projektmanagement...“* [NS99, S.166] zu beachten. Das Projektmanagement dient dazu, die eingesetzten Ressourcen zu koordinieren.

Methode

Der Begriff der Methode beschreibt die Art und Weise, wie zum Erreichen bestimmter Ziele bzw. zur Lösung bestimmter Probleme vorzugehen ist. Laut [Bal99, S. 544] ist der Begriff *Methode* in der Softwaretechnik ein Oberbegriff. Er wird zur Bezeichnung von Konzepten, Notationen und methodischen Vorgehensweisen verwendet. Bezogen auf die Notation eines objektorientierten Modells hat sich in den letzten Jahren die Modellierungssprache UML als durch die *Object Management Group* (OMG) akzeptierter Standard durchgesetzt [Bur99, NS99]. Da unterschiedliche Diagrammformen der UML im weiteren Verlauf dieser Arbeit auftreten, wird hierauf im nächsten Abschnitt einführend eingegangen.

Werkzeug

Ein Werkzeug, das als Software-Entwicklungsumgebung eingesetzt werden soll, muss sowohl die Aspekte der Modellierung als auch der Softwarentwicklung integrieren [Bur99]. Es ist ein Programm, das als Hilfsmittel zur Entwicklung von Software eingesetzt wird.

2.3.2 Die UML Modellierungssprache

Es ist die Absicht der objektorientierten Modellierung, in allen Phasen der Software-Entwicklung wie Analyse, Entwurf und Implementierung mit einheitlichen Konzepten zu arbeiten [Eng98, S. 65]. Klassische Konzepte enthalten zumeist unterschiedliche Methoden zur Erarbeitung der einzelnen Phasen. Gegenüber diesen Ansätzen ist die Objekttechnonolgie ein phasenübergreifendes Konzept, das es erlaubt, die Implementierung nahe am Modell zu halten [Bur99, S. 305]. Um das angestrebte Ziel zu erreichen, müssen geeignete objektorientierte Methoden entwickelt werden, die den gesamten Software-Entwicklungsprozess unterstützen. Ein entscheidender Schritt in Richtung der objektorientierten Modellierung wurde im Jahr 1991 gesetzt [Bur99, S. 1]. In diesem Jahr erschien eines der ersten Bücher zur Objekttechnologie von Grady Booch. Die weitere Entwicklung der Methoden ist in der Tabelle 2-1 kurz aufgeführt [Eng98, S. 67-69]. Zur näheren Erläuterung der angegebenen Methoden sei auf [Bur99, OHJ+99] verwiesen.

Tabelle 2-1: Übersicht über objektorientierte Methoden [Eng98, S. 67]

Objektorientierte Methoden	Autor	Jahr
Object Oriented Design (OOD)	Booch	1991
Object Modeling Technique (OMT)	Rumbaugh	1991
OO Software Engeering (OOSE)	Jacobson	1992
Shlaer-Mellor		1988
Object-Oriented Analysis (OOA)	Coad-Yourdon	1991
HOOD (Ada)		
Unified Method	Booch, Rumbaugh, Jacobson	1994

Die Autoren der Unified Method[7] wollten ihre drei Ansätze miteinander verbinden [Eng98, S. 68]. Unter Mitwirkung anderer Firmen[8] entstand die UML, als *„…eine Sprache und Notation zur Spezifikation, Konstruktion, Visualisierung und Dokumentation von Modellen für Softwaresysteme." [Oes98, S. 203]*. Aus dem angegebenen Zitat und der gewählten Bezeichnung als *modelling language* ist zu erkennen, dass die UML eine Sprache und Notation ist. Sie beinhaltet keine methodische Vorgehensweise und ist demnach keine Methode für die objektorientierte Modellierung [Bur99, S. 4; Oes98, S. 203;OHJ+99, S. 15]. Die UML wurde am 17. November 1997 als Standard zur objektorientierten Modellierung von der OMG akzeptiert. Auf der Grundlage der UML müssen nun Vorgehensmodelle entwickelt werden, um eine objektorientierte Methode zu erhalten. Die Initiatoren der UML haben Arbeiten zu einem auf UML basierenden Vorgehensmodell mit dem Ergebnis des *Unified Software Development Process* (USDP) veröffentlicht [OHJ+99, S. 15].

Die UML erlaubt eine grafische Notation zur Darstellung objektorientierter Modelle. Sie besteht aus einer Vielzahl von Modellelementen [Oes98, S. 204]. An dieser Stelle soll ein Überblick über die verschiedenen Diagrammtypen vermittelt werden, die in der UML verwendet werden.

---

7 Grady Booch, Jim Rumbaugh, Ivar Jacobson

8 Beispielsweise: Digital Equipment, Hewlett-Packard, i-Logix, ICON-Computing, MCI Systemhouse, Microsoft, Oracle, Texas Instruments und Unisys [Oes98, S. 203]

Die Diagramme der UML lassen sich in vier Gruppen einteilen [Eng98, S. 70], die ich näher erläutern möchte:

1. Anwendungsfalldiagramme[9]
2. Strukturdiagramme
3. Verhaltensdiagramme
4. Implementierungsdiagramme

Anwendungsfalldiagramme

Diese Diagrammtypen werden gewählt, wenn das externe Systemverhalten dargestellt werden soll. Anwendungsfälle beziehen sich zumeist auf Geschäftsprozesse und beschreiben, welche Arbeitsgänge ein Benutzer in einem Geschäftsvorfall bearbeitet. Es wird eine Menge von Aktivitäten eines Systems beschrieben, die für den Akteur (Benutzer/System) wahrnehmbar sind. Hierbei ist stets die Sicht des Akteurs zu wählen [Oes98, S. 207].

Strukturdiagramme

Diagramme dieses Typs beschreiben die Struktur des Modells. Mit Hilfe der spezifizierten Elemente wird es dem Modellierer erlaubt, statische Modellsachverhalte abzubilden. Im Vordergrund stehen hier die benutzten Klassen und Objekte und ihre Beziehungen untereinander.

Verhaltensdiagramme

Im Gegensatz zu den Strukturdiagrammen werden Verhaltensdiagramme verwendet, um dynamische Modellsachverhalte aufzuzeigen [Oes98, S. 293]. Hierbei kann sowohl der interne Zustand eines Objektes als Reaktion auf externe Einflüsse als auch die Kommunikation der Objekte untereinander modelliert werden.

Implementierungsdiagramme

Die Implementierungsdiagramme zeigen die Komponenten des Systems und ihre Beziehungen bzw. die Komponenten und die Knoten, auf denen die Komponenten ausgeführt werden. Interessant sind diese Diagrammtypen vor allem im Hinblick auf verteilte Programmierung.

Obschon die UML durch die OMG als Standard akzeptiert worden ist, werden zur objektorientierten Modellierung unterschiedliche Notationen eingesetzt. Die zunehmende Verbreitung der UML [GKM00b] und ihre Verwendung im ISILEIT-Projekt [KNN+00] führen dazu, dass in dieser Arbeit Diagramme der UML eingesetzt werden. Statische Strukturen werden nach Tabelle 2-2 durch Klassendiagramme dargestellt. Zur Beschreibung des Verhaltens und der Beziehungen einzelner Objekte untereinander können unterschiedliche Diagrammtypen

[9] Es wird darauf hingewiesen, dass auch im deutschen Sprachgebrauch der Begriff *Use Case* verbreitet ist. Daher wird auch der Begriff *Use Case Diagramme* verwendet.

herangezogen werden. Da in Simulationen die zeitliche Synchronisation der Ereignisse wichtig ist und enge Beziehungen zwischen den Objekten bestehen, werden zumeist Sequenzdiagramme verwendet. Im Gegensatz zu den Kollaborationsdiagrammen, die in [KNN+00] Objektbeziehungen modellieren, heben Sequenzdiagramme den zeitlichen Ablauf der Kommunikation hervor. Die Notation der UML ist Lehrbüchern [Bal99] bzw. der Spezifikation der UML [UML99-ol] zu entnehmen.

Tabelle 2-2: Diagrammtypen der UML

Diagrammtyp	**Diagramm-bezeichnung**	**Aufgabe**
Use Case Diagramme	Anwendungsfall-diagramm	Zeigt Akteure, Anwendungsfälle und ihre Beziehungen
Strukturdiagramme	Klassendiagramm	Zeigt Klassen und ihre Beziehungen
Verhaltens-diagramme	Aktivitätsdiagramm	Zeigt Aktivitäten, Objektzustände, Zustände und Ereignisse
	Kollaborations-diagramm	Objekte und ihre Beziehungen inklusive ihres räumlich geordneten Nachrichten-austausches
	Zustandsdiagramm	Zeigt Zustände, Zustandsübergänge und Ereignisse
	Sequenzdiagramm	Objekte und ihre Beziehungen inklusive ihres zeitlich geordneten Nachrichten-austausches
Implementierungs-diagramme	Komponenten-diagramm	Zeigt Komponenten und ihre Beziehungen
	Verteilungsdiagramm	Zeigt Komponenten, Knoten und ihre Beziehungen