

# **Web- und Komponenten-Technologien in der Modellierung und Simulation**

---

Neue Möglichkeiten der Modellerstellung,  
-ausführung und -optimierung

von der Fakultät für Informatik der Universität Karlsruhe  
angenommene Habilitationsschrift

**Michael Syrjakow**

Tag der Habilitation: 16.7.2003

Referenten: Prof. Dr. Detlef Schmid  
Prof. Dr. Axel Lehmann



# Vorwort

---

Die vorliegende Arbeit, die während meiner Tätigkeit als wissenschaftlicher Assistent am Institut für Rechnerentwurf und Fehlertoleranz an der Universität Karlsruhe entstand, wurde von der Fakultät für Informatik der Universität Karlsruhe als Habilitationsschrift angenommen.

An dieser Stelle möchte ich mich besonders bei Herrn Prof. Dr.-Ing. Detlef Schmid für die Betreuung der Arbeit und die Übernahme des Referats bedanken. Sein stetes Interesse, seine tatkräftige Unterstützung sowie der von ihm gewährte Gestaltungsspielraum in Forschung und Lehre haben wesentlich zum Gelingen dieser Arbeit beigetragen.

Besonderen Dank schulde ich auch Herrn Prof. Dr. Axel Lehmann für die Übernahme des Korreferats und die damit verbundene Mühe.

Frau Prof. Dr.-Ing. Helena Szczerbicka möchte ich für die interessanten Diskussionen und wertvollen Anregungen danken, die mir sehr dabei geholfen haben, diese Arbeit erfolgreich abzuschließen.

Auch meinen (ehemaligen) Kolleginnen und Kollegen aus der Arbeitsgruppe "Modellierung" Herrn Dr. Klaus-Peter Huber, Herrn Dr. Peter Ziegler, Herrn Dr. Jörg Berdux und Frau Dipl.-Inform. Elisabeth Syrjakow möchte ich für die stets kooperative und freundschaftliche Zusammenarbeit danken, die zu zahlreichen Publikationen führte.

Ferner möchte ich den Studentinnen und Studenten Herrn Matthias Liefländer, Herrn Christian Bentz, Herrn Bernd Zimmermann, Frau Anke Otto, Herrn Dietmar Püttmann, Frau Sabine Schillinger und Herrn Florian Schmidt für ihr großes Engagement und die zuverlässige Durchführung der Implementierungsarbeiten danken.

Mein Dank gilt auch allen Mitarbeitern und Angestellten des Instituts für die gute Zusammenarbeit und die angenehme Arbeitsatmosphäre.

Aus tiefstem Herzen danke ich meiner Frau Nadija für ihre Geduld und das große Verständnis, das sie mir während des Verlaufs meiner Habilitation entgegenbrachte.

Karlsruhe, im Juli 2003

*Michael Syrjakow*



# Zusammenfassung

---

Mitte der 90er Jahre wurde damit begonnen, den Einfluss und die Auswirkungen des World Wide Web und der damit eng verbundenen Komponenten-Technologien auf die Modellierung und Simulation eingehend zu untersuchen. Zu den bedeutendsten Teilbereichen des daraus hervorgegangenen Forschungsgebiets zählen heute Web-basierte Simulationsanwendungen, der komponentenorientierte Aufbau verteilter Simulationsmodelle und -werkzeuge sowie deren Erweiterung um bereichsfremde Spezialkomponenten.

Diese Arbeit gibt zunächst einen Gesamtüberblick über das Themengebiet "Web- und Komponenten-Technologien in der Modellierung und Simulation". Danach wird im Detail auf Komponentenarchitekturen für Modellierungswerkzeuge eingegangen. Zu einem wichtigen Bestandteil von Modellierungswerkzeugen zählen heute Spezialkomponenten zum zielgerichteten Experimentieren mit Modellen. Aufgabe dieser Komponenten ist es, aus meist sehr komplexen Modellen für den Modellierer relevante Informationen zu extrahieren beispielsweise über optimale Parametereinstellungen, Parameter hoher Empfindlichkeit, etc. Diese Aufgabe ist bei hoher Modellkomplexität meist nicht mehr von Hand durchführbar, sondern es werden dazu systematisch vorgehende Experimentierstrategien benötigt. In dieser Arbeit werden leistungsfähige Methoden zur Ermittlung optimaler Parametereinstellungen von Simulationsmodellen vorgestellt und diskutiert, die in Form einer universell einsetzbaren Optimierungskomponente realisiert wurden. Darüber hinaus werden Forschungsprojekte beschrieben, in denen die entwickelte Optimierungskomponente sowie weitere im Rahmen dieser Arbeit behandelte Methoden und Technologien konkret angewendet wurden.



# Inhaltsverzeichnis

---

<b>1 Einführung</b>	<b>1</b>
<b>2 Grundlagen</b>	<b>5</b>
<b>2.1 Modellbildung</b>	<b>5</b>
<b>2.2 Computergestützte Simulation</b>	<b>6</b>
2.2.1 Konventionelle Simulationsmethoden	7
2.2.2 Parallelisierungsansätze	8
2.2.2.1 Parallele Durchführung unabhängiger Simulationsläufe	8
2.2.2.2 Auslagerung von Hilfsfunktionen	9
2.2.2.3 Verteilung des Modells	9
<b>2.3 Modelloptimierung</b>	<b>10</b>
2.3.1 Direkte Parameteroptimierung modellbasierter Zielfunktionen	11
2.3.2 Vereinfachende Annahmen	15
2.3.3 Direkte Optimierungsverfahren	16
2.3.3.1 Lokale Optimierung	16
2.3.3.2 Globale Optimierung	19
<b>2.4 Web-Technologien</b>	<b>21</b>
2.4.1 Grundlegendes zum Internet	22
2.4.2 Das World Wide Web	24
2.4.3 Auszeichnungssprachen	24
2.4.3.1 Standard Generalised Markup Language	25
2.4.3.2 Hypertext Markup Language	25
2.4.3.3 Extensible Markup Language	26
2.4.4 Common Gateway Interface	27
2.4.5 Java	28

2.4.6 Web3D-Technologien	31
<b>2.5 Komponenten-Technologien</b>	<b>32</b>
2.5.1 Konzepte zum strukturierten Softwareentwurf	33
2.5.2 Komponentenorientierte Softwareentwicklung	34
2.5.2.1 Der Komponentenbegriff	35
2.5.2.2 Komponentenstandards	36
2.5.2.3 Frameworks	41
2.5.2.4 Middleware-Systeme	41
2.5.3 Common Object Request Broker Architecture	43
2.5.3.1 Object Management Architecture	43
2.5.3.2 Die ORB-Architektur	45
2.5.3.3 ORB-Interoperabilität	46
<b>3 Web-Technologien in der Modellierung und Simulation</b>	<b>49</b>
<b>3.1 Einführung</b>	<b>49</b>
<b>3.2 Standardisierte Modellaustauschformate</b>	<b>51</b>
<b>3.3 Online-Simulation und -Animation</b>	<b>51</b>
3.3.1 Architekturkonzepte für Simulationsanwendungen im Web	52
3.3.1.1 Client-seitige Ausführung	52
3.3.1.2 Server-seitige Ausführung	53
<b>3.4 Java-basierte Simulationspakete</b>	<b>54</b>
3.4.1 Grundsätzlicher Aufbau	56
3.4.2 Anwendungsfelder	56
<b>3.5 Web-basierte Simulationsumgebungen</b>	<b>57</b>
<b>3.6 3D-Visualisierung</b>	<b>58</b>
<b>4 Komponenten-Technologien in der Modellierung und Simulation</b>	<b>61</b>
<b>4.1 Aufbau verteilter komponentenorientierter Simulationen mit der High Level Architecture</b>	<b>61</b>
4.1.1 Entwicklungsgeschichte der HLA	62



---

4.1.2	Architekturüberblick	63
4.1.3	Der HLA-Regelsatz	64
4.1.4	Die Laufzeit-Infrastruktur	65
4.1.5	Bestehende Probleme	66
<b>4.2</b>	<b>Komponentenorientierter Aufbau von Modellierungswerkzeugen</b>	<b>67</b>
4.2.1	Klassische Modellierungswerkzeuge	67
4.2.2	Entwurf einer komponentenorientierten Softwarearchitektur für Modellierungswerkzeuge	70
4.2.3	Werkzeugkomponenten aus fachlicher Sicht	72
4.2.3.1	Modelleditoren	73
4.2.3.2	Komponenten zur Modellauswertung	73
4.2.3.3	Spezialkomponenten	74
4.2.3.4	Abhängigkeiten zwischen den Fachkomponenten	75
4.2.3.5	Beschreibung der Modellierungsdaten	76
4.2.4	Mögliche Realisierung durch technische Komponenten	77
4.2.5	Einbindung existierender Werkzeuge	78
<b>5</b>	<b>Konzeption, Realisierung und Bewertung einer Optimierungskomponente</b>	<b>81</b>
<b>5.1</b>	<b>Effiziente Verfahren zur direkten Parameteroptimierung von Simulationsmodellen</b>	<b>83</b>
5.1.1	Atomare direkte Optimierungsverfahren	83
5.1.2	Kombinierte 2-Phasen Optimierung	86
5.1.3	Mehrstufige Optimierung	87
5.1.4	Spezielle Maßnahmen zur Reduktion des Optimierungsaufwands	89
5.1.4.1	Vermeidung von Reevaluation	90
5.1.4.2	Mehrfachstart der Feinoptimierung	90
5.1.4.3	Beschleunigung der Voroptimierung	91
5.1.5	Behandlung irreführender Probleme	93
5.1.6	Umgang mit stochastischem Rauschen	96
<b>5.2</b>	<b>Realisierung der Optimierungskomponente</b>	<b>96</b>

<b>5.3 Bewertung der Optimierungsverfahren</b>	<b>99</b>
5.3.1 Leistungskenngrößen	99
5.3.2 Testprobleme	101
5.3.3 Optimierungsergebnisse	105
5.3.3.1 Ergebnisse zu Testproblem $(L_1^n, F_1^n)$	106
5.3.3.2 Ergebnisse zu Testproblem $(L_2^n, F_2^n)$	113
5.3.3.3 Ergebnisse zu Testproblem $(L_3^n, F_3^n)$	117
5.3.3.4 Ergebnisse zu Testproblem $(L_4^n, F_4^n)$	119
<b>5.4 Zusammenfassung</b>	<b>122</b>
<b>5.5 Mögliche Weiterentwicklungen</b>	<b>124</b>
<b>6 Projektbeschreibungen</b>	<b>127</b>
<b>6.1 Komponentenorientierter Aufbau eines Web-basierten Werkzeugs zur dynamischen Prozessoptimierung</b>	<b>127</b>
<b>6.2 Online-Animation direkter Optimierungsverfahren</b>	<b>131</b>
<b>6.3 Verteilte Simulation und Visualisierung künstlichen Lebens</b>	<b>135</b>
<b>7 Schlussfolgerungen</b>	<b>145</b>
<b>7.1 Auswirkungen innerhalb der Modellierung und Simulation</b>	<b>145</b>
7.1.1 Chancen	145
7.1.2 Risiken	147
<b>7.2 Auswirkungen auf andere Bereiche</b>	<b>148</b>
<b>Literaturverzeichnis</b>	<b>151</b>

Modelle und Simulationen stellen seit jeher wichtige Hilfsmittel des Menschen zum Umgang mit der Realität dar. Bereits vor Tausenden von Jahren wurden große Bauwerke, Schiffe und Maschinen zunächst als verkleinerte physikalische Nachbildungen gebaut und geprüft, bevor sie in großem Maßstab erstellt und ihrer Bestimmung zugeführt wurden. Der Computer revolutionierte die Simulation, indem er die Möglichkeit der Modellbildung auf alles erweiterte, was sich in irgendeiner Weise formalisieren und damit rechenfähig darstellen ließ. Durch leistungsfähige Rechner ließen sich auch sehr komplexe, mathematisch nicht mehr faßbare Sachverhalte in einem akzeptablen Zeitrahmen darstellen und in ihrem Verhalten beschreiben.

Abb. 1.1 zeigt die wichtigsten Entwicklungsphasen der computergestützten Modellierung und Simulation, die mittlerweile ca. 50 Jahre alt geworden ist. Da bislang noch keine allgemein anerkannte Liste von Meilensteinen zur Entwicklungsgeschichte der Simulation existiert, gibt die in Abb. 1.1 dargestellte Unterteilung die subjektive, hauptsächlich von der ereignisorientierten Simulation geprägte Sicht des Autors wieder. Eine etwas detaillierte Beschreibung der historischen Entwicklung der computergestützten Simulation findet sich in [Lore99].

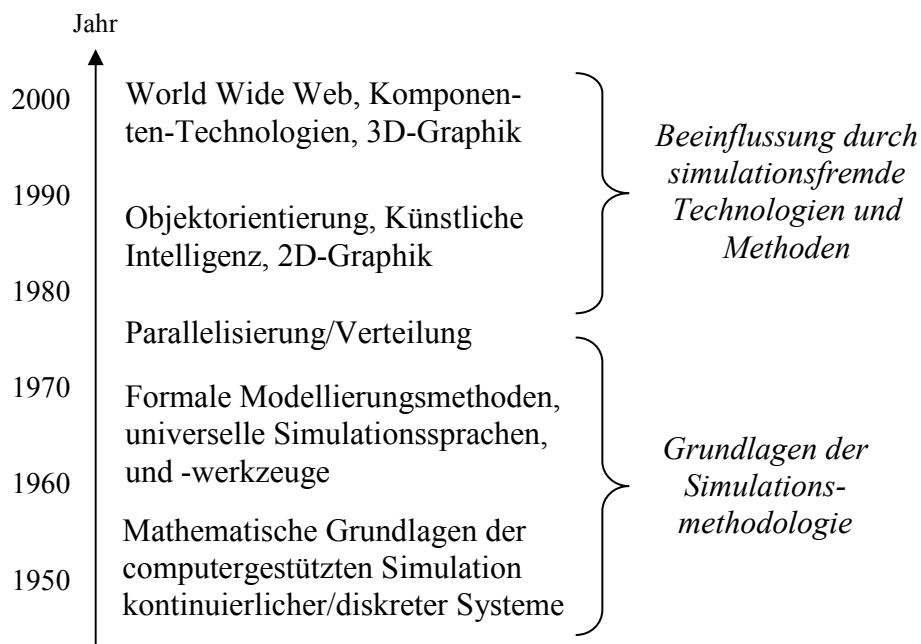


Abb. 1.1: Entwicklungsphasen in der computergestützten Modellierung und Simulation

Die genaue Geburtsstunde der computergestützten Simulation lässt sich nur schwer bestimmen, da es sich bei den ersten gegen Ende der vierziger Jahre durchgeführten Simulationen wohl um kernphysikalische Modellexperimente im Umfeld der US-amerikanischen Kernwaffenentwicklung gehandelt haben dürfte, die noch heute weitgehend militärischer Geheimhaltung unterliegen. Ausgehend von diesen ersten auf Digitalrechnern durchgeführten Simulationsexperimenten wurden bis in die späten fünfziger Jahre zunächst einmal wichtige mathematische Grundlagen der computergestützten Simulation kontinuierlicher und diskreter Systeme erarbeitet. Dazu zählen vor allem statistische Methoden sowie Verfahren zur algorithmischen Erzeugung von Pseudozufallszahlen. In der darauf folgenden Phase, die bis Anfang der siebziger Jahre reichte, wurden formale Modellierungsmethoden wie beispielsweise Warteschlangen- und Petrinetze entwickelt. Darüber hinaus gingen aus dieser Phase universelle Simulationssprachen wie z.B. GPSS (General Purpose Simulation System) und SIMULA<sup>1</sup> sowie darauf aufbauende Simulationswerkzeuge hervor. Diese zum Teil sehr mächtigen Werkzeuge ermöglichten es dem Modellierer, weitaus komplexere Modelle als bisher zu implementieren, was schließlich dazu führte, dass man sehr bald an die Grenzen der Leistungsfähigkeit der damals zur Verfügung stehenden Rechner stieß und sich verstärkt der Parallelisierung und Verteilung von Simulationen zuwandte.

Ende der siebziger Jahre als die heute bekannte Simulationsmethodologie bereits in ihren wesentlichen Grundzügen feststand, begann für den Bereich der computergestützten Simulation der nächste bedeutende und bis heute andauernde Entwicklungsabschnitt. Wesentliche Impulse für die nun folgenden Entwicklungsphasen kamen zum einen aus den vielfältigen praktischen Simulationsanwendungen, die sich in den meisten Wissenschafts- und Ingenieursbereichen sehr schnell als ein wichtiges, teilweise sogar unverzichtbares Hilfsmittel etablierten. Starken Einfluss hatte zum anderen auch die sich rasant weiterentwickelnde Informationstechnik. In den achtziger Jahren wurde die computergestützte Simulation vor allem durch die nun stark aufkommende Objektorientierung und den Einsatz von Methoden aus der künstlichen Intelligenz geprägt. Darüber hinaus beschäftigte man sich intensiv mit der 2-dimensionalen Darstellung simulierter Abläufe. In der darauf folgenden Entwicklungsphase, die sich durch die gesamten neunziger Jahre bis heute erstreckt, wurde die Simulation grundlegend durch das zunehmend an Bedeutung gewinnende Internet sowie den auf der Objektorientierung aufsetzenden Komponenten- und Agententechnologien beeinflusst. Des Weiteren ermöglichte die enorm angestiegene Rechenleistung nun auch die 3-dimensionale Darstellung simulierter Abläufe.

Die vorliegende Arbeit gibt einen umfassenden Überblick über die jüngste Entwicklungsphase der computergestützten Modellierung und Simulation, wobei der Schwerpunkt der Betrachtungen auf dem Einsatz von Web- und Komponenten-Technologien liegt<sup>2</sup>. Diese beiden Schlüsseltechnologien lassen sich nicht nur innerhalb der Modellierung und Simulation gewinnbringend einsetzen, sondern eröffnen darüber hinaus auch die Möglichkeit, äußerst fruchtbare Verbindungen zu anderen Bereichen herzustellen. In diesem Zusammenhang wird im Detail auf die Kopplung von Modellierungsmethoden mit Verfahren aus der Optimierung eingegangen. Diese Kopplung, die darauf abzielt, die heute noch weitgehend manuell durchgeführte Suche nach optimalen Modellparametereinstellungen zu automatisieren, erfordert neben technischen Infrastrukturmaßnahmen vor allem auch methodische Weiterentwicklungen auf dem Gebiet der Optimierung. Die Entwicklung von Verfahren, die den hohen Anforderungen und zahlreichen

---

<sup>1</sup> Die Bedeutung von SIMULA ist vor allem darauf zurückzuführen, dass bei der Entwicklung dieser Programmiersprache erstmalig objektorientierte Konzepte umgesetzt wurden.

<sup>2</sup> Auf die Problematik der 3-dimensionalen Darstellung simulierter Abläufe wird nur am Rande eingegangen.

Randbedingungen der Modelloptimierung genügen, deren konkrete Realisierung in Form einer universell einsetzbaren Optimierungskomponente sowie deren Bewertung bilden weitere Themenschwerpunkte dieser Arbeit.

Kapitel 2 führt in die Grundlagen dieser Arbeit ein, wobei zunächst auf den Prozess der Modellbildung eingegangen wird. Danach wird ein Überblick über die grundlegenden Methoden der computergestützten Simulation gegeben und darüber hinaus die Problematik der Optimierung modellbasierter Zielfunktionen angesprochen. Abschließend werden die heute zur Verfügung stehenden Web- und Komponenten-Technologien vorgestellt.

Kapitel 3 geht im Detail auf die vielfältigen Einsatzmöglichkeiten von Web-Technologien in der computergestützten Modellierung und Simulation ein. Zu den Themenschwerpunkten dieses Kapitels zählen XML-basierte Modellaustauschformate, Architekturkonzepte für Simulationsanwendungen im Web, Java-basierte Simulationspakete, Web-basierte Simulationsumgebungen sowie der Einsatz von Web3D-Technologien zur 3-dimensionalen Visualisierung simulierter Abläufe.

Kapitel 4 setzt sich mit dem komponentenorientierten Aufbau verteilter Simulationsmodelle und -werkzeuge auseinander. Zunächst wird mit der vom US-amerikanischen Verteidigungsministerium entwickelten High Level Architecture eine Komponenteninfrastruktur vorgestellt, die speziell auf die Problematik heterogener verteilter Simulation zugeschnitten ist. Im Anschluss daran befasst sich dieses Kapitel mit der Konzeption und Realisierung komponentenorientierter Modellierungswerkzeuge basierend auf universellen Komponentenstandards.

Kapitel 5 beschäftigt sich eingehend mit der Entwicklung von Spezialkomponenten, die zur Erweiterung der Funktionalität komponentenorientierter Modellierungswerkzeuge dienen. Von großer Bedeutung sind heute vor allem Spezialkomponenten zum zielgerichteten Experimentieren mit Modellen. Aufgabe dieser Komponenten ist es, aus meist sehr komplexen Modellen für den Modellierer relevante Informationen<sup>3</sup> zu extrahieren. Diese Aufgabe ist bei hoher Modellkomplexität meist nicht mehr von Hand durchführbar, sondern es werden dazu systematisch vorgehende Experimentierstrategien benötigt. In diesem Kapitel werden leistungsfähige Methoden zur Ermittlung optimaler Parametereinstellungen von Simulationsmodellen vorgestellt und diskutiert. Darüber hinaus wird im Detail auf die Konzeption, Realisierung und Bewertung einer universell einsetzbaren Optimierungskomponente eingegangen.

In Kapitel 6 wird über drei Forschungsprojekte berichtet, in denen die in dieser Arbeit vorgestellten Methoden und Technologien konkret angewendet wurden.

Kapitel 7 macht abschließend den Versuch, die Auswirkungen des Einsatzes von Web- und Komponenten-Technologien innerhalb der Modellierung und Simulation zusammenfassend darzustellen. Darüber hinaus wird aufgezeigt, dass die dort stattgefundenen Veränderungen das Potential haben, auch andere Bereiche nachhaltig zu beeinflussen.

---

<sup>3</sup> beispielsweise über optimale Parametereinstellungen, Parameter hoher Empfindlichkeit, etc.



In diesem Kapitel werden die für die vorliegende Arbeit relevanten Grundlagen eingeführt. In Abschnitt 2.1 wird zunächst auf den Prozess der Modellbildung eingegangen. Im Anschluss daran wird in Abschnitt 2.2 ein Überblick über die grundlegenden Methoden der computergestützten Simulation gegeben. Abschnitt 2.3 befasst sich mit dem innerhalb der Modellierung und Simulation zunehmend an Bedeutung gewinnenden Bereich der Optimierung modellbasierter Zielfunktionen. Die hier vorgestellten Optimierungsmethoden sind Ausgangspunkt für die Entwicklung der in Kapitel 5 beschriebenen Optimierungskomponente. In den beiden letzten Abschnitten werden die zurzeit im Bereich der Modellierung und Simulation am häufigsten eingesetzten Web- und Komponenten-Technologien in ihren Grundzügen vorgestellt. Auf die vielfältigen Anwendungsmöglichkeiten dieser Technologien innerhalb der Modellierung und Simulation wird ausführlich in den nachfolgenden Kapiteln eingegangen.

## **2.1 Modellbildung**

Eine naheliegende Möglichkeit, um zuverlässige Aussagen über das Verhalten eines Systems zu bekommen, besteht darin, das System selbst unter verschiedenen Bedingungen zu beobachten. Oft sind Experimente am realen System jedoch unangebracht, unzulässig oder sogar unmöglich. In diesen Fällen bietet es sich an, anstelle des Realsystems mit einem Modell zu arbeiten. Bei einem Modell handelt es sich selbst wieder um ein System, das jedoch eine vereinfachte generalisierte Abbildung des Ursprungssystems darstellt. Der Vorgang, bei dem Systeme auf Modelle abgebildet werden, wird Modellbildung genannt. Dabei ist darauf zu achten, dass das Modell all die Systemeigenschaften aufweist, die zum Erreichen der Untersuchungsziele wesentlich sind. Je nach Zielsetzung der Modellstudie und der subjektiven Sichtweise des Modellierers können zu einem gegebenen System mehrere unterschiedliche Modelle erstellt werden. Eine Modellstudie umfasst üblicherweise die in Abb. 2.1.1 dargestellten Etappen oder Arbeitsschritte. Die Arbeitsschritte einer Modellstudie laufen nicht immer streng sequentiell ab. Häufig kommt es vor, dass mehrere Arbeitszyklen (Iterationen) erforderlich sind. Weitere Informationen zur Modellbildung finden sich in [Lehm77], [Schm85], [Page91], [Boss92], [Fish95] und [Fish01].

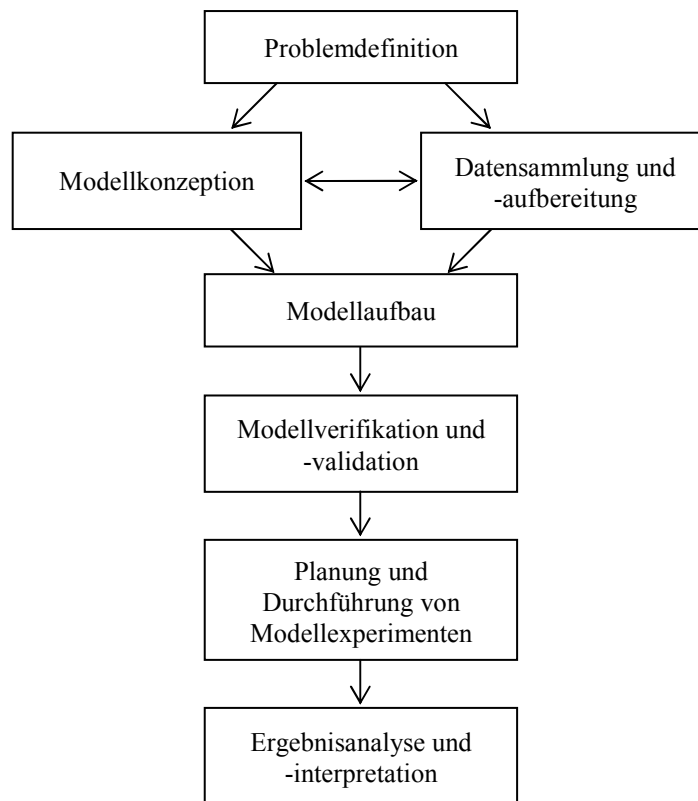


Abb. 2.1.1: Bei einer Modellstudie durchzuführende Arbeitsschritte

Die in Abbildung 2.1.1 beschriebene Vorgehensweise trifft grundsätzlich auf alle Arten von Modellen zu. Die nun folgenden Betrachtungen beschränken sich auf Modelle, die mit Hilfe von Computern erstellt und auf diesen zur Ausführung gebracht werden.

## 2.2 Computergestützte Simulation

Unter computergestützter Simulation<sup>1</sup> wird in der Informatik die Nachbildung realer Vorgänge mittels Computern auf der Grundlage formaler (analytischer, numerischer oder graphischer) Modelle verstanden [ScWe01]. Simulationsmodelle stellen somit immaterielle Ersatzobjekte dar, deren Analyse und Auswertung darauf abzielt, neue (auf die Wirklichkeit übertragbare) Erkenntnisse zu gewinnen. Ein großer Vorteil dieser Vorgehensweise besteht darin, dass keine materielle Umsetzung des zu beobachtenden Systems erforderlich ist. Darüber hinaus können komplexe Systeme aller Art unter den verschiedensten Bedingungen untersucht werden.

Heute ist die rechnergestützte Simulation nicht nur für sämtliche Ingenieursdisziplinen von fundamentaler Wichtigkeit, sondern darüber hinaus auch für viele andere Bereiche, die mit der Dynamik komplexer nichtlinearer Systeme konfrontiert sind. Aufgrund ihres interdisziplinären Charakters handelt es sich bei der computergestützten Simulation um keine feststehende Me-

---

<sup>1</sup> oder kurz Computersimulation



thode, sondern um eine Methodik oder Vorgehensweise, die je nach Problemstellung und Anwendungsbereich sehr unterschiedlich ausfallen kann. Grundsätzlich beginnt jede Simulationsstudie mit der Entwicklung eines Simulationsmodells, das die wesentlichen Eigenschaften der zu simulierenden Vorgänge und ihre gegenseitige Beeinflussung widerspiegelt (siehe dazu auch Abb. 2.1.1). Anschließend wird mit dem Simulationsmodell experimentiert, wobei es auf einem Rechner zur Ausführung gebracht wird. Die dabei anfallenden Ergebnisse werden abschließend analysiert, interpretiert und gegebenenfalls auf die Wirklichkeit übertragen.

Zur Ausführung von Simulationen haben sich heute digitale Rechner durchgesetzt, bei denen die Untersuchung kontinuierlicher Vorgänge aus der Realität anhand vereinfachter diskretisierter Darstellungen vorgenommen wird. Wie bereits angedeutet existiert heute eine Vielzahl an Simulationsmethoden für die unterschiedlichsten Aufgabenstellungen. In Abschnitt 2.2.1 wird zunächst auf die prinzipielle Arbeitsweise konventioneller Simulationsmethoden eingegangen. Abschnitt 2.2.2 gibt einen Überblick über Ansätze zur parallelen bzw. verteilten Abarbeitung von Simulationen.

## 2.2.1 Konventionelle Simulationsmethoden

Konventionelle Simulationsmethoden lassen sich zunächst einmal anhand der Art und Weise unterscheiden, wie die simulierte Realzeit, die im Folgenden auch als Simulationszeit bezeichnet wird, voranschreitet. Bei kontinuierlichen Simulationen verstreicht die Simulationszeit und damit die Folge der Zustandsänderungen der im Simulationsmodell agierenden Objekte kontinuierlich. Solche Modelle bestehen meist aus einem System von Differentialgleichungen, deren freie Variable die Zeit ist. Bei einem sprunghaften Voranschreiten der Zeit von einem Zeitpunkt zum nächsten spricht man dagegen von diskreter Simulation. Diskrete Simulationen, bei denen Zustandswechsel nur zu diskreten Zeitpunkten auftreten können, lassen sich weiter in zeit- und ereignisgesteuerte Simulationen unterteilen. Bei der zeitgesteuerten Simulation wird die Simulationszeit in Inkrementen fester oder variabler Schrittlänge vorangeschaltet, während bei der ereignisgesteuerten Simulation die Simulationszeit nur dann fortgeschaltet wird, wenn der Modellzustand durch ein Ereignis verändert wurde. Bei Ereignissen, welche die Veränderung eines Objektzustandes zu einem bestimmten Zeitpunkt bewirken, kann zwischen exogenen und endogenen Ereignissen unterschieden werden. Liegen exogene Ereignisse vor, ist der Ereigniszeitpunkt von der Systemumgebung vorgegeben (z.B. Eintreffen eines Auftrags in einer Bedienstation). Endogene Ereignisse treten dagegen als Folge von Zustandsänderungen ein, wobei die Ereigniszeitpunkte determiniert oder stochastisch sein können (z.B. Ende der Bedienung eines Auftrags). Da bei der ereignisorientierten Simulation die Ereignisse das Voranschalten der Simulation steuern, erweist sich diese Vorgehensweise vor allem dann als vorteilhaft, wenn sehr unterschiedliche Ereignisdichten (Anzahl von Ereignissen pro Zeiteinheit) im Zeitverlauf auftreten. Zum einen wird jedes Ereignis zum exakten Zeitpunkt seines Auftretens berücksichtigt und zum anderen werden die Zeitintervalle zwischen zwei Ereignissen (Totzeiten), die unter Umständen sehr lange ausfallen können, einfach übersprungen. Bei der zeitgesteuerten Simulation müssen solche Totzeiten dagegen Berücksichtigung finden und Ereignisse, die in das Intervall zwischen zwei Beobachtungszeitpunkten fallen, können nicht mehr unterschieden werden.

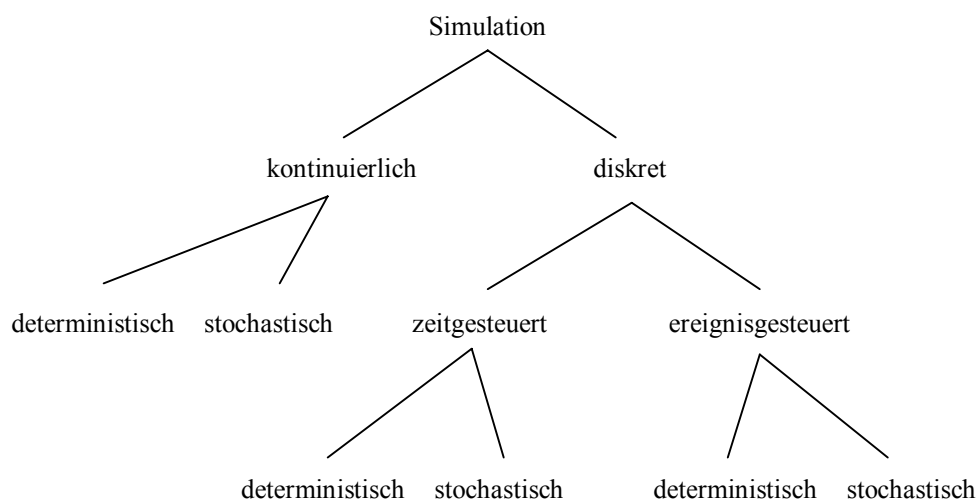


Abb. 2.2.1.1: Klassifikation von Simulationsmethoden

Abb. 2.2.1.1 fasst die oben beschriebene Klassifikation von Simulationsmethoden noch einmal zusammen. Detaillierte Beschreibungen dieser Methoden finden sich in [Page91]. Abschließend sei angemerkt, dass in dieser Arbeit im Wesentlichen diskrete ereignisorientierte Simulationsmodelle betrachtet werden. Die in Kapitel 3 und 4 beschriebenen Methoden und Konzepte sind jedoch problemlos auch auf andere Modellarten übertragbar.

## 2.2.2 Parallelisierungsansätze

Komplexe Simulationsexperimente stellen i.a. enorme Anforderungen an Rechenleistung und Hauptspeicherkapazität. Zur Beschleunigung von sequentiellen Simulationen wurde eine Reihe von Parallelisierungsansätzen entwickelt, auf die im Folgenden näher eingegangen wird. Wird zur Parallelisierung ein Multiprozessorsystem verwendet, spricht man von paralleler Simulation. Bei Verwendung eines Verbundes miteinander gekoppelter Rechner spricht man dagegen von verteilter Simulation.

### 2.2.2.1 Parallele Durchführung unabhängiger Simulationsläufe

Häufig werden ganze Serien unabhängiger Simulationsexperimente durchgeführt, beispielsweise bei der Modellvalidierung, bei der Bestimmung von Konfidenzintervallen oder bei der Optimierung von Modellparametern. Solche Serien unabhängiger Einzelexperimente können sehr leicht dadurch beschleunigt werden, dass mehrere Rechner zur Modellausführung verwendet werden. Besteht jedoch eine Abhängigkeit zwischen den durchzuführenden Experimenten, ist diese Vorgehensweise nicht mehr anwendbar. In diesem Fall benötigt man Methoden zur Beschleunigung von Einzelexperimenten. Methoden dieser Art werden in den beiden folgenden Abschnitten näher erläutert.

### 2.2.2.2 Auslagerung von Hilfsfunktionen

Ein möglicher Ansatz ein Simulationsexperiment zu beschleunigen, besteht in der Auslagerung häufig verwendeter Hilfsfunktionen<sup>2</sup> auf zusätzliche Prozessoren bzw. Rechner. Der große Vorteil dieser Vorgehensweise besteht darin, dass ein für die sequentielle Simulation ausgelegtes Modell nicht umgeschrieben werden muss. Signifikante Beschleunigungen sind durch diese Maßnahme jedoch nicht zu erwarten, da sich herausgestellt hat, dass die Berechnung von Hilfsfunktionen meist nur einen geringen Anteil der Gesamtrechnzeit beansprucht.

### 2.2.2.3 Verteilung des Modells

Weitaus viel versprechender als die Auslagerung von Hilfsfunktionen ist es, die Parallelisierung auf Modellebene vorzunehmen. Dabei ergeben sich im Wesentlichen folgende Problemstellungen:

- Aufteilung des Gesamtmodells in kooperierende Teilmodelle, die möglichst unabhängig voneinander sein sollten, um den anfallenden Kommunikations- und Synchronisationsaufwand gering zu halten.
- Geeignete Zuordnung der Teilmodelle auf Prozessoren (parallele Simulation) bzw. Rechner (verteilte Simulation). Falls ein Prozessor bzw. Rechner mehrere Teilmodelle ausführen soll, ist zusätzlich eine geeignete Auswahlstrategie (Scheduling) erforderlich.
- Synchronisation der Abläufe in den Teilmodellen zur Wahrung der Kausalität.

Die Partitionierung des Modells hat erheblichen Einfluss auf den zu erzielenden Gewinn an Ausführungsgeschwindigkeit. Um eine hohe Beschleunigung zu erreichen, sollte das Modell so zerlegt werden, dass zwischen den einzelnen Teilmodellen möglichst wenige Abhängigkeiten bestehen. Bei zu eng gekoppelten Teilmodellen ist durch den anfallenden Kommunikations- und Synchronisationsaufwand eine Beschleunigung der Simulation eher unwahrscheinlich, wobei es in Extremfällen sogar zu einer Verlangsamung gegenüber der sequentiellen Ausführung kommen kann. Die Modellpartitionierung muss heute noch weitgehend manuell vom Modellierer durchgeführt werden, da entsprechende Methoden nur für einige wenige Spezialfälle zur Verfügung stehen. Analog zum Modellierungsvorgang ist das Ergebnis einer solchen Partitionierung stark von der Erfahrung und Intuition des Modellierers abhängig.

Nach erfolgter Modellpartitionierung müssen die einzelnen Teilmodelle auf die zur Verfügung stehenden Prozessoren bzw. Rechner verteilt werden. Da die Anzahl der bereitstehenden Abarbeitungsressourcen meist weitaus geringer ist als die der auszuführenden Teilmodelle, muss eine geeignete Zuordnung der Teilmodelle auf diese Ressourcen gefunden werden. Zur Lösung dieser Problemstellung existiert heute eine breite Auswahl an so genannten Scheduling-Algorithmen, die sich grob in statische und dynamische Verfahren unterteilen lassen. Während bei den statischen Verfahren die Zuordnung bereits vor dem Start der Simulation erfolgen muss und während des gesamten Simulationsexperiments unverändert bleibt, erlaubt das dynamische Scheduling eine flexible Zuordnung zur Laufzeit.

---

<sup>2</sup> beispielsweise die Erzeugung von Pseudozufallszahlen oder die Generierung von Statistiken

Die dritte Problemstellung, die sich bei der Parallelisierung auf Modellebene ergibt, besteht in der Synchronisation der einzelnen Teilmodelle. Geht man von der diskreten ereignisorientierten Simulation aus, muss die Kausalordnung zwischen den von mehreren parallel ablaufenden Simulatoren erzeugten und auszuführenden Ereignissen entsprechend der vom Benutzer vorgegebenen Modellbeschreibung eingehalten werden. Dieses Problem wird entscheidend dadurch erschwert, dass diese Kausalordnung nur implizit definiert ist und sich die Menge der auszuführenden Ereignisse erst im Verlauf der Simulation ergibt. Abb. 2.2.2.3.1 gibt einen Überblick über die heute existierenden Methoden zur Lösung des Synchronisationsproblems bei verteilt/parallel ablaufenden diskreten ereignisgesteuerten Simulationen.

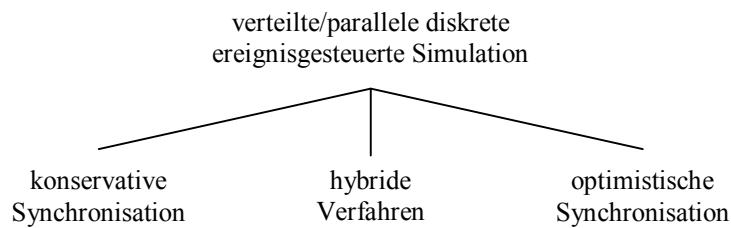


Abb. 2.2.2.3.1: Methoden zur Lösung des Synchronisationsproblems

Die Grundidee konservativer Synchronisationsverfahren besteht darin, Kausalitätsverletzungen von vornherein zu vermeiden. Ein Ereignis darf erst dann ausgeführt werden, wenn sichergestellt ist, dass es kausal unabhängig von allen anderen noch zukünftig auszuführenden Ereignissen ist. Um dies zu erreichen, werden Garantien ausgetauscht, die das vereinheitlichende zentrale Element konservativer Verfahren darstellen [Mehl94]. Im Gegensatz zu konservativen Synchronisationsverfahren werden bei optimistischer Synchronisation Ereignisse ohne Vorabgarantie über die Einhaltung der Kausalordnung ausgeführt. Eventuell auftretende Kausalitätsverletzungen werden dadurch behoben, dass die betroffenen Komponenten auf einen Zustand vor der Kausalitätsverletzung zurückgesetzt werden. Zwischen diesen beiden grundlegenden Synchronisationsverfahren reihen sich eine Vielzahl von Abwandlungen und Mischformen ein, die in Abb. 2.2.2.3.1 als hybride Verfahren bezeichnet werden. Eine ausführliche Beschreibung der heute existierenden Synchronisationsmethoden findet sich in [Mehl94] und [Fuji00].

## 2.3 Modelloptimierung

Nachdem ein Modell erstellt und sorgfältig überprüft wurde, kann damit begonnen werden, die zum Erreichen der Modellierungsziele erforderlichen Experimente mit dem Modell durchzuführen. Im Bereich des Ingenieurwesens werden Modelle i.a. zur Unterstützung des Systementwurfs eingesetzt, dessen globale Zielsetzung darin besteht, das zu entwerfende System "optimal" hinsichtlich der drei Entwurfsdimensionen Kosten, Leistung und Zuverlässigkeit auszuliegen. Optimal auslegen oder kurz "optimieren" bedeutet, unter mehreren möglichen Realisationen in Bezug auf bestimmte Kriterien eine bessere oder die beste - optimale - Realisierungsalternative zu bestimmen.

Zur Systemoptimierung existieren sehr unterschiedliche Vorgehensweisen. Eine naheliegende Möglichkeit besteht darin, mit dem realen System selbst zu experimentieren und zu versuchen,

durch schrittweise Änderung der Systemparameter und/oder der Systemstruktur das Verhalten des Systems zu verbessern. Grundvoraussetzung für diese Vorgehensweise ist jedoch, dass das betreffende System bereits existiert und zum Experimentieren geeignet ist. Andernfalls bleibt nur die Möglichkeit, die Optimierung anhand eines abstrakten Systemmodells durchzuführen, welches das Verhalten des realen Systems in ausreichendem Maße widerspiegelt. Abstrakte Modelle sind verglichen mit Realsystemen zum Experimentieren weitaus besser geeignet, da sie uneingeschränkt zugänglich sind und in ihrer Struktur und ihren Abläufen nach Belieben und ohne jegliches Gefahrenpotential manipuliert und modifiziert werden können.

Bei der Modelloptimierung ist die Zielfunktion, die es zu optimieren gilt, durch ein Modell vorgegeben. Zur Optimierung solcher modellbasierter Zielfunktionen existieren grundsätzlich folgende Alternativen:

- Indirekte (analytische) Optimierung: Bei der indirekten Optimierung wird ausgehend von einer formalen Modellbeschreibung das Optimierungsproblem mathematisch formuliert und durch geeignete mathematische Verfahren gelöst (klassische analytische Vorgehensweise). Empirisches Experimentieren mit dem Modell ist nicht notwendig.
- Direkte (empirische) Optimierung: Bei der direkten Optimierung wird keine mathematische Beschreibung der zu optimierenden Zielfunktion vorausgesetzt, sondern lediglich deren Berechenbarkeit. Zur Steuerung des Optimierungsprozesses werden ausschließlich Zielfunktionswerte verwendet, die durch Simulationsexperimente ermittelt werden (empirische Vorgehensweise).
- Hybride Optimierung: Hier wird versucht, die Vorteile der direkten und indirekten Optimierung zu vereinen und gleichzeitig deren Nachteile möglichst auszuschließen.

Optimierungsmethoden, die eine mathematische Beschreibung des Optimierungsproblems voraussetzen, lassen sich nur sehr eingeschränkt zur Optimierung komplexer Simulationsmodelle verwenden, da sich insbesondere bei diskreten ereignisorientierten Simulationsmodellen eine solche Beschreibung nur in einigen wenigen speziellen Fällen finden lässt. Die nun folgenden Betrachtungen beschränken sich daher auf die Klasse der universell anwendbaren direkten Optimierungsverfahren.

### **2.3.1 Direkte Parameteroptimierung modellbasierter Zielfunktionen**

Im Folgenden wird in die Grundlagen und den Stand der Technik auf dem Gebiet der direkten Parameteroptimierung modellbasierter Zielfunktionen eingeführt. Für den in dieser Arbeit überwiegend betrachteten Fall der Optimierung von Zielfunktionen mit kontinuierlichen Parametern lautet die Definition des globalen Optimierungsproblems<sup>3</sup> wie folgt:

---

<sup>3</sup> Die globale Optimierung hat sich als eine NP-harte Problemstellung herausgestellt.

Gegeben: Lösungsraum  $L \neq \emptyset$ , mit

$$L = \left\{ \bar{x} \in \mathbb{R}^n \mid g_i(\bar{x}) \leq 0, i \in \{1, \dots, p\}; h_j(\bar{x}) = 0, j \in \{1, \dots, q\}; p, q \in \mathbb{N} \right\},$$

mit  $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$  und  $h_j : \mathbb{R}^n \rightarrow \mathbb{R}$  Parameterrestriktionen,

$\mathbb{R}$ : Menge der reellen Zahlen,  $\mathbb{N}$ : Menge der natürlichen Zahlen

Zielfunktion  $F : L \rightarrow \mathbb{R}$ .

Gesucht: Vektor  $\bar{x}^*$ , für den gilt:  $\forall \bar{x} \in L: F(\bar{x}) \circ F(\bar{x}^*) = F^*$ ,  $\circ \in \{\leq, \geq\}$ .

Ein Parameteroptimierungsproblem kann also als ein Tupel  $(L, F)$  aufgefasst werden, wobei das Optimierungsziel darin besteht, optimale Werte für die unabhängigen Variablen (Parameter) der Zielfunktion  $F$  im Lösungsraum  $L$  zu finden. Die Dimension eines Parameteroptimierungsproblems ergibt sich aus der Anzahl der Parameter der Zielfunktion. Der Wert  $F^* = F(\bar{x}^*)$  wird globales Optimum<sup>4</sup> genannt.  $\bar{x}^*$  stellt einen globalen Optimumpunkt dar, wobei beachtet werden sollte, dass neben diesem Punkt durchaus noch weitere globale Optimumpunkte existieren können, an denen die Zielfunktion  $F$  das globale Optimum  $F^*$  erreicht. Bei nicht-linearen Zielfunktionen kann das Parameteroptimierungsproblem durch die Existenz von lokalen Optima erheblich erschwert werden. Ein lokales Optimum  $F^\wedge = F(\bar{x}^\wedge)$  ist wie folgt definiert:

$$\exists \varepsilon > 0, \forall \bar{x} \in L: \|\bar{x} - \bar{x}^\wedge\| < \varepsilon \Rightarrow F(\bar{x}) \circ F(\bar{x}^\wedge) = F^\wedge, \circ \in \{\leq, \geq\}.$$

Der Extremwert  $F^\wedge$  wird lokales Optimum genannt. Die Stelle  $\bar{x}^\wedge$ , an welcher der Extremwert  $F^\wedge$  im Lösungsraum  $L$  erreicht wird, ist der dazugehörige lokale Optimumpunkt. Hier ist zu beachten, dass ein globaler Optimumpunkt stets auch das obige Lokalkriterium erfüllt. Punkte, die das obige Lokalkriterium erfüllen, bei denen jedoch nicht bekannt ist, ob sie lokale oder globale Optimumpunkte darstellen, werden im Folgenden mit  $\bar{x}^\bullet$  bezeichnet. Analog dazu kann es sich beim Zielfunktionswert  $F^\bullet = F(\bar{x}^\bullet)$  entweder um ein lokales oder ein globales Optimum handeln.

Existieren keine lokal-optimalen Lösungen, sondern liegt genau eine global-optimale Lösung vor, spricht man von einer unimodalen Zielfunktion. Andernfalls handelt es sich um eine multimodale Funktion.

Die Grundlage zur Bildung der Zielfunktion  $F(\bar{x})$  stellt bei der Optimierung von Simulationsmodellen die in Abb. 2.3.1.1 dargestellte Modellfunktion  $f_M$  dar. I.a. handelt es sich dabei um eine mehrdimensionale Funktion

$$f_M : L \subset \mathbb{R}^n \rightarrow \mathbb{R}^m, \text{ mit } n, m \in \mathbb{N} \text{ (Menge der natürlichen Zahlen),}$$

---

<sup>4</sup> Bei einem Optimum kann weiter unterschieden werden zwischen einem Maximum ( $\circ = \leq$ ) und einem Minimum ( $\circ = \geq$ ).

die  $n$  reellwertige Eingabeparametervektoren  $\vec{x}=(x_1, \dots, x_n)$ ,  $x_i \in \mathbb{R}$ ,  $i \in \{1, \dots, n\}$ ,  $n \in \mathbb{N}$  des Simulationsmodells auf  $m$  reellwertige Modellausgabegrößen

$$a_1 = f_1(x_1, \dots, x_n), \dots, a_m = f_m(x_1, \dots, x_n), a_i \in \mathbb{R}, i \in \{1, \dots, m\}, m \in \mathbb{N}$$

abbildet.

Bei der Erstellung des Simulationsmodells ist einerseits darauf zu achten, dass sämtliche für die vorgegebene Aufgabenstellung relevante Einflussgrößen im Simulationsmodell als entsprechende Modelleingabeparameter repräsentiert sind. Andererseits sollten für die Optimierung unbedeutende Systemeigenschaften aus Aufwandsgründen im Simulationsmodell möglichst unberücksichtigt bleiben.

Eine durch ein Simulationsmodell repräsentierte Zielfunktion wird im Folgenden als simulationsbasierte Zielfunktion bezeichnet. Die Definition einer solchen Zielfunktion gestaltet sich sehr einfach, wenn nach genau einer Modellausgabegröße optimiert wird:

$$F(\vec{x}) = f_i(\vec{x}) = a_i, i \in \{1, \dots, m\}.$$

Schwierigkeiten treten i.a. dann auf, wenn nach mehreren Modellausgabegrößen gleichzeitig optimiert wird und dabei gegenläufige Zielsetzungen verfolgt werden (Polyoptimierungssituation). In solchen Fällen ist es häufig sehr schwer, die Einzelkriterien so zu gewichten, dass sich eine "vernünftige" Gesamtzielsetzung ergibt. Ein möglicher Ansatz zur Definition der Zielfunktion besteht also darin, sowohl die Ausgabegrößen des Simulationsmodells als auch deren Gewichte  $\omega_i \in \mathbb{R}$ ,  $i \in \{1, \dots, m\}$  zu berücksichtigen:

$$F(\vec{x}) = \varphi(\omega_1, \dots, \omega_m, a_1, \dots, a_m), \text{ mit } \omega_i \in \mathbb{R}.$$

Weitere Ansätze zur Festlegung der Zielfunktion finden sich in [FoF197]. In der Praxis trifft man sehr häufig auf Zielfunktionen, die gewichtete Summen verschiedener Modellausgabegrößen darstellen:

$$F(\vec{x}) = \sum_{i=1}^m \omega_i \cdot a_i, \text{ mit } \omega_i \in \mathbb{R}.$$

Eine der Hauptschwierigkeiten bei der Optimierung von Simulationsmodellen besteht darin, dass meist keine mathematische Beschreibung der modellinternen Zusammenhänge gefunden werden kann (Black-Box Situation). Aus diesem Grund steht auch keine analytische Zusatzinformation über die zu optimierende Zielfunktion zur Verfügung, die sich gewinnbringend zur Beschleunigung des Optimierungsprozesses einsetzen ließe. Wie bereits erwähnt kommen daher für die Optimierung nur direkte Verfahren in Betracht, deren Arbeitsweise auf der abschließlichen Verwendung von Zielfunktionswerten basiert.

Abb. 2.3.1.1 gibt eine Übersicht über den Gesamtablauf bei der direkten Parameteroptimierung von Simulationsmodellen. Hier wird abwechselnd zwischen zwei eigenständigen voneinander getrennten Prozessen hin und her geschaltet. Es handelt sich dabei zum einen um den Optimierungsprozess, der Parametervektoren generiert und zum anderen um den Simulationsprozess, dessen Aufgabe darin besteht, die erzeugten Parametervektoren zu bewerten. Ziel dieses alter-

nierenden Prozesses ist es, die Zielfunktion schrittweise zu verbessern und innerhalb möglichst weniger Iterationsschritte eine Stelle im Lösungsraum zu ermitteln, an der die Zielfunktion das globale Optimum erreicht.

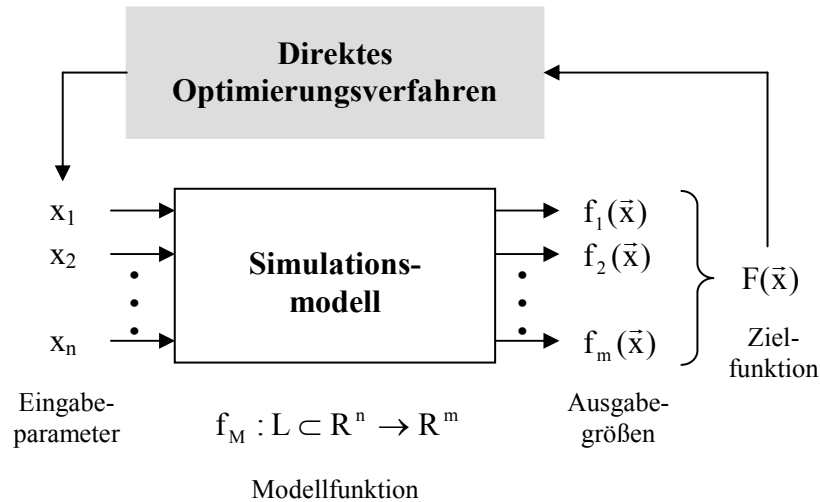


Abb. 2.3.1.1: Direkte Optimierung einer simulationsbasierten Zielfunktion

Neben der Black-Box-Situation weisen simulationsbasierte Zielfunktionen noch einige weitere unschöne Eigenschaften auf, die den Optimierungsprozess erheblich erschweren können:

- Aufwändige Berechnung der Zielfunktion

Die Berechnung der Zielfunktionswerte wird bei simulationsbasierten Zielfunktionen durch simulative Modellauswertung vorgenommen und kann in der Praxis durchaus Rechenzeiten erfordern, die im Minuten- und Stundenbereich liegen. In Extremfällen kann die Auswertung eines Simulationsmodells sogar Tage in Anspruch nehmen.

- Stochastisch verrauschte Zielfunktionswerte

Werden im Simulationsmodell Zufallsgrößen verwendet (stochastisches Simulationsmodell), was sehr häufig der Fall ist, ist die Modellfunktion  $f_M$  und damit auch die Zielfunktion mit einem stochastischen Rauschen überlagert. Je nach Simulationsmodell und ange-setzter Simulationsdauer kann das Rauschen mehr oder weniger stark ausgeprägt sein.

- Hochdimensionaler Lösungsraum mit komplexen Parameterrestriktionen

Bei der Optimierung von Simulationsmodellen aus der Praxis handelt es sich i.a. um hochdimensionale Problemstellungen, deren Lösungsräume häufig durch komplexe (nicht-lineare und/oder unstetige) Restriktionen eingeschränkt sind.



- Komplexer Ergebnisverlauf der Zielfunktion

Simulationsbasierte Zielfunktionen sind sehr häufig multimodal, nichtlinear und unstetig. Außerdem muss mit einem nicht-trivialen variantenreichen Ergebnisverlauf (gewundene Täler, Plateaus, etc.) gerechnet werden.

- Meist nur sehr wenig Vorwissen über den Ergebnisverlauf der Zielfunktion vorhanden

Bei einem komplexen Simulationsmodell ist es i.a. sehr schwierig den Ergebnisverlauf einer darauf basierenden Zielfunktion abzuschätzen. In manchen Fällen lässt sich der Wertebereich durch eine untere und obere Schranke recht gut eingrenzen. Über die genaue Anzahl der vorhandenen Extremstellen sowie die Struktur ihrer Einzugsbereiche können jedoch in den meisten Fällen keine Aussagen gemacht werden.

### 2.3.2 Vereinfachende Annahmen

Zur Lösung komplexer modellbasierter Optimierungsaufgaben haben heute direkte Suchverfahren große Bedeutung erlangt [Wrig95]. Ein großer Vorteil dieser Verfahren liegt in ihrer universellen Anwendbarkeit sowie in der guten Verständlichkeit ihrer Basisprinzipien. Von Nachteil ist jedoch der gegenüber spezialisierten analytischen Optimierungsmethoden meist sehr hohe Rechenaufwand. Dieser ergibt sich bei direkten Suchstrategien aus der Anzahl der zur Lösungsfindung benötigten Zielfunktionsevaluierungen multipliziert mit der für eine Zielfunktionsevaluierung durchschnittlich benötigten Rechenzeit. Insbesondere bei Problemstellungen mit sehr aufwändiger Zielfunktionsevaluierung ist es daher von entscheidender Bedeutung, dass das ausgewählte direkte Optimierungsverfahren mit einer möglichst geringen Anzahl von Zielfunktionsevaluierungen auskommt.

Ein sehr wichtiges Ergebnis theoretischer Analysen zur Leistungsfähigkeit direkter Optimierungsmethoden ist das so genannte "No Free Lunch Theorem" von Wolpert und Macready [WoMa97]. Grob zusammen gefasst besagt dieses Theorem, dass direkte Suchstrategien über alle Optimierungsprobleme gemittelt im Durchschnitt gleich gut arbeiten, wenn es darum geht, das globale Optimum aufzufinden. Anders formuliert gibt es kein direktes Optimierungsverfahren, das allen anderen direkten Verfahren über alle Optimierungsprobleme gemittelt überlegen ist. Demnach spielt es, wenn man über keinerlei problemspezifisches Vorwissen als Lösungshilfe verfügt, keine Rolle, für welches direkte Optimierungsverfahren man sich letztendlich entscheidet.

Dies ist zunächst einmal ein sehr deprimierendes Ergebnis, da die als sehr ineffizient geltende reine Zufallssuche damit den gleichen Stellenwert erhält, wie naturanaloge Verfahren, die den Zufall nicht völlig willkürlich, sondern zielgerichtet einsetzen. Glücklicherweise sind jedoch viele Optimierungsprobleme aus der Praxis so geartet, dass sie sich mit naturanalogen Optimierungsverfahren meist recht gut bearbeiten lassen [DrJW99].

Solche "gutartigen" Problemstellungen liegen i.a. dann vor, wenn

- das zugrunde liegende System, das es zu optimieren gilt, kein chaotisches Verhalten aufweist<sup>5</sup>,
- die Anzahl der Extremstellen der zu optimierenden Zielfunktion überschaubar ist,
- die vorkommenden Extremstellen jeweils einen ausgeprägten Einzugsbereich haben,
- die Ergebnisverläufe der Zielfunktion über den Einzugsbereichen der Extremstellen keine nadelförmigen Spitzen darstellen,
- der Wertebereich der zu optimierenden Zielfunktion grob nach oben bzw. unten abgeschätzt werden kann.

Der praktische Einsatz sowie zahlreiche empirische Untersuchungen haben gezeigt, dass unter den oben genannten Voraussetzungen naturanaloge Verfahren wie z.B. Genetische Algorithmen, Evolutionsstrategien oder das Simulated Annealing weitaus effizienter arbeiten als etwa die reine Zufallssuche. Ergebnisse einer umfangreichen empirischen Leistungsanalyse, die diese Aussage bestätigen, finden sich in Abschnitt 5.3.3.

## 2.3.3 Direkte Optimierungsverfahren

In den letzten Jahrzehnten ist das Interesse an direkten Optimierungsverfahren stark angestiegen. Die Gründe dafür liegen in dem meist sehr einfachen algorithmischen Aufbau, der universellen Anwendbarkeit sowie den bei gutartigen Problemstellungen recht hohen Erfolgchancen. Auch der teilweise hohe Ressourcenbedarf stellt durch die rasant angestiegene Rechenleistung kein größeres Problem mehr dar. Die heute existierende Vielzahl direkter Optimierungsverfahren lässt sich grob in lokale und globale Verfahren einteilen. Ziel dieses Abschnitts ist es, deren wesentliche Merkmale und Funktionsprinzipien herauszuarbeiten, um damit die Voraussetzungen für die in Abschnitt 5.1 beschriebenen Weiterentwicklungen zu schaffen.

### 2.3.3.1 Lokale Optimierung

Bei der lokalen Optimierung besteht das Ziel darin, ausgehend von einer bestimmten Startlösung  $\vec{x}_{\text{start}} \in L$ , durch schrittweise Verbesserung der Zielfunktion eine in der Nachbarschaft gelegene, so genannte "lokal-optimale" Lösung  $\vec{x}^* \in L$  zu bestimmen. Wesentliche Voraussetzung für die lokale Optimierung ist also eine Vorschrift, die jeder Lösung  $\vec{x} \in L$  eine Menge von Lösungen  $L_{\vec{x}} \subset L$  zuordnet, die zu dieser in gewisser Weise "benachbart" sind. Diese Zuordnungsvorschrift

$$\eta : L \rightarrow 2^L$$

---

<sup>5</sup> bei technischen Systemen stellt dies eine Grundvoraussetzung dar

wird im Folgenden als Nachbarschaftsstruktur über  $L$  bezeichnet. Die Menge  $L_{\bar{x}}$  heißt Nachbarschaft der Lösung  $\bar{x}$ , wobei jedes  $\bar{y} \in L_{\bar{x}}$  eine zu  $\bar{x}$  benachbarte Lösung darstellt. Des Weiteren soll gelten:  $\bar{y} \in L_{\bar{x}} \Leftrightarrow \bar{x} \in L_{\bar{y}}$ .

Zu einem Optimierungsproblem  $(L, F)$  mit vorgegebener Nachbarschaftsstruktur  $\eta$  ist  $\bar{x}^{\wedge} \in L$  eine lokal-optimale Lösung, falls gilt:

$$\forall \bar{y} \in L_{\bar{x}}: F(\bar{y}) \circ F(\bar{x}^{\wedge}) = F^{\wedge}, \circ \in \{\leq, \geq\}.$$

In Abb. 2.3.3.1.1 ist der Basialgorithmus zur lokalen Optimierung in Form eines Pseudo-Pascal Programms dargestellt. Es handelt sich hier o.B.d.A. um die Version zur lokalen Minimierung. Der Algorithmus startet mit der (meist zufälligen) Auswahl einer Startlösung aus dem Lösungsraum  $L$ . Danach wird in der Nachbarschaft von  $\bar{x}_{\text{start}}$  nach einer besseren Lösung gesucht. Wird eine solche gefunden, wird ausgehend von dieser weitergesucht. Der Algorithmus terminiert, falls in der Nachbarschaft zu einer Lösung keine bessere mehr existiert.

---

```

procedure Lokale_Optimierung;
begin
  Wähle_Startlösung( $\bar{x}_{\text{start}} \in L$ );
   $\bar{x} := \bar{x}_{\text{start}}$ ;
  repeat
    Generiere_benachbarte_Lösung( $\bar{y} \in L_{\bar{x}}$ );
    if  $F(\bar{y}) < F(\bar{x})$  then  $\bar{x} := \bar{y}$ ;
  until  $F(\bar{y}) \geq F(\bar{x}), \forall \bar{y} \in L_{\bar{x}}$ ;
end;
```

---

*Abb. 2.3.3.1.1: Pseudo-PASCAL Programm zur lokalen Optimierung*

Insbesondere in Lösungsräumen, in denen es zu einer Lösung unendlich viele benachbarte Lösungen gibt (beispielsweise im  $\mathbb{R}^n$ ), kann die Definition einer geeigneten Nachbarschaftsstruktur erhebliche Probleme bereiten. Da in der Praxis natürlich nur Nachbarschaftsstrukturen in Frage kommen, die eine eng begrenzte Auswahl benachbarter Lösungen berücksichtigen, kann es in solchen Lösungsräumen vorkommen, dass unter Umständen lediglich eine Annäherung an eine (lokal-)optimale Lösung erreicht wird.

Ein weiteres Problem der lokalen Optimierung ist, dass zu einer gefundenen lokal-optimale Lösung  $\bar{x}^{\wedge}$  i.a. nicht angegeben werden kann, wie weit sie noch von einer global-optimale Lösung  $\bar{x}^*$  entfernt ist, oder ob sie nicht schon selbst eine global-optimale Lösung darstellt. Welche (lokal-)optimale Lösung letztendlich gefunden wird, hängt bei den meist rein determi-

nistisch arbeitenden Nachbarschaftsstrukturen entscheidend von deren Parametrisierung<sup>6</sup> sowie von der Wahl der Startlösung  $\vec{x}_{\text{start}}$  ab. Hier ist man häufig gezwungen, auf eine zufällig ausgewählte Startlösung zurückzugreifen, insbesondere dann, wenn keinerlei Vorwissen über die Oberflächencharakteristik der zu optimierenden Zielfunktion vorhanden ist.

Der Hauptvorteil des in Abb. 2.3.3.1.1 dargestellten Algorithmus besteht in seiner allgemeinen Anwendbarkeit, da nur der Lösungsraum  $L$ , eine berechenbare Zielfunktion  $F$  und eine Nachbarschaftsstruktur  $\eta$  vorausgesetzt werden. Die Definition der Nachbarschaftsstruktur hängt stark vom vorgegebenen Optimierungsproblem ab. Handelt es sich um ein Kombinatorikproblem, muss die Nachbarschaftsstruktur in den meisten Fällen individuell angepasst werden<sup>7</sup>. Auch bei Parameteroptimierungsproblemen gibt es durchaus unterschiedliche Realisierungsmöglichkeiten für eine Nachbarschaftsstruktur. Daraus resultiert auch die Vielzahl der heute existierenden lokalen Parameteroptimierungsverfahren, die unter dem Namen Hill-Climbing Strategien zusammengefasst werden. Die Vorgehensweise dieser Verfahren ist vergleichbar mit der eines blinden Bergsteigers, der sich von einem Tal aus zum nächstgelegenen Gipfel empor tastet. Die Tastschritte können entweder nach rein-deterministischen Regeln vorgenommen werden oder zufallsgesteuert ablaufen (stochastisches Hill-Climbing). In manchen Fällen wird auch eine Mischform verwendet. Jede dieser drei Verfahrensklassen zeigt eigene, charakteristische Konvergenzeigenschaften. Deterministische Verfahren konvergieren i.a. dann sehr schnell, wenn die Startlösung bereits in der Nähe der optimalen Lösung liegt. Stochastische Verfahren weisen dagegen eine weitaus sicherere Konvergenz bei beliebigen Startlösungen auf, konvergieren dafür aber deutlich langsamer.

Die wichtigsten direkten Hill-Climbing Verfahren, deren Suche sich ausschließlich an der Ergebnisoberfläche der Zielfunktion orientiert, sind

- Koordinatenstrategie [Sout40], [Sout46], [FrSa47]
- Strategie von Rosenbrock (Rotierende Koordinaten) [Rose60]
- Strategie von Hooke und Jeeves (Mustersuche) [HoJe61]
- Simplex-Strategie von Nelder und Mead [SpHH62]
- Strategie von Davies, Swann und Campey (DSC) [Swan64], [BoDS69]
- Complex-Strategie von Box [Box65]
- Simultaneous Perturbation Stochastic Approximation (SPSA) [Spal98], [Spall99]

Sämtliche der oben genannten Verfahren eignen sich zur Parameteroptimierung mehrdimensionaler Zielfunktionen. Handelt es sich bei  $F$  um eine mathematische Zielfunktion und ist diese stetig und differenzierbar, können auch die erste und zweite Ableitung von  $F$  mit in die Optimierungsstrategie eingebracht werden. Diese so genannten hybriden Strategien spalten sich wiederum auf in Gradienten- und Newton-Methoden. Bei den Gradienten-Methoden wird die Existenz der ersten Ableitung der Zielfunktion vorausgesetzt. Die Newton-Methoden benötigen

---

<sup>6</sup> bei Hill-Climbing Verfahren müssen beispielsweise Anfangsschrittweiten vorgegeben werden

<sup>7</sup> In [AaKo89] wird beispielsweise eine speziell auf das Travelling Salesman Problem TSP zugeschnittene Nachbarschaftsstruktur vorgestellt.

zusätzlich die zweite Ableitung. Für die Optimierung von Simulationsmodellen sind hybride Strategien, die analytische Zusatzinformationen über  $F$  verwenden, jedoch ohne Bedeutung, da hier die dazu notwendige mathematische Beschreibung von  $F$  nicht vorausgesetzt werden kann.

Eine umfassende Übersicht über das Themengebiet Hill-Climbing sowie detaillierte Beschreibungen der oben genannten Verfahren finden sich in [HoHo71], [Schw77] und [Prüf82].

### 2.3.3.2 Globale Optimierung

Die globale Optimierung hat zum Ziel, zu einem Optimierungsproblem  $(L, F)$  das globale Optimum  $F^*$  der Zielfunktion  $F$  zu ermitteln. Dabei kann es durchaus mehrere global-optimale Lösungsalternativen  $\bar{x}_1^*, \dots, \bar{x}_k^*$  geben, an denen die Zielfunktion den optimalen Wert  $F^*$  erreicht. Die Hauptprobleme, die sich bei der globalen Optimierung ergeben, sind:

- Zur globalen Optimierung existiert kein allgemein anwendbarer Basisalgorithmus wie etwa bei der lokalen Optimierung.
- Im Gegensatz zur lokalen Optimierung, bei der verschiedene einfache zu berechnende Kriterien für das Erreichen einer lokal-optimalen Lösung existieren (alle benachbarten Lösungen sind schlechter, erste partielle Ableitungen verschwinden), sind bei der globalen Optimierung Kriterien dieser Art nur in Trivialfällen<sup>8</sup> anwendbar. Die einzige Möglichkeit, um sicher nachzuweisen, dass es sich bei einer gefundenen Lösung um eine global-optimale Lösung handelt, ist i.a. das erschöpfende Durchsuchen des Lösungsraums<sup>9</sup>, wobei beachtet werden sollte, dass die Anzahl der Lösungsalternativen mit wachsender Problemdimension in der Regel exponentiell ansteigt. Diese Eigenschaft verursacht unter anderem auch die großen Schwierigkeiten bei der Definition geeigneter Abbruchkriterien für globale Optimierungsverfahren.
- In ihrer allgemeinen Form handelt es sich bei der globalen Optimierung aus heutiger Sicht um ein mathematisch nicht lösbares Problem [TöZi89]. Die existierenden Lösungsansätze basieren daher im Wesentlichen auf Problementschärfung (Relaxierung) und/oder dem Einsatz von Heuristiken.

Da nur in wenigen Fällen eine Problementschärfung möglich ist, stellen Heuristiken oft die einzige Möglichkeit dar, um mit akzeptablem Aufwand zu einer befriedigenden Lösung zu kommen, die in der Regel jedoch nicht die eigentlich gesuchte global-optimale Lösung darstellt. Die meisten der heute für die Praxis relevanten Ansätze zur globalen Optimierung beinhalten daher heuristische Elemente und es ist zu erwarten, dass dieser Trend sich in Zukunft noch weiter verstärkt. Eine allgemeine Einführung und einen umfassenden Überblick über Eigenschaften und Arten heuristischer Verfahren sowie deren Anwendungsbereiche wird in [Zimm87] gegeben.

In der Fachliteratur wird heute ein sehr breites Spektrum unterschiedlicher Lösungsansätze und Vorgehensweisen zur globalen Optimierung angeboten. Abhängig von Sichtweise und Anwen-

---

<sup>8</sup> beispielsweise bei unimodalen Problemstellungen

<sup>9</sup> was natürlich nur bei endlichen Lösungsräumen mit einem digitalen Rechner praktisch durchführbar ist

dungsgebiet ergibt sich eine Vielzahl an Möglichkeiten zur Klassifikation dieser Methoden (siehe dazu auch [TöZi89]). Bei dem in Abb. 2.3.3.2.1 dargestellten Klassifikationsvorschlag wird zunächst einmal zwischen Verfahren mit und ohne garantierten Erfolg unterschieden. Zu den Verfahren mit garantiertem Erfolg zählt zunächst einmal die vollständige Enumeration des Lösungsraums. Da bei vielen Problemstellungen die Anzahl an Lösungsalternativen mit wachsender Problemdimension exponentiell ansteigt, ist diese Methode jedoch nur bei geringer Problemdimension auch praktisch durchführbar. Hierzu ist noch anzumerken, dass das erschöpfende Durchsuchen nur bei endlichen Lösungsräumen das Auffinden des globalen Optimums auch tatsächlich "garantiert". Bei unendlich großen Lösungsräumen, beispielsweise bei Teilbereichen des  $\mathbb{R}^n$ , gilt dies nicht mehr. Hier kann mit einem Rechner aufgrund seiner begrenzten Genauigkeit immer nur eine endliche Anzahl von Punkten enumeriert werden, was unter Umständen lediglich zu einer Approximation des globalen Optimums führt.

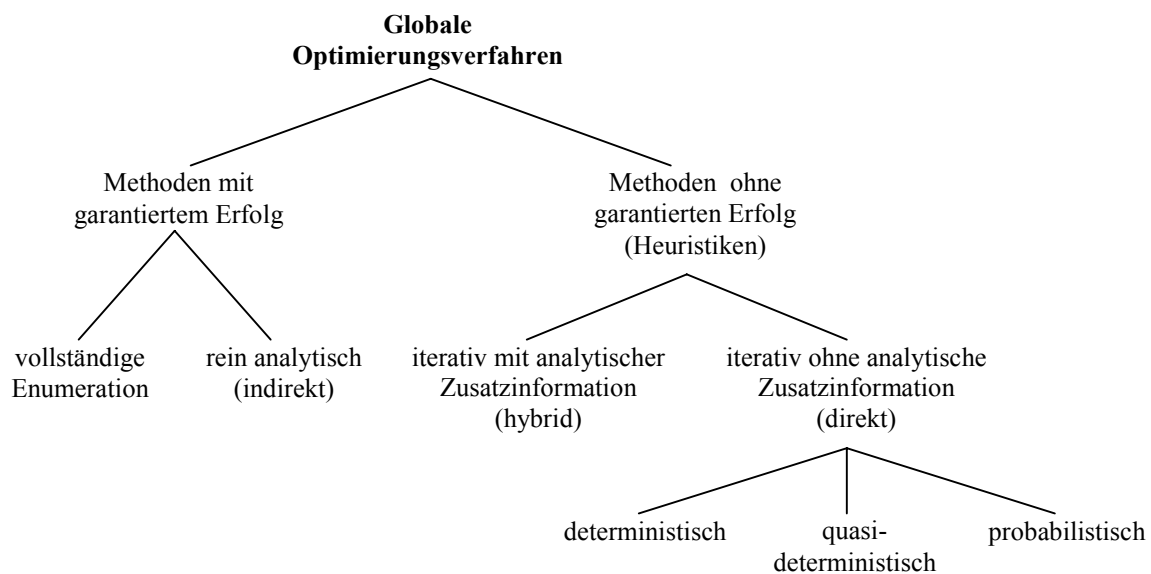


Abb. 2.3.3.2.1: Klassifikation globaler Optimierungsstrategien

Klassische mathematische Ansätze mit Erfolgsgarantie sind meist nur bei sehr eingeschränkten Problemstellungen anwendbar, beispielsweise dann, wenn die Zielfunktion und die den Lösungsraum einschränkende Nebenbedingungen linear von den Variablen abhängen. Speziell zur Lösung dieser Probleme wurde im Bereich des Operations-Research die Theorie der linearen Programmierung entwickelt. Die meisten niederdimensionalen Probleme dieser Art lassen sich heute problemlos auf Rechnern lösen. Mit wachsender Problemdimension wird die praktische Durchrechnung jedoch zunehmend schwieriger, da Speicherbedarf und Rechenzeit stark ansteigen. Eine Übersicht über die Theorie der linearen Programmierung sowie darüber hinausgehende Ansätze aus dem Operations-Research Bereich geben [BlOe75] und [BoGr93].

Bei den globalen Optimierungsverfahren ohne garantierten Erfolg handelt es sich um so genannte Heuristiken, die analog zu den im vorangegangenen Abschnitt vorgestellten lokalen Optimierungsverfahren i.a. schrittweise vorgehen. Hier kann weiter unterschieden werden zwischen hybriden Methoden, die analytische Zusatzinformationen verwenden, und direkten Methoden, die gänzlich ohne solche Zusatzinformationen auskommen. Da bei der Optimierung von Simulationsmodellen i.a. keine analytischen Zusatzinformationen vorausgesetzt werden

können, scheiden hybride Methoden für diesen Anwendungsbereich aus. Hier sind prinzipiell nur direkte Methoden anwendbar, wobei unterschieden werden kann zwischen rein-deterministischen, quasideterministischen und probabilistischen Methoden [Prüf82]. Rein deterministische Verfahren, die an keiner Stelle des Optimierungsprozesses von Zufallsprozessen Gebrauch machen, sind nur dann Erfolg versprechend, wenn man bereits über eine gute Anfangsannäherung an eine global-optimale Lösung verfügt. Da das i.a. nicht vorausgesetzt werden kann, greift man zur globalen Optimierung meist auf quasideterministische oder probabilistische Verfahren zurück. Bei den quasideterministischen Verfahren folgt auf einen Rechenschritt mit Zufallscharakter wenigstens ein rein-deterministischer Schritt. Probabilistische Verfahren weisen den höchsten Grad an Zufälligkeit auf. Hier herrscht ein deutliches Übergewicht an Rechenschritten mit Zufallscharakter.

Bei den zur globalen Optimierung von Simulationsmodellen relevanten Verfahren handelt es sich um direkte, auf probabilistischen Prinzipien basierende Methoden. Aus Effizienzgründen sollte der Zufall jedoch nicht willkürlich eingesetzt werden (reine Monte-Carlo Verfahren), sondern mit einer gewissen Systematik. Die heute bekanntesten globalen Optimierungsverfahren, deren Suche auf dem systematischen Einsatz probabilistischer Operatoren basiert, sind

- Genetische Algorithmen [Holl75], [Gold89], [Mich92]
- Evolutionsstrategien [Rech73], [Schw77], [Bäck96]
- Simulated Annealing [KiGV82], [KiGV83], [Kirk84], [Cern85], [AaKo89]

Sämtliche der oben aufgeführten Verfahren orientieren sich an Vorbildern aus der Natur und werden daher häufig auch als naturanaloge Verfahren bezeichnet. Aufgrund ihrer probabilistischen Suchoperatoren sind sie prinzipiell in der Lage, aus sub-optimalen Regionen des Lösungsraums zu entkommen und eignen sich aus diesem Grund sehr viel besser zur Exploration des Lösungsraums als etwa lokale Suchstrategien. Der große Nachteil dieser Verfahren ist jedoch die niedrige Konvergenzgeschwindigkeit, die sich vor allem dann bemerkbar macht, wenn das Verfahren sich bereits einer global-optimale Lösung angenähert hat. Dann steht der Aufwand an Zielfunktionsevaluierungen in keinem Verhältnis mehr zum Optimierungsschritt. Zur exakten globalen Optimierung aufwändig zu berechnender Simulationsmodelle sind diese Verfahren aufgrund ihrer Ineffizienz und Ungenauigkeit daher nicht zu empfehlen.

## 2.4 Web-Technologien

In diesem Abschnitt werden die für Kapitel 3 relevanten Grundlagen eingeführt, das sich mit den vielfältigen Anwendungsmöglichkeiten von Web-Technologien in der computergestützten Modellierung und Simulation auseinandersetzt. Bei den dort eingesetzten Web-Technologien handelt es sich im Wesentlichen um Auszeichnungssprachen, das Common Gateway Interface, die Programmiersprache Java und Web3D-Technologien. Nach einer kurzen Einführung in das Internet werden diese Technologien in ihren Grundzügen vorgestellt. Einen umfassenden Überblick über das Internet und dessen Kerntechnologien geben [Abra98], [Blac99], [CoKr99] und [Wild99].

## 2.4.1 Grundlegendes zum Internet

Das was heute als "Internet" bezeichnet wird, wurde Ende der 60er Jahre vom US-amerikanischen Verteidigungsministerium ins Leben gerufen. Zu einer weiten Verbreitung der auf dem Prinzip der Paketvermittlung basierenden Internet-Protokolle kam es gegen Ende der 70er Jahre vor allem an Universitäten und Forschungseinrichtungen. Anfang der 90er Jahre ermöglichte die Erweiterung der Internet-Dienste durch das so genannte World Wide Web (abgekürzt WWW oder Web) schließlich einer breiten Öffentlichkeit den Zugang zum "Netz". Heute kann das Internet durchaus als ein Massenphänomen bezeichnet werden, an das weltweit Millionen von Teilnehmern angeschlossen sind.

Betrachtet man die technischen Aspekte, auf denen das Internet basiert, kann man folgende Bereiche unterscheiden:

- Infrastruktur- und Protokollfragen
- Internet-Dienste und -Anwendungen sowie die dazugehörigen Programme

Das Herz der Internet-Kommunikationstechnologie bildet die Protokollfamilie TCP/IP, die nach ihren Hauptbestandteilen TCP (Transmission Control Protocol) und IP (Internet Protocol) benannt ist. TCP/IP stellt einen allgemein anerkannten Standard dar, der ein funktional gegliedertes Schichtenmodell sowie darin eingeordnete Kommunikationsprotokolle umfasst. In Abb. 2.4.1.1 ist das Schichtenmodell der Internet-Protokollfamilie dargestellt. Es setzt sich aus insgesamt vier Schichten zusammen, die im Folgenden näher beschrieben werden:

- Der Datenübertragungsschicht ordnet man i.a. die Gerätetreiber sowie die korrespondierende Hardware zu, bei der es sich um eine Schnittstelle handelt, die ein bestimmtes Übertragungsmedium und -verfahren, wie z.B. Ethernet, Token Ring oder ISDN unterstützt. Damit beschreibt die Datenübertragungsschicht die möglichen Teilnetze des Internet und deren Zugangsprotokolle.
- Die Netzwerkschicht vereinigt alle Teilnetze zum Internet. Das IP, welches das Hauptprotokoll der Netzwerkschicht darstellt, realisiert eine paketorientierte Datenübertragung sowie die damit verbundene Wegwahl für Datenpakete vom Sender zum Empfänger über einzelne Teilnetze (Routing).
- Die Protokolle der Transportschicht setzen auf den von der Netzwerkschicht angebotenen Diensten auf. Das TCP stellt den Anwendungen als verbindungsorientiertes Protokoll eine byteorientierte, vollduplexfähige Datenübertragung zur Verfügung. Das ebenfalls in der Transportschicht angesiedelte UDP (User Datagram Protocol) ordnet die Dateneinheiten den jeweiligen Anwendungen zu, wobei es als verbindungsloses Protokoll keine Garantie für die korrekte Übertragung der Dateneinheiten übernimmt.
- Die Anwendungsschicht stellt eine Reihe von Anwendungsprotokollen zur Verfügung, von denen einige in Abb. 2.4.1.1 aufgeführt sind. Telnet, HTTP (Hypertext Transfer Protocol) und FTP (File Transfer Protocol) nutzen TCP als unterliegendes Transportprotokoll, während TFTP (Trivial File Transfer Protocol) als einfachere Variante des Dateiaustausches meist das verbindungslose Transportschichtprotokoll UDP verwendet.



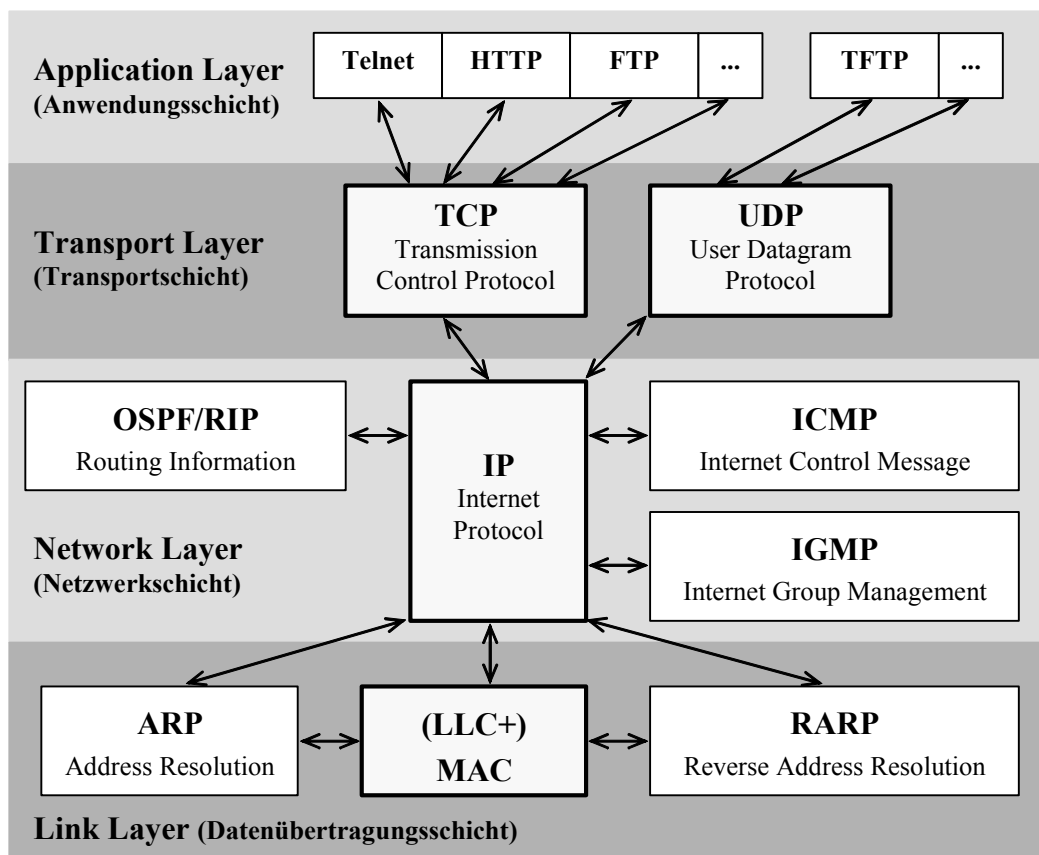


Abb. 2.4.1.1: Die TCP/IP-Protokollfamilie

Heute werden auf Basis des oben beschriebenen TCP/IP-Standards im Internet verschiedene Dienste und Anwendungen angeboten. Dabei kann es sich um standardisierte Anwendungen handeln, die eines der in Abb. 2.4.1.1 dargestellten Protokolle der Anwendungsschicht verwenden. Daneben sind aber auch individuell entwickelte Anwendungen möglich, die entweder direkt auf der Transportschicht aufsetzen oder standardisierte Protokolle wie beispielsweise HTTP entsprechend erweitern. Zu den wichtigsten standardisierten Internetanwendungen zählen heute:

- Telnet: Terminal-Emulation für die Benutzung externer Rechner auf dem Netz
- E-Mail: Elektronische Post und Datenaustausch auf Textbasis
- Mailing Lists: Diskussionsgruppen auf E-Mail-Basis
- USNET Newsgroups: Diskussionsgruppen mit netzweiter Verteilung der Nachrichten
- FTP: Dateitransfer und -suche
- Gopher: Passiver Informationsdienst und passive Informationssuche
- Internet Relay Chat: Tastaturbasierte Realzeit-Kommunikation
- World Wide Web: Aktiver Informationsdienst

Im Folgenden wird näher auf das World Wide Web und dessen Kerntechnologien eingegangen, die neben vielen anderen Bereichen auch die Modellierung und Simulation wesentlich beeinflusst haben.

## 2.4.2 Das World Wide Web

Die mit Abstand populärste Anwendung im Internet stellt heute das World Wide Web dar, das Anfang der 90er Jahre aus einer Initiative von Wissenschaftlern hervorging, um das bereits bestehende Internet für einen neuartigen Hypertext-basierten Informationsaustausch zu nutzen. Um dies zu erreichen, wurde die Dokumentbeschreibungssprache HTML (Hypertext Markup Language) und mit HTTP (Hypertext Transfer Protocol) ein neues Internet-Anwendungsprotokoll entwickelt, das die Übertragung von HTML-Dokumenten ermöglichen sollte. Gleichzeitig dazu entstanden so genannte Web-Browser, um die auf Web-Servern bereitgestellten Dokumente anzeigen zu können, sowie das Konzept der URL (Uniform Resource Locator), das jedem dieser Dokumente eine eigene unverwechselbare Adresse zuordnet. Die lawinenartige Ausbreitung des WWW setzte Anfang 1993 ein, als weltweit zirka 50 Web-Server existierten. In engem Zusammenhang mit dieser Entwicklung steht das Aufkommen und die Verbreitung weiterer wichtiger Technologien wie beispielsweise Java, CGI (Common Gateway Interface), XML (Extensible Markup Language) und VRML (Virtual Reality Modeling Language). Diese Technologien, die heute auch verstärkt im Bereich der Modellierung und Simulation Anwendung finden, werden in den nun folgenden Abschnitten näher vorgestellt.

## 2.4.3 Auszeichnungssprachen

Grundlegendes Prinzip einer Auszeichnungssprache ist das Markieren (Auszeichnen) bestimmter Elemente eines Dokuments durch spezielle Befehle. Durch diese so genannten Auszeichnungsbefehle (Markups) werden Zusatzinformationen zur inhaltlichen und optischen Strukturierung in das eigentliche Dokument hineinkodiert. Da Auszeichnungsbefehle besonderer Verarbeitung bedürfen, müssen sie durch spezielle vorher definierte Zeichen (meistens nutzt man hierzu die Größer- und Kleiner-Zeichen) eingegrenzt werden. Man unterscheidet grundsätzlich zwischen logischer (oder semantischer) und physischer Auszeichnung. Bei den logischen Auszeichnungen handelt es sich um inhaltliche Definitionen des gekennzeichneten Textes. Beispielsweise kann festgelegt werden, dass es sich bei dem ausgezeichneten Text um einen Namen, eine Adresse oder eine Überschrift handelt. Physische Auszeichnungen verfolgen dagegen das Ziel, Möglichkeiten zur visuellen Textdarstellung zu eröffnen. Mit solchen Formatierungsbefehlen kann beispielsweise angezeigt werden, dass eine bestimmte Textstelle normal, kursiv oder fett dargestellt werden soll. Die meisten Auszeichnungssprachen verwenden beide Arten der Auszeichnung, wobei die eine oder die andere Auszeichnungsart je nach verwendeter Sprache stärker oder schwächer ausgeprägt sein kann.

Zu den bekanntesten Auszeichnungssprachen zählen SGML (Standard Generalised Markup Language), HTML (Hypertext Markup Language) und XML (Extensible Markup Language), die im Folgenden näher erläutert werden.

### 2.4.3.1 Standard Generalised Markup Language

1969 wurde in den IBM Labors die Generalised Markup Language (GML) entwickelt. GML stellt kein simples Auszeichnungssystem dar, sondern eine generelle Grammatik, die es gestattet, konkrete Auszeichnungssprachen mit Hilfe von Document Type Definitions (DTD) formal zu definieren. GML kann also als eine Sprache zur Beschreibung von Sprachen aufgefasst werden und wird daher auch als Metasprache bezeichnet. 1978 wurde ein ANSI-Komitee (American National Standards Institute) zur Entwicklung eines Standards für Auszeichnungssprachen auf Basis von GML gegründet. Das Ergebnis war SGML die Standardized Generalised Markup Language, die 1986 zum ISO-Standard (ISO 8879) wurde. Jedes dem SGML-Standard entsprechende Dokument gehört einem bestimmten Typ an. Ein Dokumenttyp hat eine eindeutige hierarchische Struktur und einen festgelegten Satz an zur Verfügung stehenden Markups, die oft auch als "Tags" bezeichnet werden. Die Deklaration von Struktur und Tags wird in der DTD vorgenommen. Mit der DTD wird also Syntax und Semantik der Auszeichnungssprache für diesen Dokumenttyp festgelegt, wodurch die Notwendigkeit entfällt, für jede neue Auszeichnungssprache einen neuen Interpreter schreiben zu müssen. Als Co-Standard zu SGML wurde DSSSL (Document Style Semantics and Specification Language) entwickelt, der als eine entkoppelte Präsentationsebene verstanden werden kann, welche die Festlegung der Dokumentenvisualisierung ermöglicht. DSSSL stellt eine algorithmisch vollständige Skriptsprache dar, anhand derer für jeden SGML-Elementtyp eine Regel festgelegt werden kann, die dessen Visualisierung steuert.

SGML enthält Sprachmittel für die unterschiedlichsten Zwecke und ist damit eine äußerst flexible Architektur, mit der Dokumente für beliebige Medien aufbereitet werden können, ohne die Struktur der Daten zu verlieren. So ist HTML beispielsweise eine Anwendung von SGML. Problematisch ist die Komplexität von SGML, welche die Entwicklung von SGML-Anwendungen teuer und kompliziert macht und bisher einer weiten Verbreitung von SGML entgegenstand. Während HTML also aufgrund der fehlenden Erweiterbarkeit für komplexere Anwendungen ungeeignet ist, erweist sich SGML wegen seiner sehr hohen Komplexität als im Internet nur begrenzt einsetzbar. Um eine für das Internet geeignete Metasprache zu etablieren, wurde im Februar 1998 vom World Wide Web Consortium<sup>10</sup> (W3C) die Extensible Markup Language (XML) als Standard verabschiedet, die eine auf das Internet zugeschnittene Teilmenge von SGML darstellt. Die mit dem World Wide Web in engem Zusammenhang stehenden Auszeichnungssprachen HTML und XML werden in den beiden folgenden Abschnitten näher vorgestellt.

### 2.4.3.2 Hypertext Markup Language

Die wohl bekannteste Anwendung von SGML ist die speziell für das WWW entwickelte Auszeichnungssprache HTML, die seit Version 2.0 durch eine DTD definiert ist. Neben der Auszeichnung von Textelementen kann man mit HTML auch Verweise (Hyperlinks) im Dokument definieren. Diese Verweise können dabei sowohl auf lokale Dateien zeigen als auch auf Da-

---

<sup>10</sup> Das W3C wurde im Oktober 1994 am MIT Laboratory for Computer Science (MIT/LCS) in Boston (USA) gegründet und befasst sich hauptsächlich mit der Entwicklung und Standardisierung neuer Protokollspezifikationen und Architekturen für das World Wide Web. Weitere Informationen dazu finden sich im WWW unter [W3C1].

teilen, die auf einem beliebigen Server im Internet zur Verfügung stehen. Da zur Definition dieser Verweise der Uniform Resource Locator (URL) zur Anwendung kommt, muss sich der Anwender weder um den Namen des Rechners, noch das Übertragungsprotokoll (beispielsweise HTTP oder FTP) und die Position der Datei im Dateisystem des Servers kümmern.

HTML ist primär auf die einheitliche Präsentation von Informationen innerhalb von Web-Browsern ausgerichtet. Zur semantischen Auszeichnung von Daten ist HTML jedoch nicht qualifiziert, da die Möglichkeit fehlt, neue Tags zu definieren. Komplexe Strukturen wie das Relationenschema einer Datenbank oder Objekthierarchien sind daher in HTML nicht abbildbar. Sollen über die reine Dokumentenvisualisierung hinaus auch Inhalte (logische Strukturen) in einheitlicher Form repräsentiert werden, muss auf Metasprachen wie SGML oder XML zurückgegriffen werden.

### **2.4.3.3 Extensible Markup Language**

Um über die reine Dokumentenvisualisierung hinaus auch Inhalte (logische Strukturen) in einheitlicher Form repräsentieren zu können, wurde mit XML ein weiterer offener Standard entwickelt. XML ist keine SGML-Anwendung wie beispielsweise HTML, sondern ein SGML-Profil, das es ermöglicht, eigene neue Auszeichnungssprachen für die verschiedensten Arten von Dokumenten zu definieren. Damit lassen sich Daten bzw. Dokumente derart beschreiben und strukturieren, dass sie zwischen einer Vielzahl von Anwendungen ausgetauscht und weiterverarbeitet werden können. Wie SGML ermöglicht auch XML beliebig viele neue Tags und Attribute individuellen Zielen entsprechend zu definieren. Um eine hohe Akzeptanz zu erreichen, wurde XML von seinen Entwicklern jedoch sehr viel einfacher als SGML gehalten. Zur Reduzierung der Komplexität wurden alle für das Internet als überflüssig angesehenen SGML-Eigenschaften sowie eine Vielzahl als zu kompliziert erachteter und zu selten genutzter SGML-Bestandteile nicht in XML übernommen. Trotz der Reduzierungen ist XML aufwärtskompatibel zu SGML und wird daher gelegentlich auch "SGML-Lite" genannt.

Das Grundkonzept von XML besteht in der strikten Trennung von Inhalt, Struktur und Layout. Die Darstellung eines XML-Dokuments erfolgt mit Hilfe einer separaten Formatvorlage, die auch als Style Sheet bezeichnet wird. Durch diese Style Sheets, mit denen das Layout des Dokuments festgelegt wird, können XML-Dokumente beliebig präsentiert werden, ohne dabei auf geräte- oder anwendungsunabhängige Weiterverarbeitbarkeit verzichten zu müssen. So kann die Verwendung unterschiedlicher Style Sheets zur Anpassung eines Dokumentes für verschiedene Zwecke wie beispielsweise Ausdruck und Bildschirmanzeige genutzt werden. Das W3C entwickelt derzeit mit der Extensible Style Sheet Language (XSL) eine eigene Style Sheet Sprache für XML. XSL ist von DSSSL abgeleitet, das seinen Ursprung in der SGML-Entwicklung hat.

XML bietet wesentlich weitergehende Möglichkeiten des Verweizens als HTML. Das W3C entwickelt momentan ein Verweismodell für XML, das aus der XML Linking Language (XLink) und der XML Pointer Language (XPointer) besteht. Das Modell soll neben den aus HTML bekannten unidirektionalen auch multidirektionale Verweise, die auf mehrere Ressourcen gleichzeitig verweisen sowie bidirektionale Verweise, die in beide Richtungen referenzieren, unterstützen. Darüber hinaus wird ein mächtiges Modell für das Verweisen innerhalb von XML-Dokumenten bereitgestellt.

Obwohl XML mit Hilfe von Style Sheets zur Repräsentation und Darstellung von Web-Seiten genutzt werden kann, wird es HTML zukünftig wohl nicht als dessen Nachfolger verdrängen können. HTML und XML sollten eher als komplementäre Technologien betrachtet werden, die sich ergänzen, anstatt zu konkurrieren. XML wird sich überall dort durchsetzen, wo HTML Schwächen aufweist. Diese liegen nicht in den Darstellungsmöglichkeiten, sondern darin, Informationen so aufzubereiten und zu strukturieren, dass sie leicht weiterverarbeitet werden können.

## 2.4.4 Common Gateway Interface

Statische Web-Seiten, bei denen der Seiteninhalt nur durch äußeren Eingriff des Autors geändert werden kann, stellen die einfachste und ursprüngliche Form der Informationsbereitstellung im WWW dar. Für viele Anwendungen sind Seiten dieser Art jedoch zu unflexibel, insbesondere dann, wenn die angebotenen Informationen ständig aktualisiert werden müssen (Wettervorhersage, Börsenkurse, etc.). In solchen Fällen benötigt man Seiten, deren Inhalte durch eine auf dem Web-Server implementierte Anwendungslogik zur Laufzeit erzeugt werden. Zur Erstellung derartiger dynamischer Seiten existieren heute verschiedene Techniken, die unter dem Begriff Server-Side-Scripting zusammengefasst werden. Der Grund für diese Bezeichnung liegt darin, dass viele dieser Techniken auf dem Einsatz von Skriptsprachen wie beispielsweise Perl, Tcl (Tool Command Language) oder der Unix-Shell basieren. Ein Skript zeichnet sich gegenüber übersetzten Programmen dadurch aus, dass es durch ein anderes Programm ausgeführt oder direkt interpretiert wird.

Der vom National Center for Supercomputer Applications (NCSA) entwickelte CGI-Standard (Common Gateway Interface) ist seit Jahren die Schlüsseltechnologie zur dynamischen und interaktiven Erzeugung von Web-Seiten. Er stellt eine Programmierschnittstelle dar, um externe Programme unter Steuerung eines Informationsservers (bei Web-Anwendungen ein HTTP-Server) ablaufen zu lassen. Traditionell werden CGI-Programme in Skriptsprachen geschrieben. Es können aber durchaus auch kompilierte Sprachen wie Fortran, C oder C++ verwendet werden. In vielen Fällen werden CGI-Programme über HTML-Formulare aufgerufen. Dort füllt man ein oder mehrere Formularfelder aus, schickt das ausgefüllte Formular durch Drücken auf eine Schaltfläche ab und bekommt nach einigen Sekunden eine Antwort in Form einer HTML-Seite. Dabei passiert im Wesentlichen folgendes:

1. Der Browser kodiert die Eingaben in ein übertragungsfreundliches Format.
2. Diese so genannten CGI-Parameter werden entweder als eine Art erweiterte URL an den Server weitergegeben (GET-Methode), oder über eine separate logische Verbindung geschickt (POST-Methode).
3. Der Server ruft das CGI-Programm auf und übergibt ihm die Parameter (bei GET als Umgebungsvariable, bei POST über die Standardeingabe).
4. Das CGI-Programm dekodiert die Parameter, führt bestimmte Aktionen aus (beispielsweise eine Datenbankabfrage) und gibt über die Standardausgabe eine Antwort, meist eine HTML-Seite.
5. Der Anwender sieht die Antwort in seinem Browser.

In Abb. 2.4.4.1 sind die oben beschriebenen Abläufe noch einmal zusammengefasst. Zur Nutzung der CGI-Technologie wird ein Web-Server mit aktivierter CGI-Schnittstelle benötigt. Diese stellt ein Verzeichnis (meist cgi-bin) zur Verfügung, in dem die einzelnen CGI-Programme abgelegt sind.

CGI-Programme stellen heute eine bereits veraltete Technologie dar, die einige entscheidende Nachteile aufweist. So müssen beispielsweise CGI-Prozesse bei jeder Anfrage neu gestartet werden, was eine speicher- und zeitaufwändige Ausführung zur Folge hat und die Leistung der Web-Server erheblich beeinträchtigt. CGI-Programme werden daher zunehmend durch modernere auf Java basierende Technologien wie beispielsweise Java-Servlets verdrängt.

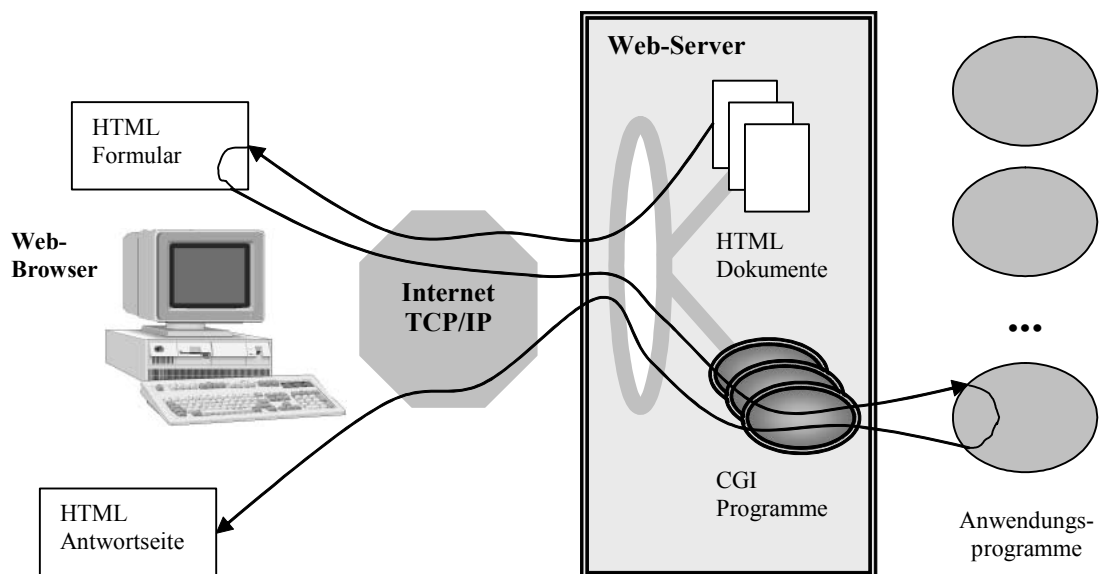


Abb. 2.4.4.1: Das CGI-Kommunikationsmodell

## 2.4.5 Java

Die weite Verbreitung und die große Popularität, die Java heute besitzt, stehen in engem Zusammenhang mit der explosionsartigen Ausbreitung des World Wide Web. In der ursprünglichen Planung von Java, die auf das Jahr 1991 zurückgeht, spielte das Internet jedoch noch keine Rolle. Damals bildete die Firma Sun Microsystems unter der Führung von James Gosling und Billy Joy eine Gruppe unter dem Namen "Green Project", deren Aufgabe es war, Entwicklungsarbeiten für den Zukunftsmarkt der eingebetteten Systeme zu leisten. Als das bedeutendste Ergebnis dieses Forschungsprojekts stellte sich Java heraus, das im Mai 1995 erstmals im Internet zum kostenlosen Herunterladen angeboten wurde. Seitdem etablierte sich Java in atemberaubender Geschwindigkeit sowohl als Programmiersprache als auch als Plattform für die Entwicklung heterogener Netzwerk-zentrierter Systeme. Bis heute wurde Java millionenfach installiert und fast jede Firma aus dem Computer- und Telekommunikationsbereich hat die Java-Technologie von Sun Microsystems lizenziert.

Interessant an Java ist vor allem, dass es weit mehr als nur eine neue Programmiersprache darstellt. Mit Java werden außerdem ein Laufzeitsystem, eine Menge von Entwicklungswerkzeugen und ein sehr mächtiges API (Application Programming Interface) assoziiert. Die Beziehungen zwischen all diesen Elementen sind in Abb. 2.4.5.1 dargestellt.

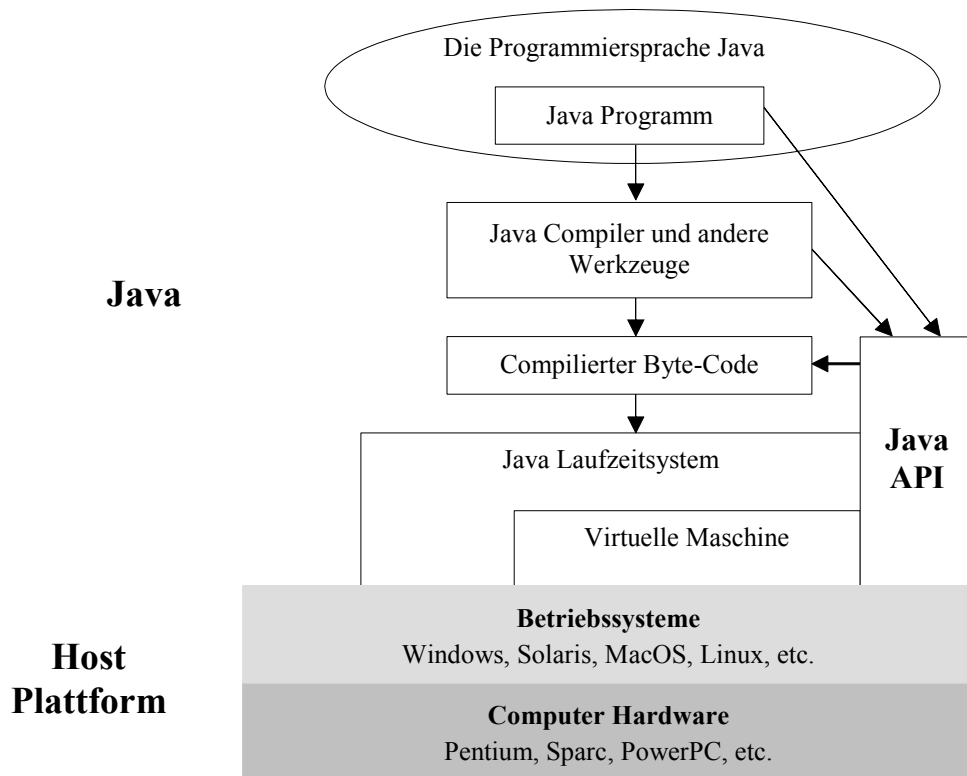


Abb. 2.4.5.1: Die Hauptbestandteile von Java

Java hat in kürzester Zeit enorme Popularität erlangt und ist heute dabei, C++ als führender Programmiersprache der 90er Jahre den Rang abzulaufen. Ursachen dieser phänomenalen Entwicklung sind einige wichtige Eigenschaften, die Java als ideale Programmiersprache für eine zunehmend stärker vernetzte, heterogene Rechnerwelt und insbesondere für das World Wide Web erscheinen lassen:

- Java ist vertraut und einfach

Java orientiert sich stark an C und C++, ist demgegenüber jedoch radikal vereinfacht und verbessert worden. Für einen Großteil der Programmierer dürfte Java also bereits vertraut erscheinen. Aufgrund der überschaubaren Anzahl von Sprachkonstrukten lässt sich Java aber auch sehr schnell erlernen.

- Java ist objektorientiert

Trotz der vorgenommenen Vereinfachungen gegenüber C++ ist Java eine vollständige und sehr elegante objektorientierte Sprache. Im Gegensatz zu C++ wurde Java von Grund auf

objektorientiert konzipiert. Die Klasse bildet die Basiseinheit der Kompilierung und Ausführung in Java. Alle Java-Programme sind Klassen.

- Java ist robust

Durch das Sprachdesign von Java werden bestimmte Arten von Programmierfehlern von vornherein ausgeschlossen. Beispielsweise wird der direkte Zugriff auf den Speicher vollständig unterbunden, was eine große und sehr unangenehme Fehlerklasse völlig ausschließt. Das "Exception Handling" stellt eine weitere Maßnahme zur Entwicklung von sehr zuverlässiger Software dar.

- Java betont den Sicherheitsaspekt

Aufgrund der verteilten Natur von Java wurde beim Sprachentwurf insbesondere auf Sicherheitsbelange geachtet. Das Java Laufzeitsystem überprüft den zu interpretierenden Byte-Code und verhindert unerlaubten Zugriff auf lokale Ressourcen.

- Java unterstützt Mehrfädigkeit

Durch Mehrfädigkeit lassen sich verschiedene Aufgaben auf einem Einprozessorsystem scheinbar gleichzeitig erledigen. Zur mehrfädigen Programmierung stellt Java die Klasse "Thread" zur Verfügung, welche Methoden enthält, mit denen Programmfäden<sup>11</sup> gestartet und gestoppt sowie die Prioritäten von Programmfäden eingestellt werden können.

- Java ist plattformunabhängig und portabel

Java-Programme werden in ein architekturneutrales Byte-Code Format übersetzt. Daher können Java-Anwendungen auf jedem System ausgeführt werden, auf dem das Java Laufzeitsystem implementiert ist. Diese Eigenschaft ist vor allem für Anwendungen wichtig, die über das Internet verteilt werden.

- Java verfügt über eine mächtige Sammlung von Klassenbibliotheken

Seit der Einführung von Java wurden die Klassenbibliotheken ständig erweitert. Sie lassen sich heute in drei große Kategorien unterteilen: 1.) Kernklassen, die in jedem JDK (Java Development Kit) enthalten sein müssen; 2.) Standard Erweiterungen zum JDK; 3.) Klassenbibliotheken für spezielle Hardware oder hoch spezialisierte Anwendungsbereiche.

- Java ist in Web-Browsern ausführbar

Java-Applets ermöglichen es, Java-Anwendungen in Web-Browsern auszuführen. Damit können Java-Programme von jedermann überall in der Welt, zu jeder Zeit, auf jeder Plattform und ohne jeden Installationsaufwand ausgeführt werden. Das Applet-Konzept trug entscheidend zur enormen Popularität von Java bei und erklärt, warum Java heute auch als die Programmiersprache des Internet bezeichnet wird.

Um den unterschiedlichen Anforderungen möglichst vieler Anwendungsbereiche gerecht zu werden, wurde Ende 1998 die mittlerweile sehr umfangreich gewordene Java-Technologie von

---

<sup>11</sup> häufig spricht man in diesem Zusammenhang auch von Leichtgewichtsprozessen



Sun Microsystems neu strukturiert. Als Ergebnis wurde im Juni 1999 auf der JavaOne Konferenz in San Francisco die Java 2 Plattform vorgestellt, die eine Unterteilung der Java-Technologie in drei auf unterschiedliche Anwendungsbereiche zugeschnittene Editionen vorsieht. Dabei handelt es sich im Einzelnen um die

- Java 2 Standard Edition (J2SE), die den normalen Sprachumfang für die Entwicklung von Client-seitigen Applikationen mit grundlegender Netzwerkunterstützung enthält,
- Java 2 Enterprise Edition (J2EE), die eine integrierte Plattform für die Server-seitige Entwicklung portabler Software-Komponenten (Enterprise JavaBeans) darstellt,
- Java 2 Micro Edition (J2ME), die speziell auf die Restriktionen von eingebetteten Systemen zugeschnitten ist.

Die virtuellen Maschinen sowie die Programmierschnittstellen der oben aufgeführten Java-Editionen sind über die Web-Seiten von Sun Microsystems verfügbar [JAVA1] und können für verschiedene Betriebssysteme (Solaris, Windows, MacOS) in komprimierter Form heruntergeladen werden [JAVA2], [JAVA3].

Neben zahlreichen Vorteilen weist Java jedoch auch einige Nachteile auf. Der wohl bedeutendste Nachteil besteht darin, dass Java als interpretierte Sprache kompilierten Programmiersprachen hinsichtlich der Ausführungsgeschwindigkeit grundsätzlich unterlegen ist. Durch Just-in-Time Compiler und der Java HotSpot Technologie ist es jedoch mittlerweile gelungen, sehr nahe an die Ausführungsgeschwindigkeit des größten Konkurrenten C++ heranzukommen. Darüber hinaus wird durch die Entwicklung spezieller Hardware (Java Prozessoren) versucht, eine weitere Beschleunigung von Java zu erreichen.

## 2.4.6 Web3D-Technologien

Der Begriff "Web3D" bezeichnet alle offenen, aber auch firmeneigene Technologien, die eine interaktive Darstellung von 3D-Grafiken im Web erlauben. Der Begriff Technologie wird hier deshalb verwendet, weil zur Repräsentation von Web3D-Szenen neben einem Format auch entsprechende Technologien wie beispielsweise Client/Server-Architekturen, Autorenwerkzeuge sowie Skalierungs-, Kompressions- und Streamingmechanismen gehören. Eine Web3D-Szene bezeichnet einen dreidimensionalen Raum, in dem 3D-Objekte angeordnet sind. Diese Objekte können durch Oberflächen- und Volumenmodelle sowie Freiformobjekte repräsentiert werden.

Zu den bedeutendsten Web3D-Technologien zählen heute Java3D und VRML (Virtual Reality Modeling Language). Bei Java3D handelt es sich um eine Erweiterung der weit verbreiteten Programmiersprache Java. VRML dagegen stellt eine offene, plattformunabhängige Beschreibungssprache dar, die deutlich einfacher zu handhaben ist als Java3D, da keine Programmierkenntnisse notwendig sind und man sich nicht mit der Programmierung der Darstellung beschäftigen muss. Parallel zu VRML und Java3D sind weitere Konzepte und Technologien für den Aufbau und die Verbreitung dreidimensionaler Darstellungen im World Wide Web entstanden, deren Anwendungsgebiete sich zum Teil erheblich unterscheiden. Das Web3D-Konsortium bemüht sich derzeit um eine Integration der besten Elemente dieser Technologien unter dem Namen X3D [X3D1]. Eine ausführliche Beschreibung und Bewertung der heute existie-

renden Formate, Technologien und Architekturkonzepte für Web3D-Applikationen findet sich in [Rukz01].

Abschließend sei noch auf einige Problemfelder beim Umgang mit den heute zur Verfügung stehenden Web3D-Technologien hingewiesen. Diese setzen zum einen eine hohe Bandbreite zur Übertragung der i.a. sehr großen Datenmenge einer Web3D-Szene zwischen Client und Server voraus. Zum anderen werden leistungsstarke Web3D-Plattformen benötigt, um das Visualisieren von und Interagieren mit einer Web3D-Szene zu ermöglichen. Darüber hinaus mangelt es heute noch an spezialisierten Autorenwerkzeugen zur Erstellung von Web3D-Applikationen.

## 2.5 Komponenten-Technologien

In diesem Abschnitt werden die für Kapitel 4 relevanten Grundlagen eingeführt, das sich mit dem Einsatz von Komponenten-Technologien zum strukturierten Aufbau von Simulationssoftware auseinandersetzt. Simulationen stellen i.a. hochkomplexe Softwaresysteme dar, die wie Anwendungen aus anderen Bereichen auch vielen teilweise sehr schwer zu erfüllenden Anforderungen gerecht werden müssen. Der Endanwender beispielsweise legt großen Wert auf einfach zu bedienende und leicht verständliche Programme, die ohne hohen Installationsaufwand ausführbar und nach Möglichkeit zeit- und ortsunabhängig über das World Wide Web verfügbar sein sollten. Entwickler dagegen verlangen leistungsfähige Unterstützungswerkzeuge, die insbesondere die flexible Vernetzung der entstehenden Applikationen untereinander, aber auch deren Anbindung an die heute existierenden heterogenen Altsysteme ermöglichen. Unternehmen stellen schließlich hohe Anforderungen an die Zuverlässigkeit sowie an die Wartungs- und Erweiterungsmöglichkeiten ihrer Anwendungssysteme. Um all diesen Anforderungen gerecht zu werden, ist eine ingenieurmäßige Vorgehensweise bei der Entwicklung von Software unabdingbar. In diesem Zusammenhang wird heute in zunehmendem Maße das Schlagwort "komponentenorientierter Softwareentwurf" genannt, worunter eine Weiterentwicklung der objektorientierten Techniken verstanden werden kann, wobei der verteilte Charakter von Komponentensystemen hervorgehoben und als Abgrenzung zu klassischen objektorientierten Systemen gesehen wird [Grif98]. Software aus Komponenten, kurz "Componentware", gilt als viel versprechende Technologie, um Interoperabilität von Software zu gewährleisten sowie die Hindernisse bei deren Wiederverwendung zu überwinden.

Um einen Überblick über die verschiedenen Aspekte des Komponentenbegriffs zu geben, werden zunächst die wichtigsten der heute existierenden Konzepte zum strukturierten Softwareentwurf einander gegenübergestellt. Im Anschluss daran werden Methoden und Techniken der komponentenorientierten Softwareentwicklung erläutert. Abschließend wird im Detail auf CORBA (Common Object Request Broker Architecture) eingegangen, das einen weit verbreiteten und im Rahmen dieser Arbeit häufig verwendeten Standard für verteilte Objektsysteme darstellt.

## 2.5.1 Konzepte zum strukturierten Softwareentwurf

In Abb. 2.5.1.1 sind die wichtigsten der heute existierenden Konzepte zum strukturierten Softwareentwurf aufgeführt und hinsichtlich der Kriterien Eigenständigkeit und Abstraktionsniveau bewertet. Eigenständigkeit ist dabei ein Maß für den Grad der Unabhängigkeit von den umgebenden Strukturen innerhalb einer Gesamtanwendung. Das Abstraktionsniveau bezeichnet die Nähe zum Anwendungskontext bzw. die Entfernung zur programmiertechnischen Systemebene.

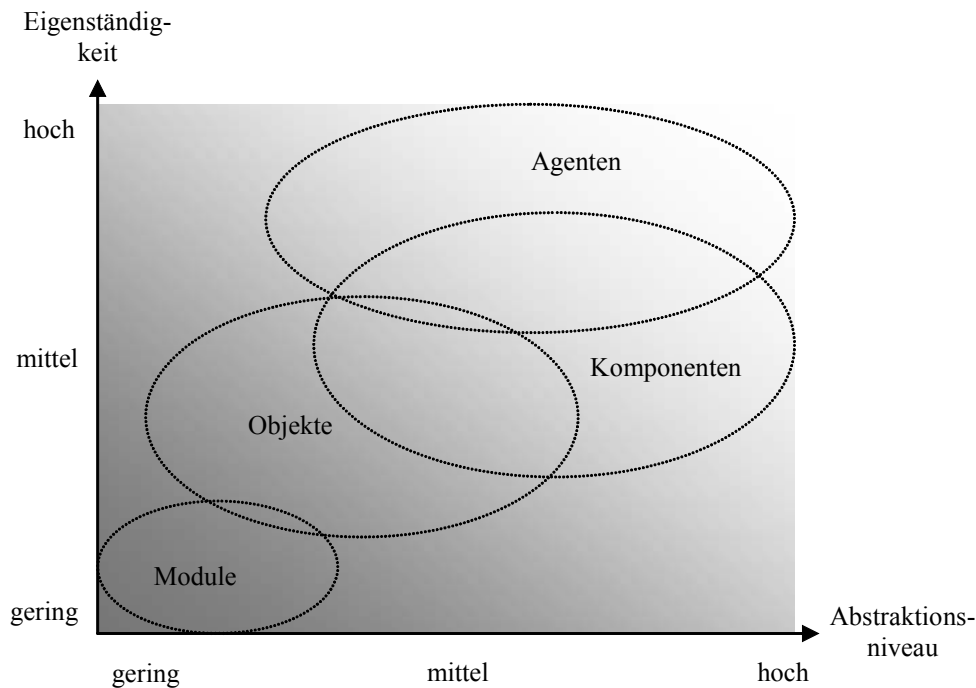


Abb. 2.5.1.1: Konzepte zum strukturierten Softwareentwurf im Vergleich

Die Bereichsüberschneidungen in Abb. 2.5.1.1 deuten an, dass in der Fachwelt über die Begriffe Modul, Objekt, Komponente und Agent noch längst kein allgemeiner Konsens besteht. Vielmehr ist gerade beim Begriff der Softwarekomponente häufig eine Vermischung mit den Begriffen Objekt und Agent festzustellen. Mit Softwarekomponenten werden meist die Eigenschaften Verteilung und Wiederverwendbarkeit assoziiert. Softwareagenten werden dagegen Eigenschaften wie Reaktivität<sup>12</sup>, Proaktivität<sup>13</sup>, kommunikative Kompetenz<sup>14</sup>, Mobilität und Unabhängigkeit (Autonomie) zugesprochen, wobei sich letztere nach dem an den Agenten übertragenen Verantwortungsgrad bemessen lässt. Unter Softwaremodulen versteht man i.a. Strukturen der nicht-objektorientierten Programmierung, während Objekte als gekapselte Datenstrukturen die Grundbausteine objektorientierter Programme darstellen. Je nach verwendeter Sprache und Vorgehensweise können jedoch sehr unterschiedliche Aspekte mit dem Objektbegriff verbunden sein, so dass es auch hier häufig zu Vermischungen mit den anderen in Abb.

<sup>12</sup> Fähigkeit auf Veränderungen wie beispielsweise externe Ereignisse zu reagieren

<sup>13</sup> Fähigkeit von sich aus Änderungen zielorientiert zu initiieren

<sup>14</sup> Fähigkeit mit anderen Agenten zu kommunizieren

2.5.1.1 dargestellten Begriffen kommt. Da objektorientierte Denkweisen und Techniken in vielen Fällen die Basis für die Entwicklung von komponentenorientierter Software bilden, ist gerade zwischen Objekten und Komponenten eine sehr enge Verwandtschaft festzustellen. Komponenten unterscheiden sich jedoch von Objekten bzw. Klassen in folgender Hinsicht:

- Im Gegensatz zu Komponenten sind Objekte meist nicht groß genug, um eine komplexe in sich geschlossene Aufgabenstellung abzudecken.
- Während Objekte in der Regel für eine bestimmte (Teil-)Aufgabe entwickelt werden, sollten Komponenten in verschiedenen Anwendungsszenarien einsetzbar und damit mehrfach wiederverwendbar sein.
- Da bei Komponenten verstärkt auf Verteilung, Wiederverwendbarkeit, Integrationsfähigkeit und Interoperabilität geachtet werden muss, fällt der dafür notwendige Entwicklungsaufwand wesentlich höher aus als bei herkömmlichen Objekten.

In Abschnitt 2.5.2.1 wird der Komponentenbegriff weiter vertieft. Ein fundiertes Verständnis dieses Begriffes und der damit verbundenen Merkmale ist wichtig, um die Stärken und Schwächen der heute miteinander konkurrierenden in Abschnitt 2.5.2.2 beschriebenen Komponentenstandards in Bezug auf unterschiedliche Anwendungsfelder wie beispielsweise der Modellierung und Simulation (siehe Kapitel 4) beurteilen zu können.

## 2.5.2 Komponentenorientierte Softwareentwicklung

Die Grundidee der komponentenorientierten Softwareentwicklung besteht darin, komplexe Softwaresysteme aus einfacheren vorgefertigten Komponenten zu erstellen (Legosteine-Metapher). Dabei wird einerseits maximale Unabhängigkeit bei der Entwicklung der Einzelteile, andererseits aber auch größte Flexibilität bei ihrer späteren (Re-)Kombination angestrebt. Von dieser Vorgehensweise verspricht man sich vor allem

1. eine Vereinfachung des SW-Entwicklungsprozesses durch Trennung von Verantwortlichkeiten<sup>15</sup>,
2. Anhebung des Abstraktionsniveaus der SW-Entwicklung auf Anwendungsniveau durch Bereitstellung von vorgefertigten Anwendungsbausteinen für unterschiedliche Anwendungsdomänen,
3. Vereinfachung der SW-Wartung, -Änderung, -Erweiterung und -Verteilung, die wiederum eine Fokussierung auf neuartige Anwendungen für bereits vorhandene Basissysteme ermöglicht,

---

<sup>15</sup> Bei der komponentenorientierten SW-Entwicklung werden zwei Arten von Entwicklern mit jeweils streng voneinander getrennten Verantwortungsbereichen angestrebt. Zum einen Anwendungsentwickler, deren Aufgabe darin besteht, komplexe Anwendungen aus vorgefertigten Komponenten zusammensetzen, wobei hohe Kompetenz bezüglich der zugrunde liegenden Anwendungslogik vorausgesetzt wird. Zum anderen technisch versierte Komponentenentwickler, die für die Realisierung der entsprechenden Komponenten zuständig sind.

4. einen unmittelbaren Beitrag zum Erreichen betriebswirtschaftlicher Ziele durch

- plattformübergreifende Nutzung von Komponenten,
- Wiederverwendung von Komponenten mit allgemeiner Bedeutung,
- Zukauf von fremd entwickelten Komponenten und deren Integration,
- Verkauf von selbst entwickelten Komponenten.

Im Folgenden wird zunächst der Komponentenbegriff näher erläutert. Im Anschluss daran werden die wichtigsten heute existierenden Komponentenstandards vorgestellt. Abschließend wird auf die eng mit Komponenten-Technologien in Beziehung stehenden Frameworks und Middleware-Systeme eingegangen.

### 2.5.2.1 Der Komponentenbegriff

Bislang existiert keine allgemein anerkannte Definition für den Begriff "Software-Komponente". In der Literatur finden sich eine Reihe teilweise sehr unterschiedlicher Begriffsdefinitionen. Eine bekannte und häufig zitierte Definition stammt von Clemens Szyperski [Szyp99]:

*Komponenten sind in Binärform vorliegende Lösungen softwaretechnischer Probleme, die so zusammengefügt werden können, dass sie ein lauffähiges Softwaresystem bilden. Sie können unabhängig voneinander hergestellt, erworben und eingesetzt werden. Dazu besitzen Komponenten eine feste Schnittstelle und enthalten potentiell auch eigene Ressourcen, auf die sie bei der Ausführung zurückgreifen.*

Da zur Zeit objektorientierte Techniken und Denkweisen als die wahrscheinlich beste verfügbare Ausgangsbasis für komponentenorientierte Software gelten, stößt man in diesem Zusammenhang häufig (etwa in [Same97]) auf die Definition:

*Komponenten = Objekte + irgendetwas.*

Komponenten müssen jedoch nicht zwingend objektorientiert realisiert sein. Wird von der Realisierungsform gänzlich abstrahiert, läuft die Diskussion um den Komponentenbegriff in jüngster Zeit auf folgende obligatorische Merkmale hinaus:

1. Komponenten stellen keine eigenständigen für sich alleine lebensfähigen Anwendungen dar, sondern dienen einem wohldefinierten Zweck innerhalb eines bestimmten Anwendungskontextes.
2. Komponenten stellen grundsätzlich Bausteine für die Wiederverwendung dar.
3. Komponenten verbergen (kapseln) ihre innere Struktur und Implementierung hinter einer strikt davon getrennten Schnittstelle und lassen sich daher vollständig unabhängig von jeglichen Implementierungsdetails spezifizieren.
4. Komponenten haben eine oder mehrere explizite Schnittstellen, über die sie fachlich kohärente Dienstleistungen anbieten.

5. Komponenten weisen über eine explizite Schnittstelle aus, welche Dienstleistungen sie von anderen Komponenten oder ihrer einbettenden Umgebung benötigen.

Neben diesen obligatorischen Merkmalen werden von Komponenten zunehmend folgende Eigenschaften verlangt [Grif98]:

6. Komponenten sollten in ausführbarer, auf verschiedenen Plattformen lauffähiger Form vorliegen (Binärcode-Verfügbarkeit) und sofort nach Auslieferung problemlos installierbar und nutzbar sein (plug & play).
7. Komponenten sollten möglichst übergreifend über Hardwareplattformen, Betriebssysteme, Programmiersprachen, Netzwerktechnologien und Entwicklungsumgebungen zusammenwirken können (Offenheit und Umgebungsunabhängigkeit).
8. Die zu einer solchen netzwerkübergreifenden Nutzung notwendigen Mechanismen sollten für den Benutzer nicht sichtbar sein (Ortstransparenz).
9. Zur Laufzeitkopplung sowie zur besseren Wiederverwendbarkeit sollte eine Komponente Auskunft über ihre eigenen Möglichkeiten und Zugriffspunkte geben können (selbst beschreibende Schnittstellen).
10. Komponenten sollten ein gewisses Maß an Konfigurierbarkeit besitzen, um sie einfach und schnell an neue Situationen und Anwendungskontexte anpassen zu können (Konfigurierbarkeit und Anpassbarkeit).
11. Unterschiedliche Versionen und Varianten einer Komponente sollten „spät“, d.h. beim Installieren, Laden oder während der Ausführung des Anwendungssystems ausgewählt werden können.
12. Eine Komponente sollte mit anderen Komponenten flexibel zu einer neuen, funktionsfähigen Komponente zusammengesetzt werden können (Integrations- und Kompositionsfähigkeit).
13. Komponenten sollten ausgiebig erprobt und sorgfältig getestet werden, um eine hohe Zuverlässigkeit zu gewährleisten (Bewährtheit).

Die oben aufgeführte Liste von Komponentenmerkmalen stellt ein wichtiges Hilfsmittel dar, um zu überprüfen, ob sich ein Komponentenstandard für einen bestimmten Einsatzzweck eignet oder nicht. Auf Komponentenstandards, die häufig auch als Komponentenmodelle bezeichnet werden, wird im folgenden Abschnitt näher eingegangen.

### **2.5.2.2 Komponentenstandards**

Durch einen Komponentenstandard wird ein verbindlicher Rahmen für die Entwicklung und Ausführung von Komponenten festgelegt, der strukturelle Anforderungen hinsichtlich Verknüpfungs- bzw. Kompositionsmöglichkeiten sowie verhaltensorientierte Anforderungen hinsichtlich Kollaborationsmöglichkeiten an die Komponenten stellt [GrTh00].

Zu den bedeutendsten Komponentenstandards, die heute zur Verfügung stehen, zählen

- (Enterprise) JavaBeans

JavaBeans stellt ein Komponentenmodell dar, das von der Firma Sun Microsystems speziell für die Programmiersprache Java entwickelt wurde. Das zugrunde liegende Konzept sieht im Wesentlichen einige wenige Entwurfsregeln vor, deren Einhaltung automatisch zu einer JavaBeans konformen Softwarekomponente führt, die mit einem so genannten Builder-Tool visuell manipuliert werden kann. JavaBeans erfordern daher keine Änderung oder Erweiterung der zugrunde liegenden Programmiersprache, sondern stellen Java-Klassen dar, die einem bestimmten in der JavaBeans-Spezifikation beschriebenen Schnittstellenprotokoll gehorchen. Neben JavaBeans hat Sun Microsystems mit Enterprise JavaBeans (EJB) ein weiteres Komponentenmodell entwickelt und standardisiert. Trotz der ähnlichen Bezeichnungen weisen beide Komponentenmodelle gravierende Unterschiede auf. Im Gegensatz zu JavaBeans, die vor allem auf Client-Seite eingesetzt werden und keinen verteilten Charakter aufweisen, handelt es sich bei Enterprise JavaBeans um ein dazu komplementäres Konzept, das speziell auf die serverseitige Nutzung im Rahmen von verteilten transaktionsorientierten Geschäftsanwendungen zugeschnitten ist. Weitere Details zu diesen Komponentenmodellen finden sich in [ONei98], [RoAJ01] und [DePe00].

- (D)COM/COM+/.NET

(D)COM, COM+ sowie .NET stellen Komponenten-Technologien der Firma Microsoft dar, die aufgrund ihrer langjährigen evolutionären Entwicklung als Vorreiter in diesem Bereich angesehen werden können. Das Rückgrat dieser Technologien, die eher aus einem natürlichen Produktpflegeprozess als aus einer klaren Konzeption hervorgegangen sind, bildete bislang das Component Object Model (COM) bzw. dessen verteilte Version, das Distributed Component Object Model (DCOM). COM+ stellt eine evolutionäre Weiterentwicklung von DCOM dar, während das .NET-Framework Microsofts neueste sich noch in Entwicklung befindende Komponentenplattform darstellt, die unter anderem eine gemeinsame Laufzeitbibliothek und ein gemeinsames Typsystem für alle von Microsoft unterstützten Programmiersprachen bietet. Die Zielsetzung dieser Plattform, welche als die Basis der .NET-Gesamtinitiative von Microsoft aufgefasst werden kann, besteht vor allem in der Verbesserung der Entwicklungsmöglichkeiten unter Windows sowie der Unterstützung von Techniken für die Entwicklung verteilter Webdienste (Web Services) und Anwendungen. So ist beispielsweise das Datenformat XML für den standardisierten Austausch von Daten über das Web vollständig in .NET eingebunden. Weitere Details zu den Microsoft Komponentenstandards finden sich in [Löwy01] und [Nath02].

- CORBA Component Model (CCM)

CCM wurde vom Standardisierungskonsortium OMG<sup>16</sup> (Object Management Group) entwickelt und ist ein wichtiger Bestandteil der CORBA 3.0 Spezifikation [COR1]. Auf CORBA (Common Object Request Broker Architecture), das eine Middleware zur plattformübergrei-

---

<sup>16</sup> Die OMG wurde im April 1989 von elf Firmen (darunter Sun Microsystems, 3Com, Hewlett-Packard und Canon) gegründet. Ziel der OMG ist es, Standards und Spezifikationen für eine Infrastruktur zu entwickeln, die für verteilte objektorientierte Anwendungen erforderlich ist. Heute arbeitet die OMG als gemeinnützige Organisation, der mittlerweile über 800 Unternehmen angehören. Weitere Informationen zur OMG finden sich im WWW unter [OMG1].

fenden Kooperation verteilter Objekte darstellt, wird ausführlich in Abschnitt 2.5.3 eingegangen. CCM kann als eine Erweiterung des CORBA-Objektmodells aufgefasst werden, das zwar stark an konkurrierenden Konzepten wie EJB und den neueren Microsoft-Komponentenstandards orientiert ist<sup>17</sup>, sich aber im Gegensatz zu diesen nicht auf eine festgelegte Programmiersprache oder eine Betriebssystemplattform beschränkt. Das CCM ist explizit auf Server-seitige Komponenten mit relativ grober Granularität festgelegt. Diese können in beliebigen Programmiersprachen implementiert werden, solange sie über eine CORBA-konforme Schnittstellenbeschreibung verfügen. Komponenten residieren zur Laufzeit in so genannten Containern, welche den Komponenten über Programmierschnittstellen bestimmte Dienste (Transaktion, Sicherheit, Persistenz, Lebenszyklus, etc.) anbieten. Ein solcher Container wiederum nutzt die Dienste eines darunter liegenden ORB (Object Request Broker) bzw. eines Applikationsservers. Weitere wichtige Bestandteile des CCM sind ein Komponenten-Server, der eine beliebige Anzahl von Komponenten-Containern enthalten kann sowie ein standardisierter Entwicklungs- und Distributionsprozess. Ein einheitliches Distributionsformat sorgt dafür, dass Komponenten zwischen unterschiedlichen Container Implementierungen ausgetauscht werden können. Abschließend sei angemerkt, dass das CCM zwar den konzeptionell ausgereiftesten Komponentenstandard darstellt, eine konkrete Implementierung dieses Standards bislang jedoch noch nicht verfügbar ist.

Die oben genannten Komponentenmodelle basieren alle auf denselben grundlegenden Prinzipien und Mustern. Kernprinzip ist die Trennung technischer und funktionaler Belange, um diese möglichst unabhängig voneinander und durch verschiedene Personen verändern und weiter entwickeln zu können. Zu den technischen Belangen zählen beispielsweise Sicherheit, Transaktion, Nebenläufigkeit, Ressourcenmanagement und Persistenz. Bei den funktionalen Belangen handelt es sich um die fachlichen Anforderungen des Systems. Die Gesamtfunktionalität einer Anwendung wird in einzelne Komponenten aufgeteilt, die ausschließlich über ihre Schnittstellen miteinander kommunizieren. Damit die Gesamtanwendung richtig funktionieren kann, müssen jedoch auch die technischen Belange berücksichtigt werden. Dazu werden die bereits im Zusammenhang mit CCM erwähnten Container angeboten, die eine Ablaufumgebung für Komponenten darstellen und Zugriff auf bestimmte von verteilten Anwendungen immer wieder benötigte Dienste ermöglichen.

Eine weitere wichtige Gemeinsamkeit der oben genannten Komponentenstandards besteht in der Bereitstellung mächtiger Kommunikationsinfrastrukturen, die den Aufbau mehrschichtiger Systemarchitekturen ermöglichen, in denen die Kommunikation zwischen den beteiligten Komponenten in der Regel nach dem Client/Server-Prinzip erfolgt. Dadurch lassen sich Gesamtanwendungen flexibel auf verschiedene Rechner verteilen und damit ausfallsicherer und skalierbarer machen. Auf solche Kommunikationsinfrastrukturen, die auch als Middleware-Systeme bezeichnet werden, wird in Abschnitt 2.5.2.4 näher eingegangen.

Viele Komponentenmodelle bauen auf die Entstehung eines boomenden Komponentenmarktes, auf dem sich preiswert und schnell fertige Subsysteme einkaufen lassen. Bislang existieren jedoch nur für sehr wenige Domänen solche Märkte<sup>18</sup>. Einen neuen Schub diesbezüglich ver-

---

<sup>17</sup> Es existiert eine standardisierte Abbildung, mit deren Hilfe EJBs als CCM-Komponenten (oder umgekehrt) betrachtet werden können.

<sup>18</sup> Den heute wohl erfolgreichsten Komponentenmarkt stellen grafische Oberflächenelemente für GUIs dar. Die verwendeten Komponenten-Technologien sind Microsofts VBX (bzw. OCX) oder JavaBeans von Sun Microsystems.



spricht man sich vor allem durch die heute sehr populären Web-Services, die als eine Weiterentwicklung des Konzepts der komponentenorientierten Softwareentwicklung aufgefasst werden können hin zu in weltweiten Netzwerken verteilten, lose gekoppelten Anwendungen, die in der Lage sind, sich zur Erbringung einer Dienstleistung dynamisch zusammenzuschließen. Dazu ist es notwendig, die Web Services völlig gekapselt zu implementieren und in eine standardisierte logische Infrastruktur einzubinden, welche die automatische Lokalisierung und Einbindung eines Dienstes durch einen anderen Dienst ermöglicht. Unter einem Web Service wird dabei eine verteilte, lose gekoppelte und wiederverwendbare Software-Komponente verstanden, auf die über Standard-Internetprotokolle zugegriffen werden kann. Durch eine konsequente Nutzung von Standards wird Unabhängigkeit von bestimmten Plattformen, Betriebssystemen und Programmiersprachen erreicht.

Web Services zielen auf den Aufbau einer so genannten dienstorientierten Architektur<sup>19</sup> ab, in der Anbieter (Provider) ihre Applikationen in einem zentralen föderierten Verzeichnis registrieren und dort alle nötigen Informationen über deren Gebrauch ablegen. Über dieses Verzeichnis werden potentielle Nutzer (Requestor) von Diensten über Vermittler (Broker) mit Dienst Anbietern zusammengebracht, wobei die Broker unter anderem als Zahlungsstelle dienen.

Web Services, die technisch gesehen keine wesentliche Neuerung darstellen, können als eine Vielzahl von ineinander greifenden Standards aufgefasst werden. Bei ihrer Spezifikation durch das World Wide Web Consortium (W3C) wurde vor allem auf den konsequenten Gebrauch von Web-Standards geachtet. Neben dem durchgängig verwendeten Beschreibungsformat XML, sind als weitere Schlüsseltechnologien für die Funktionalität, Architektur und Distribution von Web Services folgende W3C-Standards zu nennen:

- SOAP (Simple Object Access Protocol)

SOAP stellt ein Protokoll für entfernte Methodenaufrufe über das Hyper Text Transfer Protokoll (HTTP) dar, wobei die ausgetauschten Nachrichten in XML kodiert sind. Die Verwendung von HTTP hat den Vorteil, dass die Firewall-Problematik<sup>20</sup>, die den Einsatz anderer Middleware-Technologien wie beispielsweise CORBA, RMI und DCOM schwierig macht, umgangen wird.

- WSDL (Web Services Description Language)

WSDL stellt eine formale Beschreibung der Eigenschaften und Funktionen von Web Services analog zum Konzept der Interface Definition Language (IDL) bei CORBA dar.

- UDDI (Universal Description, Discovery, and Integration)

UDDI stellt einen DNS-ähnlichen verteilten Verzeichnisdienst für Web Services dar, der es Anwendern und Anwendungen erlaubt, Informationen über Web Services Internet-weit zu veröffentlichen und zu finden, wobei die Beschreibungen in einem einheitlichen XML-Format abgelegt werden.

---

<sup>19</sup> Ein neues Paradigma für verteilte Systeme ist die Sichtweise, alles als Dienst aufzufassen.

<sup>20</sup> das HTTP-Protokoll ist in vielen Firewalls freigeschaltet

Web Services sind heute vor allem deshalb von so großer Bedeutung, weil sich praktisch die gesamte Software-Industrie zu diesem offenen Standard verpflichtet hat. Bedeutende Software-Hersteller wie beispielsweise Microsoft, Sun Microsystems, IBM, Oracle und Hewlett-Packard haben eine breite und aktive Unterstützung dieser Technologie angekündigt und mittlerweile ihre Produktpalette stark auf Web Services ausgerichtet. Beim Einsatz von Web Services ist jedoch zu beachten, dass wesentliche Teile dieses Standards derzeit noch definiert werden. Dies betrifft vor allem die Bereiche Sicherheit, Authentifizierung und Abrechnungsmöglichkeiten. So lange diese Unzulänglichkeiten und Lücken bei den Web-Service-Standards nicht behoben sind, empfiehlt sich aufgrund der zu erwartenden Fluktuationen und Änderungen ein eher vorsichtiger Umgang mit dieser neuen Technologie.

Neben den bisher genannten Komponentenstandards existiert eine Reihe weiterer Standards, die meist auf spezielle Anwendungsfelder spezialisiert sind. Drei prominente Beispiele solcher Spezialstandards sind

- Java Intelligent Network Infrastructure (JINI)

Sun Microsystems hat schon vor Jahren die Idee propagiert, Software als Dienstleistung zu nutzen und diesbezüglich eine Lösung namens JINI entwickelt. JINI stellt eine Architektur dar, welche die spontane Komposition und Interaktion bzw. Plug-and-Play-Kompatibilität von Diensten bzw. Geräten in einer Föderation ermöglicht. Dazu werden von JINI eine Infrastruktur und ein Programmiermodell bereitgestellt. Die zentrale Komponente dieser Infrastruktur ist der so genannte Lookup Service, der analog zu UDDI einen Verzeichnisdienst darstellt, bei dem sich alle verfügbaren Dienste registrieren. Klienten können im Lookup Service nach Diensten suchen, die sie interessieren und erhalten als Antwort ein so genanntes Proxy-Objekt, das mobilen Java-Code darstellt. Diese Proxy-Objekte nehmen eine besondere Rolle ein, weil sie die direkte Interaktion mit dem Dienst vollständig übernehmen. JINI basiert also wie die Web Services auch auf einer dienstorientierten Sichtweise, bei der es grundsätzlich darum geht, in einer hochgradig dynamischen Umgebung Dienste und Klienten, die spontan auftauchen und wieder verschwinden können, zu vermitteln. Trotz dieser gemeinsamen Sichtweise gibt es jedoch auch einige Unterschiede. Während Web Services auf allgemeinen Standards wie XML und SOAP aufsetzen, handelt es sich bei JINI um eine Java-zentrierte Technologie, die hauptsächlich auf dem Java-spezifischen Kommunikationskonzept RMI (Remote Method Invocation), serialisierbaren Java-Objekten und dem Java-basierten Tupelraumkonzept JavaSpaces basiert. Des Weiteren ist JINI vor allem auf die spontane Vernetzung von Diensten ausgerichtet, die von eingebetteten Systemen zur Verfügung gestellt werden.

- Voyager

Voyager wurde von der Firma ObjectSpace entwickelt und stellt eine zu Java vollkommen kompatible Laufzeitumgebung und Bibliothek dar, die umfassende Lösungen für viele Bereiche und Problemfelder der verteilten Programmierung anbietet. Analog zu CORBA werden mächtige, einfach zu benutzende Kommunikationskonzepte bereitgestellt, die neben der entfernten Nutzung gewöhnlicher Java-Objekte auch deren Mobilität und Ausbau zu Agenten erlauben.

- High Level Architecture (HLA)

Die High Level Architecture wurde vom US-amerikanischen Verteidigungsministerium initiiert und stellt einen speziell auf die Bedürfnisse verteilter Simulationen zugeschnittenen Komponentenstandard dar. Auf die HLA wird detailliert in Abschnitt 4.1 eingegangen.

### 2.5.2.3 Frameworks

Nach [RePo02] können Frameworks als Organisationsformen von Sammlungen objektorientierter Komponenten aufgefasst werden. Im Gegensatz zu Komponenten, die abgegrenzte technische bzw. anwendungsfachliche (Teil-)Lösungen darstellen, unter denen der Anwendungsentwickler auswählen kann, bestimmen Frameworks die Anwendungsarchitektur im Großen und stellen die wesentlichen Grundkonzepte und Lösungsschemata dieser Architektur abstrakt bereit [WRKZ01]. Beide Ansätze verfolgen grundsätzlich die gleichen Ziele, nämlich Wiederverwendbarkeit in größerem Stil und bessere Strukturierung von Anwendungssystemen. Technisch unterscheiden sie sich jedoch beträchtlich. Wie bereits angedeutet, stellen Frameworks eine Menge von zusammenarbeitenden Klassen dar, die einen wieder verwendbaren Entwurf für einen bestimmten Anwendungsbereich implementieren. Beim Einsatz von Frameworks wird also vor allem programmiert, wobei die vorgefertigten Klassen für die eigene Anwendungsentwicklung benutzt oder beerbt werden. Beim Einsatz von Komponenten steht dagegen die Idee der Komposition im Vordergrund. Vererbung wird gar nicht verwendet und die Komponenten werden als Black-Boxes betrachtet. Programmiert wird im Idealfall nur minimal, um den zur Verbindung der Komponenten notwendigen so genannten Glue-Code herzustellen.

Trotz ihrer signifikanten Unterschiede eignen sich Komponenten- und Framework-Technologien sehr gut für den komplementären Einsatz. Naheliegender wäre beispielsweise dasselbe Framework als integrierenden Rahmen für eine Familie von Komponenten zu verwenden. In diesem Fall könnten die Einzelkomponenten auf einer einheitlichen technischen Infrastruktur der verwendeten Programmiersprache und des verwendeten Komponentenmodells aufbauen. Hinzu kommen die Basiskonzepte und Abstraktionen des zugrunde liegenden Frameworks, wie etwa ein einheitlicher Benachrichtigungsmechanismus, eine Interpretation des Vertragsmodells oder die durchgehend einheitliche Verwendung von Wert- und Referenzsemantik.

### 2.5.2.4 Middleware-Systeme

Der Begriff Middleware, zu dem es ebenfalls bisher keine scharf umrissene Definition gibt, steht in sehr enger Beziehung zu Client/Server Systemen. Middleware stellt die Bindung im Begriffspaar Client/Server dar und kann als Software-Schicht aufgefasst werden, die zwischen Betriebssystem und Anwendungsebene angesiedelt ist. In Zusammenhang mit komponentenorientierten Anwendungen versteht man darunter Kommunikationsinfrastrukturen, die eine flexible und dynamische Verteilung von Komponenten auf heterogene physikalisch möglicherweise weit voneinander entfernt lokalisierte Plattformen erlauben. Eine solche Verteilung erfordert zunächst einmal die Definition eines abstrakten auf verschiedene Kommunikationstechnologien abbildbaren Kommunikationsmodells. Darüber hinaus müssen zentrale Dienste bereitgestellt werden, die in verteilten Anwendungen immer wieder benötigt werden. Dabei handelt es sich beispielsweise um die globale Verwaltung von Transaktionen (Transaktions-

dienst), die Zuordnung von Namen mit Objekten bzw. Objektreferenzen (Namensdienst) sowie die Unterstützung von Sicherheitsaspekten (Sicherheitsdienst). Das Kommunikationsmodell zusammen mit den zur Verfügung gestellten zentralen Diensten wird schließlich als Middleware-System bezeichnet.

Anhand des Kommunikationsmodells lässt sich innerhalb der heute existierenden Middleware-Lösungen grob folgende Unterteilung vornehmen:

### 1.) Middleware zur direkten synchronen eng gekoppelten Kommunikation

Das Kommunikationsmodell dieser traditionellen Art von Middleware basiert auf Prozedur-Fernauffufen (Remote Procedure Call (RPC)) bzw. Methoden-Fernauffufen (Remote Method Invocation (RMI)), welche das Pendant der Prozedur-Fernauffufe in der objektorientierten Welt darstellen. Der Programmierer muss dabei nur die Prozedur- bzw. Methodenschnittstelle spezifizieren. Der Programmcode, der die Kommunikation zwischen Client- und Server-Komponente ermöglicht, wird automatisch erzeugt. Tabelle 2.5.2.4.1 zeigt weit verbreitete Komponentenstandards, deren Kommunikationsmodelle auf traditionellen Prozedur- bzw. Methoden-Fernauffufskonzepten basieren.

Komponentenstandard	EJB	CCM	DCOM	Web Services
verwendetes Prozedur- bzw. Methoden-Fernauffufskonzept	Java RMI	CORBA Methodenfernauffuf	DCP (Distributed Computing Environment) RPC	SOAP (Simple Object Access Protocol)

*Tabelle 2.5.2.4.1: Komponentenstandards mit traditionellen Prozedur- bzw. Methodenfernauffufskonzepten*

Bei RPC- bzw. RMI-basierter Middleware wird eine direkte Verbindung zwischen den kommunizierenden Komponenten (Client und Server) vorausgesetzt, die beide gleichzeitig aktiv sein müssen. Die Client-Anwendung kann i.a. erst dann weiterarbeiten, wenn der Server deren Anfrage beantwortet hat (verbindungsorientierte synchrone Kommunikation). Weitere Details zu dieser Art von Middleware-Systemen finden sich in [PuRö01], [Myer02] und [SeCr02].

### 2.) Middleware zur indirekten asynchronen lose gekoppelten Kommunikation

Bei dieser so genannten Message-Oriented Middleware (MOM) basiert das zugrunde liegende Kommunikationsmodell auf dem Austausch von Nachrichten (message passing). Diese werden von Quell-Komponenten (Produzenten) erzeugt und zu Ziel-Komponenten (Konsumenten) transportiert. Produzenten und Konsumenten müssen dabei nicht gleichzeitig aktiv sein, da die MOM Nachrichten puffern kann (verbindungslose asynchrone "store-and-forward" Kommunikation) und damit eine weitaus stärkere Entkopplung von Client und Server ermöglicht als

RPC- bzw. RMI-basierte Middleware. MOM-Systeme lassen sich wiederum in verschiedene Kategorien einteilen wie z.B. Publish/Subscribe- und Message Queuing-Systeme. Weitere Details zu MOM-Systemen finden sich in [AIEr99] und [BCSS99]. Abschließend sei angemerkt, dass sich MOM-Systeme insbesondere für ereignisorientierte Anwendungen eignen. Die Kommunikationsinfrastruktur der HLA, die ausführlich in Abschnitt 4.1.4 beschrieben wird, stellt daher ein solches MOM-System dar.

## 2.5.3 Common Object Request Broker Architecture

Die Common Object Request Broker Architecture (CORBA) ist eine vom Standardisierungskonsortium OMG erarbeitete Spezifikation, welche die Zusammenarbeit von Softwaresystemen über die Grenzen von Programmiersprachen, Betriebssystemen und Netzwerken hinweg regelt. Im Folgenden wird zunächst mit der ebenfalls von der OMG entwickelten Object Management Architecture (OMA) der Ausgangspunkt für die Entstehung von CORBA vorgestellt. Im Anschluss daran wird die Object Request Broker (ORB) Architektur beschrieben, die den Kern der OMA bildet. Abschließend wird auf Protokolle zur Gewährleistung von Interoperabilität zwischen ORBs unterschiedlicher Hersteller eingegangen.

### 2.5.3.1 Object Management Architecture

Die Basis für die von der OMG vorgenommene Standardisierung bildet die in Abb. 2.5.3.1.1 dargestellte Object Management Architecture (OMA), welche die Verteilung und Zusammenarbeit objektorientierter Softwarekomponenten in heterogenen und vernetzten Systemen ermöglichen soll.

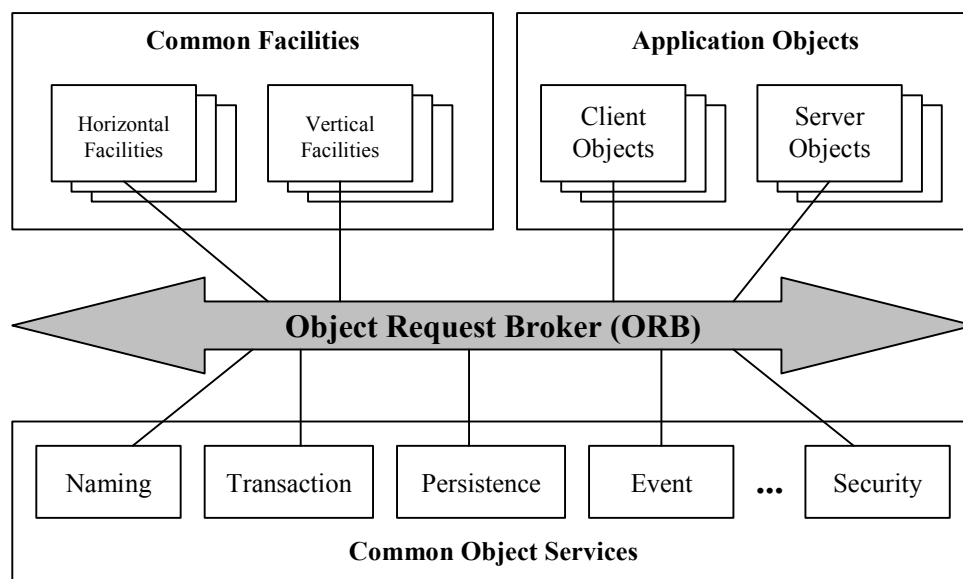


Abb. 2.5.3.1.1: Die Object Management Architecture (OMA)

Die wesentlichen Ziele, die mit der OMA verfolgt werden, bestehen in der Definition einer einheitlichen Terminologie, der Partitionierung des komplexen Problemraums und der Definition eines Objektmodells. Die OMA wird daher auch als die Referenzarchitektur der OMG bezeichnet, deren Bestandteile von der OMG im Laufe der Zeit mit Leben gefüllt wurden bzw. immer noch werden. Den Kern der OMA bildet der Object Request Broker (ORB). Dieser stellt eine Kommunikationsinfrastruktur zur Verfügung, die einen weitgehend transparenten Informationsaustausch zwischen Client- und Serverobjekten ermöglicht, unabhängig von spezifischen Plattformen und Techniken der Objektimplementierung. Da von dieser Architekturkomponente alle anderen Komponenten der OMA abhängen, wurde sie von der OMG unter dem Namen CORBA als erstes standardisiert.

Bei den weiteren Komponenten der OMA handelt es sich um Anwendungsobjekte (Application Objects), CORBA-Dienste (Common Object Services) und so genannte CORBA-Facilities (Common Facilities), die im Folgenden näher erläutert werden:

- Anwendungsobjekte: Objekte bzw. Komponenten, die nicht Bestandteil der Standardisierung sind, sondern die vom Entwickler einer CORBA-Anwendung selbst entwickelt werden, stellen CORBA-Anwendungsobjekte dar. Grundsätzlich wird unterschieden zwischen Objekten, die Dienste anbieten (Server-Objekte) und solchen, die Dienste in Anspruch nehmen (Client-Objekte), wobei Client-Objekte gleichzeitig auch Server-Objekte sein können und umgekehrt. Ein CORBA-Anwendungsobjekt kann sowohl ein einfaches Objekt als auch eine komplexe aus vielen Einzelobjekten zusammengesetzte Softwarekomponente sein. Auch in nicht-objektorientierten Sprachen geschriebene Programme können durch einen so genannten Object Wrapper<sup>21</sup>, der ihre Funktionalität hinter einer objektorientierten Schnittstelle kapselt, CORBA-fähig gemacht werden.
- CORBA-Dienste: Aufgabe der CORBA-Dienste ist es, die Grundfunktionalität des ORB auf Systemebene zu ergänzen bzw. zu erweitern. Diese Dienste können wie Anwendungsobjekte durch entfernten Methodenaufruf verwendet werden. Zu den wichtigsten CORBA-Diensten zählen:
  - Der Namensdienst, der einen zentralen Dienst in einer verteilten Objektumgebung darstellt und Objekten das Auffinden anderer bereits registrierter Objekte anhand ihres Namens ermöglicht.
  - Der Transaktionsdienst, der die Ausführung verteilter Transaktionen unterstützt, an denen mehrere Objekte beteiligt sind und sicherstellt, dass Transaktionen entweder komplett oder gar nicht ausgeführt werden.
  - Der Persistenz-Dienst, der das persistente Speichern von Objekten auf nicht-flüchtigem Speicher (Datei, Datenbank) ermöglicht.
  - Der Ereignis-Dienst, der es Objekten erlaubt, Interesse an bestimmten Ereignissen anzuzeigen, die von anderen Objekten ausgelöst werden. Dabei müssen Objekte nicht permanent verbunden sein, um das Auftreten von Ereignissen zu überwachen, da der Ereignisdienst dafür zuständig ist, Ereignisse vor deren Auslieferung zu speichern.

---

<sup>21</sup> Für den Begriff Wrapper wird häufig auch der Begriff Adapter verwendet.

- Der Sicherheitsdienst, der es ermöglicht, Nachrichten zu authentifizieren, den Objektzugriff zu autorisieren und damit eine sichere Kommunikation umzusetzen.
- CORBA-Facilities: Dabei handelt es sich um ein komplettes Rahmenwerk anwendungsspezifischer Dienste, die wiederum in horizontale, funktionsspezifische (Benutzerschnittstellen, Systemmanagement) und vertikale, auf spezielle Anwendungsdomänen (mobile Agenten, Workflow Management, Business Objects) zugeschnittene Dienste unterteilt werden.

### 2.5.3.2 Die ORB-Architektur

Zur Programmierung von CORBA-Anwendungen benötigt man eine Programmiersprache, die CORBA unterstützt. Dazu gehören heute z.B. C, C++, Smalltalk, Ada, Cobol und natürlich Java.

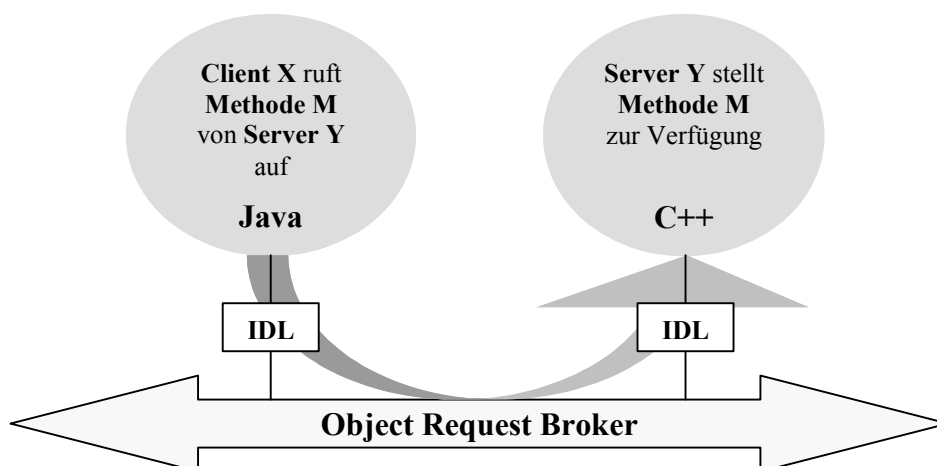


Abb. 2.5.3.2.1: Kommunikation zwischen Client- und Server-Objekt über den ORB

Damit die heterogenen in verschiedenen Programmiersprachen geschriebenen und auf unterschiedlichen HW-Plattformen ablaufenden Teilstücke einer CORBA-Anwendung miteinander kooperieren können, stellt CORBA die Schnittstellenbeschreibungssprache IDL (Interface Definition Language) zur Verfügung. Syntaktisch ist die IDL stark an C++ angelehnt, besitzt jedoch, da sie ausschließlich zur einheitlichen implementierungsunabhängigen Beschreibung der exportierten Objektattribute und -methoden dient, keine algorithmischen Elemente.

Abb. 2.5.3.2.1 zeigt das Prinzip der CORBA-basierten objektorientierten Client/Server-Kommunikation. In diesem Beispiel ruft ein in Java geschriebenes Client-Objekt eine Methode eines in C++ implementierten Server-Objekts auf. Für den Client ist der Prozess der Lokalisierung des Server-Objekts verdeckt, d.h. Objekte werden durch Objektreferenzen identifiziert, die nicht die Kenntnis des aktuellen Rechners, auf dem sich das Objekt befindet, voraussetzen (transparente Verteilung von Objekten). Zum besseren Verständnis des ORBs wird seine Ar-

chitektur in Abb. 2.5.3.2.2 weiter aufgeschlüsselt. Zentraler Bestandteil ist der ORB-Kern, der als Transportschicht angesehen werden kann.

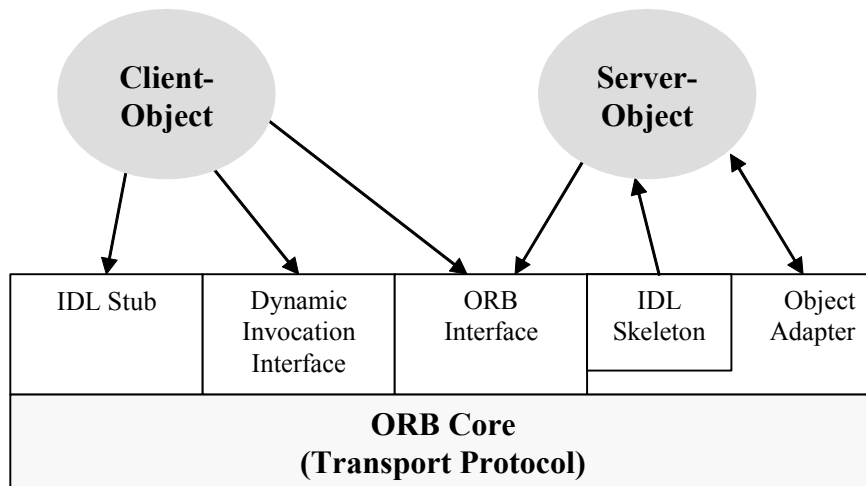


Abb. 2.5.3.2.2: Die ORB-Architektur entsprechend dem CORBA-Standard

Objekte, die Dienstleistungen zur Verfügung stellen bzw. in Anspruch nehmen wollen, müssen die entsprechenden Operationen in einer Schnittstelle beschreiben, die mit der IDL spezifiziert wird. IDL-Schnittstellenbeschreibungen werden durch so genannte IDL-Compiler in die jeweilige Programmiersprache der an der Kommunikation beteiligten Client- bzw. Server-Objekte übersetzt. Die zwei zentralen Produkte des IDL-Übersetzungsprozesses sind folgende Dateien:

- IDL-Stub: Stellt Routinen zur Verfügung, die das Client-Objekt benutzt, um Dienstleistungen des Server-Objekts anzufordern.
- IDL-Skeleton: Nimmt auf Server-Seite Nachrichten vom ORB entgegen und gibt diese als Methodenaufrufe an das dahinterliegende Server-Objekt weiter.

Client Objekte können mit statischen oder dynamischen Methodenaufrufen arbeiten. Statische Methodenaufrufe werden unter Verwendung der entsprechenden IDL-Stubs zur Übersetzungszeit generiert. Dynamische Methodenaufrufe sind flexibler, da sie dem Client-Objekt die Definition und Verwendung entfernter Objekte zur Laufzeit erlaubt. Zur Lösung dieser komplexen Aufgabe wird von CORBA das Dynamic Invocation Interface zur Verfügung gestellt. Als letzte Komponente der in Abb. 2.5.3.2.2 dargestellten ORB-Architektur sei noch der Objekt-Adapter erwähnt, der den Server-Objekten eine Schnittstelle zu den vom ORB angebotenen Dienstleistungen anbietet.

### 2.5.3.3 ORB-Interoperabilität

Um Interoperabilität zwischen ORBs unterschiedlicher Hersteller zu gewährleisten, wurde das General Inter ORB Protocol (GIOP) definiert. GIOP wurde sehr allgemein für alle Arten verbindungsorientierter Transportprotokolle entworfen, die lediglich ein Minimum an Voraus-



setzungen erfüllen müssen. IIOP (Internet Inter ORB Protokoll) stellt einen Spezialfall des GIOP dar, wobei hier auf dem TCP/IP Protokoll aufgesetzt wird. Ab CORBA in der Version 2.0 sind sowohl GIOP als auch IIOP integrale Bestandteile der CORBA-Spezifikation. Abb. 2.5.3.3.1 zeigt zwei über das GIOP miteinander verbundene ORBs.

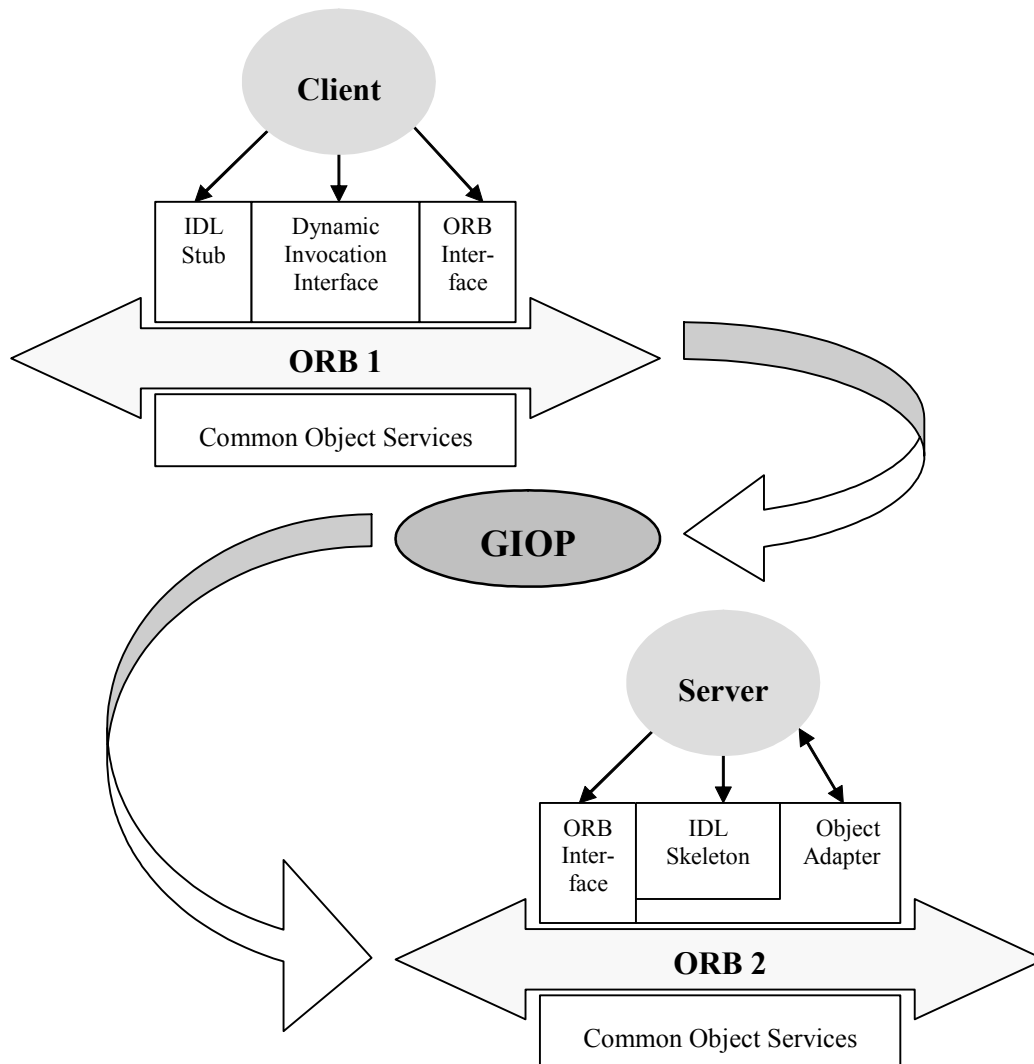


Abb. 2.5.3.3.1: Verbund verschiedener ORBs über GIOP



## *Kapitel 3*

# **Web-Technologien in der Modellierung und Simulation**

---

Kurz nachdem die ersten Web-Browser zur Verfügung standen, wurden diese bereits dazu verwendet, um Simulationen auf entfernten Rechnern auszuführen. Dabei wurden zunächst CGI-Programme eingesetzt, um die Simulationen zu starten und deren Ergebnisse zurückzuliefern. Gleichzeitig wurde damit begonnen, Java-basierte Simulationspakete und -umgebungen zu entwickeln. Erste Ergebnisse dieser Arbeiten wurden 1996 auf der Winter Simulation Conference [BrSw96] im Rahmen einer Session namens Web-based Simulation vorgestellt. Dieses Ereignis gilt heute als die Geburtsstunde der Web-basierten Modellierung und Simulation, einem interdisziplinären Forschungsbereich, der Ende der neunziger Jahre parallel zur Internet-Euphorie wohl den Höhepunkt seiner bisherigen Entwicklung erlebte. Zu den bedeutendsten Konferenzen, welche dieser Bereich hervorbrachte, zählen [FiHS98], [BrUP99], [SiBl00], [SiWM01] und [SiWi02]. Heute ebbt das Interesse an Web-Technologien innerhalb der Simulationsgemeinde allmählich ab und verlagert sich zunehmend auf die damit eng in Beziehung stehenden Komponenten- und Agenten-Technologien.

Das nun folgende Kapitel befasst sich zunächst einmal etwas näher mit dem Begriff "Web-basierte Simulation". Zu den weiteren Themenschwerpunkten dieses Kapitels zählen XML-basierte Modellaustauschformate, Architekturkonzepte für Simulationsanwendungen im Web, Java-basierte Simulationspakete, Web-basierte Simulationsumgebungen sowie der Einsatz von Web3D-Technologien zur 3-dimensionalen Visualisierung simulierter Abläufe.

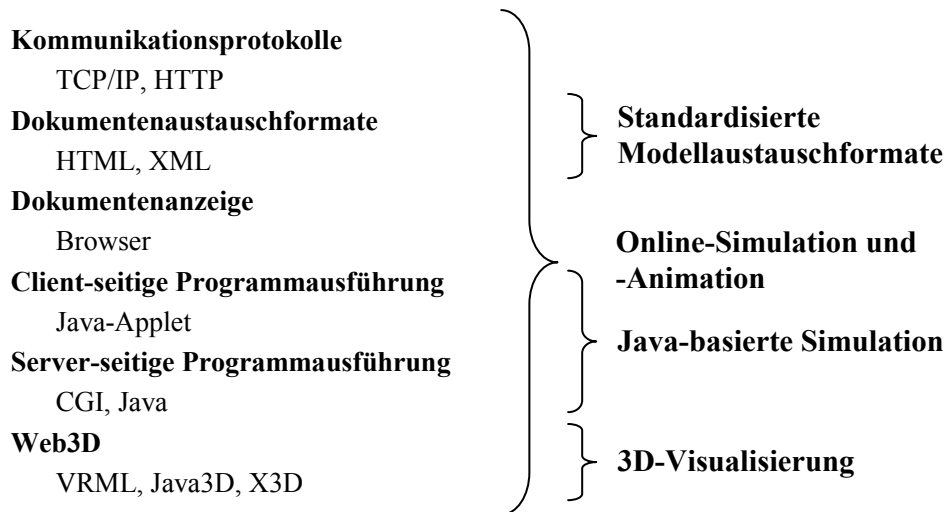
## **3.1 Einführung**

Für den Begriff Web-basierte Simulation gibt es bislang keine eindeutige Definition. Im engeren Sinn sei hier darunter die Ausführung von Simulationsmodellen über das Internet verstanden, wobei grundsätzlich unterschieden werden kann zwischen:

1. Server-seitiger Ausführung des Simulationsmodells entweder auf dem Web-Server selbst oder einem dahinter liegenden Anwendungsserver.
2. Client-seitiger Ausführung des Simulationsmodells als Applet innerhalb eines Web-Browsers.

Die diesen beiden Ausführungsmöglichkeiten zugrunde liegenden Client/Server-Architekturen werden im Detail in Abschnitt 3.3.1 vorgestellt.

Eine Simulation gilt im weiteren Sinne als Web-basiert, wenn in ihrem Lebenszyklus (Planung, Konzeption, Implementierung, Überprüfung, Einsatz) das World Wide Web und damit eng verbundene Technologien eine wesentliche Rolle spielen. Abb. 3.1.1 gibt einen groben Überblick über die wichtigsten Web-Technologien und deren Anwendungsmöglichkeiten in der Modellierung und Simulation.



*Abb. 3.1.1: Web-Technologien und ihre Anwendungsmöglichkeiten in der computergestützten Modellierung und Simulation*

Die Vorteile Web-basierter Simulationen ergeben sich aus den allgemeinen Vorteilen der Arbeit im Web, also aus

- orts-, zeit- und plattformunabhängigem Zugriff
- hoher Integrationsfähigkeit durch Verwendung weltweit akzeptierter Standards
- minimalem Wartungs- und Installationsaufwand beim Benutzer
- einfachen Vertriebs- und Wartungsmöglichkeiten für den Anbieter

Neben diesen Vorteilen gibt es jedoch auch einige Nachteile. So sollte beispielsweise bei der Entwicklung Web-basierter Simulationen berücksichtigt werden, dass sich die zugrunde liegenden Web-Technologien immer noch rasant weiterentwickeln. Auch bei der Benutzung Web-basierter Simulationen kann es durch Schwankungen der Dienstgüte im Netz (Latenzzeit, Bandbreite, Verfügbarkeit) zu Unannehmlichkeiten kommen.

In den nun folgenden Abschnitten werden die in Abb. 3.1.1 dargestellten Anwendungsmöglichkeiten von Web-Technologien im Bereich der computergestützten Modellierung und Simulation im Detail erläutert.

## 3.2 Standardisierte Modellaustauschformate

Heute etabliert sich zunehmend die eXtensible Markup Language (XML) als Universalformat zur Speicherung von Dokumenten und für den Datenaustausch zwischen heterogenen Anwendungen. Als Metasprache, die auf der Grundlage des ISO-Standards SGML (Standard Generalized Markup Language) entwickelt wurde, eignet sich XML hervorragend zur Spezifikation von spezifischen Auszeichnungssprachen für den Datenaustausch in speziellen Anwendungsgebieten. Beispiele für solche Auszeichnungssprachen aus unterschiedlichen Wissenschaftsbereichen sind MathML (Mathematical Markup Language), CML (Chemical Markup Language) oder AIML (Astronomical Instrument Markup Language). Neben den technischen Möglichkeiten machen vor allem folgende Eigenschaften die Verwendung von XML attraktiv:

- XML ist als etablierter W3C<sup>1</sup>-Standard firmen- und plattformunabhängig.
- Zu XML existiert eine Vielzahl an Unterstützungswerkzeugen (Parser, Transformatoren, etc.) für alle gängigen Programmiersprachen.
- XML ist ein textbasiertes Beschreibungsformat, das nicht nur von Maschinen, sondern auch von Menschen lesbar ist.

All diese Eigenschaften haben dazu beigetragen, dass XML heute als ideales technisches Beschreibungsformat für den Import und Export von Daten zwischen verschiedenen Applikationen gilt, die unabhängig voneinander ihre eigene, für ihre Belange optimierte Datenhaltung betreiben können. Auch zur Beschreibung von Daten, die bei der Arbeit mit Modellierungswerkzeugen erzeugt und ausgetauscht werden, ist XML aus heutiger Sicht wohl das geeignetste Format. Bei Modellierungswerkzeugen ist vor allem ein allgemein akzeptierter Beschreibungsstandard für die unterstützte Modellierungstechnik von großer Bedeutung, da davon entscheidend abhängt, wie einfach sich der Austausch von Modellen mit anderen Modellierungswerkzeugen, die auf derselben Modellierungstechnik basieren, gestaltet. Ein solcher Austausch ist deshalb sehr wichtig, weil dadurch der kombinierte Einsatz verschiedener Modellierungswerkzeuge ermöglicht wird. Auf solche Werkzeugkombinationen sollte stets hingearbeitet werden, da es wohl niemals das universelle Modellierungswerkzeug geben wird, das allen anderen Werkzeugen in allen Belangen überlegen ist. Weitere Details zu XML-basierten Modellaustauschformaten finden sich in Abschnitt 4.2.3.5.

## 3.3 Online-Simulation und -Animation

Da sich das World Wide Web zu einem der wichtigsten Kommunikationsmedien unserer Zeit entwickelt hat, sind Softwarehersteller in zunehmendem Maße gezwungen, ihre Produkte Webtauglich zu machen, um konkurrenzfähig zu bleiben. Dies trifft auch auf die Hersteller von Simulationssoftware zu. Im Folgenden werden die grundsätzlichen Möglichkeiten aufgezeigt, um Simulationen und -Animationen über das Web zugreifbar zu machen.

---

<sup>1</sup> World Wide Web Consortium

### 3.3.1 Architekturkonzepte für Simulationsanwendungen im Web

Geht man von dem heute üblichen Client/Server-Modell aus, gibt es grundsätzlich zwei Möglichkeiten, um Computeranwendungen im World Wide Web verfügbar zu machen. Die beiden Möglichkeiten unterscheiden sich vor allem darin, wo das Anwendungsprogramm zur Ausführung gebracht wird. Dies kann entweder auf dem Rechner des aufrufenden Client geschehen oder auf einem dahinter liegenden Server. Beide Möglichkeiten werden im Folgenden vorgestellt und bezüglich ihrer Vor- und Nachteile diskutiert.

#### 3.3.1.1 Client-seitige Ausführung

Bei der Client-seitigen Ausführung läuft die Anwendung in einem Java-fähigen Browser ab, der wiederum auf einem beliebigen Rechner mit Internetanschluss installiert sein kann. Die Anwendung liegt in Form eines Java-Applets vor, das in eine HTML-Seite eingebunden ist. Wird die Seite aufgerufen, wird das Applet über das Internet vom Web-Server zum Client übertragen und dort von der JVM (Java Virtual Machine) des Browsers ausgeführt.

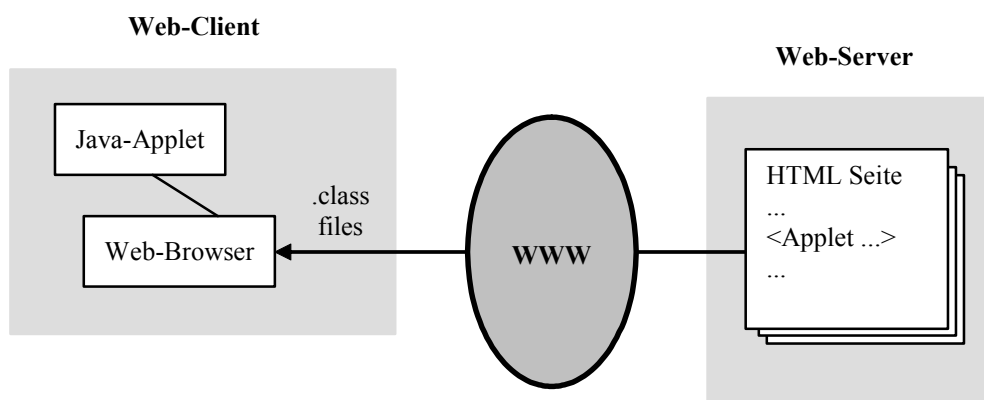


Abb. 3.3.1.1.1: Client-seitige Programmausführung unter Verwendung eines Java-Applets

Die Vorteile dieser in Abb. 3.3.1.1.1 dargestellten Realisierungsalternative liegen darin, dass die Anwendung orts-, zeit- und plattformunabhängig sowie ohne jeglichen Installationsaufwand ausgeführt werden kann. Die Client-seitige Ausführung hat jedoch auch einige schwerwiegende Nachteile. Diese bestehen vor allem darin, dass

- die Anwendung vollständig in Java implementiert sein muss,
- rechenintensive Anwendungen entsprechend leistungsfähige Client-Rechner voraussetzen,
- die gesamte Anwendung über das Internet übertragen werden muss, was unter Umständen sehr zeitaufwändig sein kann.

Weitere Nachteile ergeben sich aus der stetigen Weiterentwicklung sowohl von Java als auch den Browser-Technologien. Dies führt dazu, dass auf ein und demselben Rechner die Ausführungsgeschwindigkeit und Darstellung einer Anwendung je nach verwendeter Browserart<sup>2</sup> und -version sehr unterschiedlich ausfallen kann, obwohl die Funktionalität von Java-Applets von ihrer Konzeption her für alle Web-Browser und Betriebssysteme die Gleiche sein sollte. Des Weiteren haben Java-Applets aus Sicherheitsgründen nur beschränkten Zugriff auf lokale Ressourcen. Um berechnete Resultate abspeichern zu können, muss der Benutzer die dazu notwendigen Zugriffsmöglichkeiten der Anwendung explizit einräumen.

Aufgrund der oben aufgezählten Vor- und Nachteile eignet sich die Client-seitige Programmausführung mittels Java-Applets vor allem für Simulationsanwendungen mit kleinem bis mittelgroßem Rechen- und Speicherplatzbedarf. Simulationen solcher Art werden zunehmend im universitären Bereich zur Unterstützung der Lehre oder zur Präsentation von Forschungsergebnissen eingesetzt. Für speicher- und rechenintensive Simulationsanwendungen ist eine Client-seitige Programmausführung jedoch ungeeignet, weil Client-Rechner meist nicht die dazu erforderliche Leistungsfähigkeit aufweisen. Für solche Simulationsanwendungen empfiehlt sich die Server-seitige Ausführung, die im folgenden Abschnitt näher erläutert wird.

### 3.3.1.2 Server-seitige Ausführung

Bei der Server-seitigen Ausführung initiiert der Benutzer vom Client-Rechner aus die Ausführung einer Anwendung auf dem Web-Server oder einem dahinter liegenden Rechner. Die traditionelle Technik, um Server-seitige Programmausführungen anzustoßen, sind CGI-Programme, die jedoch heute zunehmend von Java-basierten Technologien (Java-Servlets, Java Server Pages, etc.) abgelöst werden. Um Server-seitige Programmausführungen zu initiieren, stellt HTML so genannte Formulare zur Verfügung. In diese werden die Eingabedaten für das auszuführende Anwendungsprogramm eingetragen. Ist die Formularseite ausgefüllt, kann sie an den Web-Server weitergeleitet werden, der das dazugehörige CGI-Programm bzw. Servlet aktiviert. Dieses wiederum leitet die Eingabedaten an das Anwendungsprogramm weiter, das, um den Web-Server nicht zu überlasten, üblicherweise auf einem dahinter liegenden (Groß-)Rechner ausgeführt wird. Nachdem das Simulationsprogramm abgearbeitet wurde, werden die Simulationsergebnisse an das auf dem Web-Server laufende CGI-Programm bzw. Servlet übermittelt, das die HTML-Antwortseite erzeugt und diese an den Client weiterleitet. Die Abläufe bei der Server-seitigen Programmausführung sind noch einmal in Abb. 3.3.1.2.1 zusammengefasst.

Die Vorteile der Server-seitigen Programmausführung liegen vor allem darin, dass

- auf Clientseite keine leistungsstarken Rechner vorausgesetzt werden, da hier lediglich ein moderner Web-Browser ausgeführt werden muss,
- keine (Re-)Implementierung des auszuführenden Anwendungsprogramms in Java notwendig ist, da dieses meist in unveränderter Form und auf der ihm angestammten Plattform eingesetzt werden kann,

---

<sup>2</sup> Der Browser-Markt wird heute vom Microsoft Internet Explorer und Netscape Navigator dominiert. Alternativen dazu stellen die beiden Browser Opera und Mozilla dar, die jedoch bislang keine weite Verbreitung gefunden haben.

- die Netzbelastung in der Regel sehr gering ausfällt.

Aufgrund der oben genannten Eigenschaften eignet sich die Server-seitige Programmausführung vor allem für Simulationsanwendungen mit hohem Ressourcenbedarf, die für leistungsstarke Spezialplattformen (Groß- und Parallelrechner) ausgelegt sind.

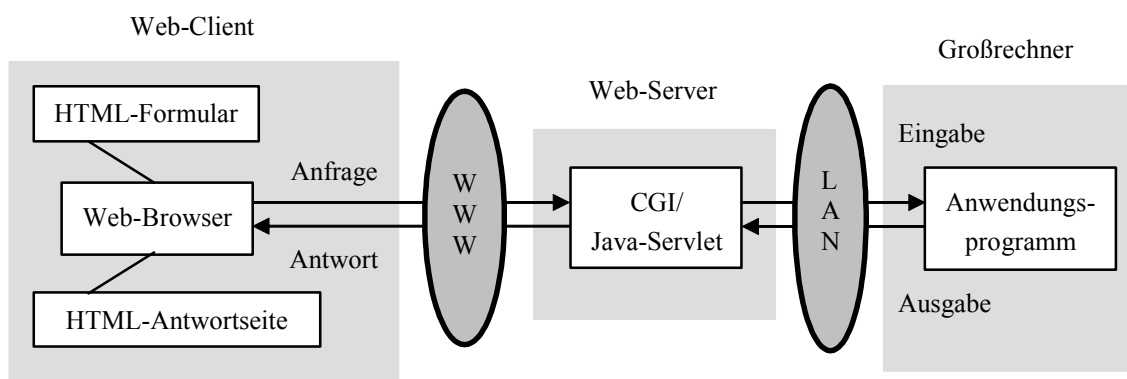


Abb. 3.3.1.2.1: Server-seitige Programmausführung mit CGI-Programmen bzw. Java-Servlets

Neben den oben aufgezählten Vorteilen haben Server-seitige Lösungen jedoch auch einige Nachteile. Die bei Simulationsanwendungen heute häufig geforderte zeitgleiche Animation der simulierten Vorgänge auf Clientseite ist i.a. schwierig zu realisieren und auch nur dann möglich, falls das Übertragungsnetzwerk eine entsprechende Bandbreite zur Verfügung stellt. Darüber hinaus sind heterogene drei- und mehrschichtige Server-seitige Lösungen meist sehr aufwändig zu entwickeln, zu installieren und zu testen. Große Schwierigkeiten bereiten in diesem Zusammenhang vor allem die heute noch häufig eingesetzten CGI-Programme, die jedoch zunehmend von leistungsstärkeren Java-basierten Technologien wie beispielsweise Java-Servlets ersetzt werden.

### 3.4 Java-basierte Simulationspakete

Kurz nach dem Aufkommen der Programmiersprache Java entstanden bereits die ersten Java-basierten Simulationspakete [BuSt96]. Heute existiert eine Vielzahl solcher Pakete, was auf die zahlreichen Vorzüge von Java und insbesondere auf das Applet-Konzept zurückzuführen ist, das die Möglichkeit eröffnet, Simulationen orts-, zeit- und plattformunabhängig innerhalb von Web-Browsern auszuführen. In Tabelle 3.4.1 sind einige weit verbreitete Java-basierte Simulationspakete aufgeführt. Die meisten der dort dargestellten Simulationspakete entstammen dem universitären Bereich. Es gibt aber auch bereits einige kommerzielle Simulationspakete wie z.B. Silk, das von der Firma ThreadTec vertrieben wird.

Im Gegensatz zu Simulationssprachen wie beispielsweise GPSS, Siman und SLX, die höhere speziell unter dem Aspekt des Einsatzes zur Simulation entwickelte Programmiersprachen darstellen, versteht man unter Simulationspaketen Aufsätze auf weit verbreitete, nicht simulationsspezifische Programmiersprachen, die den speziellen Anforderungen der Simulation entge-



genkommen. Simulationspakete haben gegenüber Simulationssprachen folgende Vorteile: Ihre Entwicklung gestaltet sich i.a. einfacher, da man sich auf die simulationsspezifischen Erweiterungen beschränken kann. Darüber hinaus braucht der Benutzer keine weitere Programmiersprache zu erlernen.

Bezeichnung	Herkunft	Literatur
Simjava	Institute for Computing Systems Architecture, Division of Informatics, University of Edinburgh, <a href="http://www.dcs.ed.ac.uk/home/hase/simjava/">http://www.dcs.ed.ac.uk/home/hase/simjava/</a>	McHo96, KrHM97, PaMG97, HoMc98
Silk	ThreadTec Incorporation, St. Louis, USA, <a href="http://www.threadtec.com/">http://www.threadtec.com/</a>	HeKi97, HeKi98, HeKK98
JavaSim	Department of Computing Science, University of Newcastle upon Tyne, <a href="http://javasim.ncl.ac.uk/">http://javasim.ncl.ac.uk/</a>	Litt99
JSIM	University of Georgia, <a href="http://chief.cs.uga.edu/~jam/jsim/">http://chief.cs.uga.edu/~jam/jsim/</a>	Nair97, Zhan97, MiSX00
DEVJSJAVA	University of Arizona, <a href="http://www-ais.ece.arizona.edu/SOFTWARE/software.html">http://www-ais.ece.arizona.edu/SOFTWARE/software.html</a>	SaZe98, ZeHS97a, ZeHS97b
DESMO-J	Universität Hamburg, <a href="http://asi-www.informatik.uni-hamburg.de/themen/sim/forschung/Simulation/Desmo-J/">http://asi-www.informatik.uni-hamburg.de/themen/sim/forschung/Simulation/Desmo-J/</a>	Lech99, Claa99, PaLC00

*Tabelle 3.4.1: Java-basierte Simulationspakete*

Bei der Verwendung von Java als Wirtssprache, lassen sich deren zahlreiche vorteilhafte Eigenschaften ausnutzen. Dazu zählen:

- Objektorientierung
- Plattformunabhängigkeit und Portabilität
- Applet-Konzept (mobiler Code)
- Unterstützung von Nebenläufigkeit (häufig ist jedem Simulationsobjekt ein eigener Thread zugeordnet, in dem es ausgeführt wird)
- Sicherheitskonzept
- umfangreiche Klassenbibliothek
- weite Verbreitung
- große syntaktische Ähnlichkeit mit C/C++
- enge Verbindung zu anderen Schlüsseltechnologien

Neben seinen vielen Vorzügen hat Java jedoch auch einen schwerwiegenden Nachteil. Als interpretierte Programmiersprache weist Java deutlich langsamere Ausführungszeiten auf als kompilierte Sprachen wie beispielsweise C oder C++. Zur Realisierung von sehr rechenintensiven Simulationen, die darauf abzielen, die spezifischen Fähigkeiten einer bestimmten Hardware-Plattform optimal auszunutzen, sind Java-basierte Simulationspakete daher nicht geeig-

net. Eine Auflistung der möglichen Anwendungsfelder von Java-basierten Simulationspaketen findet sich in Abschnitt 3.4.2.

### 3.4.1 Grundsätzlicher Aufbau

Java-basierte Simulationspakete setzen unmittelbar auf der Programmiersprache Java auf und stellen häufig Reimplementierungen bewährter C++-basierter Simulationspakete dar. Zur Grundausstattung eines Java-basierten Simulationspakets gehören Klassenpakete zur Erstellung ereignisorientierter Simulationen, zur Animation der simulierten Abläufe sowie zur Aufbereitung und Ausgabe von Simulationsergebnissen. Eine eigene Entwicklungsumgebung wird i.a. nicht mitgeliefert, da auf komfortable Java-Entwicklungsumgebungen (JBuilder, Symantec Visual Cafe, etc.) zurückgegriffen werden kann. Diese Entwicklungsumgebungen unterstützen graphische Modellierung und Animationen, indem entsprechende JavaBeans-Komponenten eingebunden werden können. Unabhängig von der gewählten HW-Plattform gestaltet sich der Gebrauch eines Java-basierten Simulationspakets wie folgt:

1. Installation von Java und einer Java-Entwicklungsumgebung.
2. Installation der Klassenpakete des Java-basierten Simulators und gegebenenfalls Einbinden der JavaBeans-Komponenten in die Entwicklungsumgebung.
3. Erstellung eines Simulationsmodells mit den zur Verfügung gestellten Klassen und Komponenten, wobei natürlich auch auf allgemeine Java Klassen zurückgegriffen werden kann<sup>3</sup>.
4. Übersetzung der erstellten Simulation als Java-Anwendung oder als Java-Applet.
5. Ausführung der Simulation als Anwendung oder als Applet.

### 3.4.2 Anwendungsfelder

Wird eine Anwendung in Java implementiert, muss zunächst entschieden werden, ob sie als Java-Applet oder als Java-Applikation realisiert werden soll. Java-Applets werden automatisch über das Internet auf den Rechner des Anwenders geladen und durch die Java Virtual Machine des dort verwendeten Web-Browsers ausgeführt. Aus Sicherheitsgründen ist jedoch der Zugriff auf lokale Ressourcen des Client-Rechners stark eingeschränkt<sup>4</sup>. In dieser sicheren Umgebung, der so genannten Sandbox, ist beispielsweise kein Lesen und Schreiben von Dateien auf dem Client-Rechner möglich, es können keine weiteren Programme auf dem Client gestartet werden und es kann auch keine Netzwerkverbindung zu anderen Rechnern aufgebaut werden, außer zu dem Rechner, von dem das Applet stammt. Im Gegensatz zu Java-Applets stellen Java-Applikationen vollwertige Anwendungen dar, für die diese Beschränkungen nicht gelten.

---

<sup>3</sup> Dem Entwickler steht also die ganze Sprachmächtigkeit von Java zur Verfügung.

<sup>4</sup> In den neueren Versionen von Java ist es durch eine Reihe zusätzlicher Maßnahmen möglich, diese Sperren zu umgehen und dem Applet vollen Ressourcenzugriff zu erlauben.

Wie bereits in Abschnitt 3.3.1.1 erwähnt, sollten nur kleinere bis mittelgroße Simulationen und Animationen als Java-Applet realisiert werden, da bei Client-Rechnern in der Regel keine hohe Rechenkapazität vorausgesetzt werden kann. Geeignet für eine Realisierung als Java-Applet sind vor allem Simulationen, die darauf abzielen, bestimmte Sachverhalte möglichst anschaulich zu demonstrieren und, aufgrund der bei Java-Applets geltenden Beschränkungen, keinen Zugriff auf lokale Ressourcen benötigen. Simulationen dieser Art findet man heute vor allem im Bereich

- der Lehre, um komplexe Lerninhalte zu vermitteln
- der Forschung, um aktuelle Forschungsergebnisse zu präsentieren
- des Marketings, um bestimmte Eigenschaften eines Produktes hervorzuheben

Bei der Realisierung einer Simulation als Java-Applikation, entfallen die bei Java-Applets geltenden Einschränkungen, wobei jedoch nach wie vor der Geschwindigkeitsnachteil gegenüber kompilierten Programmiersprachen zu beachten ist. Bei den neueren Java-Versionen, die immer ausgereiftere JIT<sup>5</sup>-Compiler zur Verfügung stellen, nimmt dieser Geschwindigkeitsunterschied aber mehr und mehr ab und es ist eine Annäherung an die Ausführungsgeschwindigkeit von C++ zu beobachten. Bei der heute zur Verfügung stehenden Hardware ist es daher durchaus vertretbar, auch größere rechenintensive Simulationen in Java zu realisieren. Gerade im Hinblick auf verteilte komponentenorientierte Simulationen, bei denen sich die Gesamtlast auf mehrere Rechner verteilen lässt, werden die noch vorhandenen Geschwindigkeitsnachteile bei weitem vom Vorteil der plattformübergreifenden Nutzbarkeit Java-basierter Simulationen übertroffen. In einem solchen verteilten Umfeld eignet sich Java nicht nur zur Realisierung von Simulationskomponenten, sondern aufgrund seines leistungsfähigen Graphik-APIs vor allem auch zur Realisierung von Komponenten zur Animation simulierter Abläufe.

### 3.5 Web-basierte Simulationsumgebungen

Simulationswerkzeuge, die auf bewährten Simulationssprachen und/oder -paketen aufsetzen, werden auch als Simulationsumgebungen bezeichnet. Bei den heute verfügbaren Simulationsumgebungen, deren Aufgabe im Wesentlichen darin besteht, die in Abb. 2.1.1 dargestellten klassischen Arbeitsschritte einer Simulationsstudie zu unterstützen, kann zwischen offenen und geschlossenen Systemen unterschieden werden. In geschlossenen Simulationsumgebungen sind die Werkzeuge, mit denen Modelle entwickelt und Simulationsstudien durchgeführt werden, fest vorgegeben. Bei offenen Umgebungen ist es dagegen möglich, Softwarewerkzeuge unter Nutzung eines definierten Protokolls je nach Erfordernis hinzuzufügen bzw. zu entfernen und Werkzeuge innerhalb der Umgebung für jedes spezifische Anwendungsgebiet zu modifizieren [StCe94]. Neben Offenheit wird heute von Simulationsumgebungen vor allem die Nutzbarkeit der zur Verfügung gestellten Funktionalität über das World Wide Web gefordert. Darüber hinaus sollten sich auf einfache Weise Konzepte zum ortsungebundenen kooperativen Arbeiten an und mit Simulationsmodellen umsetzen lassen. Beispiele für solche innovativen Web-basierten Simulationsumgebungen finden sich in [DoGL96], [KaZY97], [Seib97],

---

<sup>5</sup> Just-in-Time

[LDRS97], [Nair97], [IaDA98] und [SeSB98]. Die im Rahmen dieser Arbeit entwickelte offene Komponentenarchitektur für Web-basierte Modellierungswerkzeuge wird in Abschnitt 4.2 beschrieben.

## 3.6 3D-Visualisierung

Wie bereits in Abschnitt 2.4.6 beschrieben, haben sich Grafikformate für das Web in den letzten Jahren rasant weiterentwickelt. Nachdem sich 2D-Formate, wie Shockwave Flash (SWF) von Macromedia und Scalable Vector Graphics (SVG) vom W3C bereits seit längerer Zeit im Web etabliert haben, wurde nun auch mit Extensible 3D (X3D) ein akzeptabler Nachfolger für das bereits etwas betagte Virtual Reality Modeling Language (VRML) Format verabschiedet. Die enge Zusammenarbeit des Web3D Konsortiums und der Moving Picture Experts Group (MPEG) führte unter anderem dazu, dass das X3D-Format heute den Kern der MPEG-4 3D-Integration bildet. Parallel zu dem auf XML basierenden X3D-Format wurde das Binary Format for Scenes (BIFS) als eine weitere 3D-Format Alternative in MPEG-4 integriert. Nachfolger des MPEG-4 Formats (MPEG-7 und MPEG-21) sind bereits in Planung. Schließlich sollte noch auf Java3D hingewiesen werden, das als eine Java Erweiterung ebenfalls die Entwicklung komplexer 3D-Konstrukte und visueller Umgebungen auf einer hohen Abstraktionsstufe ermöglicht.

Die heute zur Verfügung stehenden Web3D-Technologien sind nicht nur für die Entwickler von Webseiten und virtuellen Welten interessant, sondern werden auch zunehmend im Bereich der Modellierung und Simulation zur dreidimensionalen Darstellung simulierter Prozesse eingesetzt. Dabei lassen sich grundsätzlich folgende Darstellungsformen unterscheiden:

- **Prozessvisualisierung:** Hier werden simulierte Prozesse in Abhängigkeit ihrer Zeit und/oder Ortskoordinaten als Kurven auf entsprechenden Ausgabegeräten (Drucker, Monitor) visualisiert. Dies kann entweder Offline (nach der Simulation) oder Online (während der Simulation) geschehen, wobei letzteres für interaktive Simulationen unabdingbare Voraussetzung ist.
- **Prozessanimation:** Neben der Prozessdarstellung in Kurvenform werden auch häufig Animationen der simulierten Objekte selbst und ihrer Bewegungs- und oder Verformungsabläufe verlangt. Diese Form der Prozessdarstellung mit Mitteln der Graphikanimation kann ebenfalls entweder Offline oder Online (in Echtzeit) vorgenommen werden.
- **Virtuelle Realität:** Werden bei interaktiver Echtzeit-Prozessanimation zusätzlich zum visuellen auch andere sensorische Kanäle (Audition, Kienästhetik, etc.) des Menschen in die Darbietung der Simulation einbezogen, so spricht man von virtueller Realität.

Bei der klassischen Visualisierung und/oder Animation simulierter Prozesse ist ein Simulationsmodell direkt mit einem entsprechenden Darstellungsmodell verbunden, wobei je nach der Art der Synchronisation der lokalen Zeiten in den beiden Modellen zwischen einer Offline- und Online-Kopplung unterschieden wird. Die verteilte komponentenorientierte Simulation ermöglicht prinzipiell die strikte Trennung von Simulations- und Darstellungsmodell und damit schließlich auch deren Ausführung auf unterschiedlichen Plattformen. Beispiele für solche

Konstellationen, deren primäres Ziel es ist, die Darstellungsfunktionalität auf ein höheres Abstraktionsniveau zu heben, finden sich in [RKSD98] und [JLKS00].



## *Kapitel 4*

# **Komponenten-Technologien in der Modellierung und Simulation**

---

Im Bereich der Modellierung und Simulation werden Komponenten-Technologien heute in zunehmendem Maße zum strukturierten Aufbau verteilter Simulationen eingesetzt. Speziell zu diesem Zweck wurde vom US-Verteidigungsministerium mit der High Level Architecture (HLA) ein auf die spezifischen Probleme der verteilten Simulation zugeschnittenes Komponentenmodell entwickelt, das im September 2000 von der IEEE als internationaler Standard (IEEE 1516) verabschiedet wurde. Neben dem Aufbau verteilter Simulationen bieten sich Komponenten-Technologien jedoch auch zur Erstellung von Modellierungswerkzeugen an, deren Komplexität in den letzten Jahren ebenfalls stark zugenommen hat. In diesem Bereich werden vor allem universelle Komponentenstandards wie beispielsweise (Enterprise) JavaBeans, Corba Component Model oder (D)COM/COM+ eingesetzt.

Der nun folgende Abschnitt 4.1 gibt einen Überblick über die High Level Architecture. Im Anschluss daran wird in Abschnitt 4.2 ausführlich auf den komponentenorientierten Aufbau von Modellierungswerkzeugen eingegangen.

## **4.1 Aufbau verteilter komponentenorientierter Simulationen mit der High Level Architecture**

Heute existierende Simulationsanwendungen stellen in der überwiegenden Mehrzahl monolithische Softwaresysteme dar, die auf genau einer Modellierungstechnik basieren und an eine bestimmte Hard- und Software-Plattform gebunden sind. Ab einer gewissen Größe werden solche Simulationen extrem unübersichtlich und sind entsprechend schwierig zu warten. Erweiterungen oder gar Übertragungen auf andere Problemstellungen sind wegen des damit verbundenen Aufwands fast gänzlich ausgeschlossen. Um die vielfältigen Schwierigkeiten, die ein monolithischer Modellaufbau bei Änderungen der Funktionalität oder Konnektivität mit sich bringt, zu umgehen, hat das US-amerikanische Verteidigungsministerium im September 1996 mit der High Level Architecture (HLA) einen Standard zur komponentenorientierten Erstellung verteilter Simulationsmodelle geschaffen. Von herkömmlichen Komponentenstandards unterscheidet sich die HLA vor allem dadurch, dass spezielle zum Aufbau verteilter Simulationsanwendungen erforderliche Infrastrukturdienste angeboten werden. Ein in diesem Zusammenhang sehr wichtiger Dienst ist beispielsweise das in der HLA integrierte Zeitmanagement, das die Synchronisation der an einer verteilten Simulation beteiligten Komponenten zur Laufzeit erlaubt.

Im Gegensatz zu anderen Technologien und Konzepten aus der verteilten Simulation, die vor allem eine bestmögliche Parallelisierung auf Modellebene als Zielsetzung haben (siehe dazu auch Abschnitt 2.2.2.3), besteht das Hauptanliegen der HLA nicht in der Steigerung der Ausführungsgeschwindigkeit. Es liegt vielmehr darin, den auch im Simulationsbereich immer lauter werdenden Forderungen nach Wiederverwendbarkeit und Interoperabilität gerecht zu werden. Dazu werden von der HLA eine Reihe neuer Ansätze und Mechanismen bereitgestellt, die es erlauben, Computermodelle aus modularen Teilmodellen aufzubauen. Diese Teilmodelle können auf beliebigen heterogenen Simulationssystemen implementiert sein, sofern diese eine HLA-Anbindung besitzen. Die Teilnehmer eines gemeinsamen Simulationslaufs werden in der HLA-Terminologie als Föderierte (Federates) bezeichnet. Der Zusammenschluss mehrerer Föderierter zu einer Gesamtsimulation stellt eine so genannte Föderation (Federation) dar. Ein großer Vorteil der HLA besteht in der Offenheit für verschiedene Arten von Föderierten. Dabei muss es sich nicht unbedingt um Simulationen handeln, sondern Föderierte können auch andere Softwarebausteine (Datenbanken, etc.) oder Systeme der realen Außenwelt (Hardware aller Art) darstellen.

### **4.1.1 Entwicklungsgeschichte der HLA**

Die HLA wurde wie bereits erwähnt vom US-Verteidigungsministerium entwickelt, das seit jeher den weltweit bedeutendsten Auftraggeber für Computersimulationen darstellt. Für den Militärbereich sind Simulationen deshalb so attraktiv, weil sich damit Waffensysteme und Strategiekonzepte ohne deren konkrete Realisierung in ihrer Wirkungsweise erproben lassen. Anfang der 90er Jahre ist die Zahl vorhandener Simulationen schließlich derart angewachsen, dass der Überblick über diese in der Regel sehr teuren Systeme verloren zu gehen drohte. Außerdem stellte man fest, dass die existierenden Simulationen zum größten Teil auf genau eine Anwendungsproblematik zugeschnitten waren und sich aufgrund ihres monolithischen Aufbaus nur schwer auf andere Problemstellungen übertragen ließen. Das Ende des kalten Krieges und die damit verbundenen drastischen Einsparungen im Militärhaushalt zwangen schließlich zu einem radikalen Umdenken hinsichtlich kostengünstiger, wiederverwendbarer und interoperabler Simulationssoftware. Das Ergebnis dieses Umdenkens stellt die HLA dar, die nach längerer Entwicklungszeit vom US Department of Defense (DoD) im September 1996 zum verbindlichen Standard für alle künftigen Neuentwicklungen von Simulationssoftware im militärischen Bereich erklärt wurde. Gleichzeitig wurde die HLA zur freien Nutzung für den zivilen Bereich offen gelegt.

Die HLA wurde vom DoD aufbauend auf den Erfahrungen mit der DIS-Technologie (Distributed Interactive Simulation) und dem ALSP-Protokoll (Aggregate Level Simulation Protocol) entwickelt. DIS stellt ein Synchronisationsprotokoll dar, das Echtzeitsimulationen ohne Kausalitätserhaltung miteinander verknüpft. Es wird hauptsächlich zur Vernetzung militärischer Trainingssimulatoren eingesetzt, die gemeinsame Übungen ermöglichen sollen. Die Kommunikation erfolgt nach einem Broadcast-Ansatz, wobei zur Minimierung der Netzlast Dead Reckoning Mechanismen zum Einsatz kommen, welche eine Extrapolation z.B. von Objektbewegungen ermöglichen. Durch die Einbettung der Nachrichtenstruktur in die Architektur ist DIS jedoch relativ unflexibel und ineffektiv.

Das Aggregate Level Simulation Protocol (ALSP) stellt eine Weiterentwicklung von DIS dar, um auch die umfangreiche Klasse der ereignisgesteuerten Simulatoren zu erschließen. ALSP



ist ein militärischer Standard, der Nachrichtenstrings austauscht und bereits strukturelle Ähnlichkeiten zur HLA aufweist. ALSP setzt im Gegensatz zu DIS ein zuverlässiges Kommunikationsmedium und strenge Kausalität voraus. Hierzu wird das Chandy/Misra/Bryant (CMB) Null-Message-Protokoll eingesetzt. Anwendung findet ALSP bei der Vernetzung militärischer konstruktiver Simulationen zum Führungskräfte-Training, bei denen simulierte Kräfte in einem simulierten System agieren. ALSP soll ebenso wie DIS in naher Zukunft von der HLA abgelöst werden.

## 4.1.2 Architekturüberblick

Abb. 4.1.2.1 gibt einen Überblick über die wesentlichen Bestandteile der HLA, deren Grundidee in der strikten Trennung von Simulationsfunktionalität und Infrastruktur für die Interoperabilität besteht.

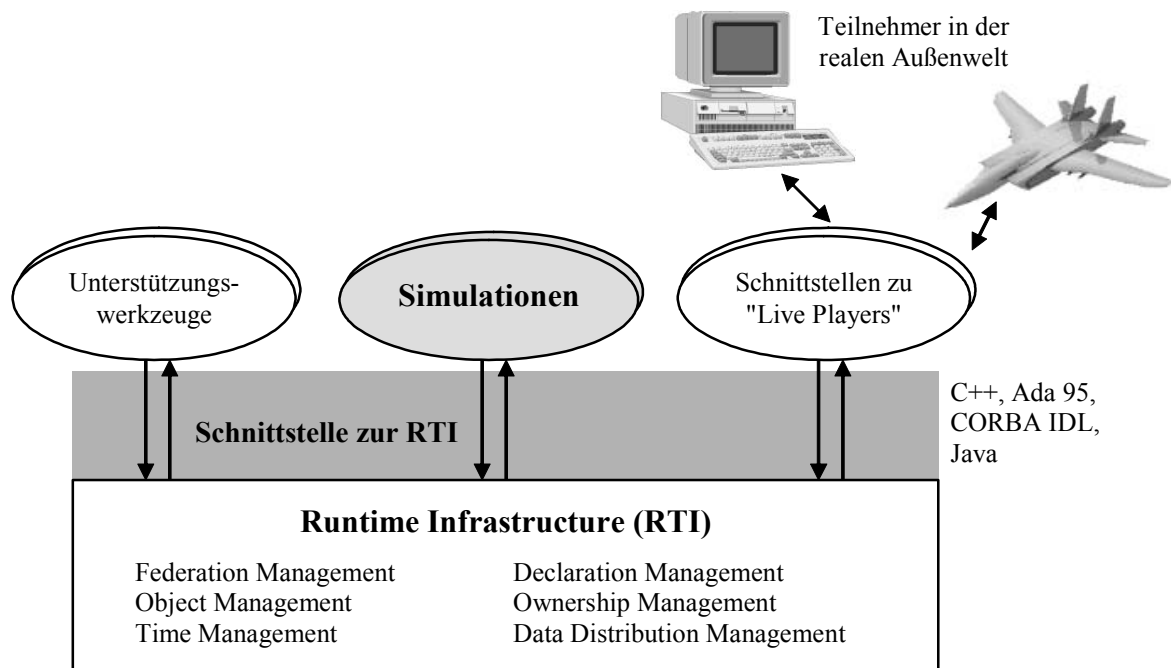


Abb. 4.1.2.1: Aufbau der High Level Architecture

Die Simulationsfunktionalität wird durch mehrere Föderierte realisiert, die im Rahmen einer Föderation unter Echtzeitbedingungen zusammenwirken. Eine Föderation kann als ein Vertrag zur Durchführung eines Simulationslaufs (Federation Execution) zwischen den an der Simulation beteiligten Föderierten angesehen werden. In diesem Vertrag sind die Objektmodelle der Föderierten (Simulation Object Model (SOM)) und der Gesamtföderation (Federation Object Model (FOM)) in einer definierten dem Object Model Template (OMT) entsprechenden Form festgelegt. Darüber hinaus legt die HLA zwingende Verhaltensregeln für Föderierte und Föderationen fest. Dieser HLA-Regelsatz, der aus insgesamt 10 Regeln besteht, beschreibt die Art und Weise, wie die HLA zu verwenden ist. Beispielsweise darf, um Interoperabilität und Platt-

formunabhängigkeit zu erreichen, die Kommunikation zwischen Föderierten ausschließlich über die Laufzeit-Infrastruktur RTI (Runtime Infrastructure) erfolgen. Um mit dieser in Kontakt zu treten, müssen die Föderierten eine einheitliche Schnittstelle (HLA Interface Specification) aufweisen. Durch die HLA-Schnittstellenspezifikation werden die von der RTI zur Verfügung gestellten Kommunikations- und Koordinationsdienste sowie die von den Föderierten zu erbringenden Callback-Funktionen beschrieben.

In den folgenden Abschnitten werden die wichtigsten Bestandteile der HLA kurz beschrieben. Eine ausführliche Beschreibung der HLA ist in [KuWD99] zu finden. Informationen über aktuelle Entwicklungen werden vom US-Verteidigungsministerium über das WWW unter [HLA2] zur Verfügung gestellt.

### 4.1.3 Der HLA-Regelsatz

Die HLA-Regeln definieren, wie sich Föderationen und die daran beteiligten Föderierten zu verhalten haben. Die Regeln sind in zwei Gruppen aufgeteilt, wobei 5 Regeln die Föderation betreffen, die übrigen fünf Regeln die einer Föderation angehörenden Föderierten. Im Folgenden sind die Regeln in vereinfachter Form wiedergegeben:

#### **Regeln für Föderationen:**

1. Föderationen müssen ein FOM (Federation Object Model) besitzen, das gemäß dem OMT (Object Model Template) aufgebaut ist.
2. Die Werte sämtlicher simulationsbezogener Objektinstanzen müssen in den Föderierten gehalten werden und nicht in der Laufzeit-Infrastruktur RTI.
3. Der Austausch von Daten aus dem FOM zwischen den Föderierten darf nur über die RTI erfolgen.
4. Föderierte müssen sich bei Interaktion mit der RTI strikt an die HLA Schnittstellenspezifikation halten.
5. Ein Attribut einer Objektinstanz darf von höchstens einem Föderierten verwaltet werden.

#### **Regeln für Föderierte:**

6. Föderierte müssen ein SOM (Simulation Object Model) besitzen, das gemäß dem OMT (Object Model Template) aufgebaut ist.
7. Föderierte müssen in der Lage sein, gemäß ihrer SOM Spezifikation Objektattribute zu aktualisieren und/oder zu reflektieren sowie Interaktionen zu versenden und/oder zu empfangen.

8. Föderierte müssen in der Lage sein, gemäß ihrer SOM Spezifikation die Verwaltung von Objektattributen dynamisch während der Föderationsausführung abzugeben und/oder zu übernehmen.
9. Föderierte müssen in der Lage sein, gemäß ihrer SOM Spezifikation die Bedingungen zu verändern, unter denen sie Aktualisierungen von Objektattributen vornehmen.
10. Föderierte müssen in der Lage sein, ihre lokale Simulationszeit so zu verwalten, dass ein koordinierter Datenaustausch mit anderen Föderierten gemäß den Diensten des Time Management erfolgen kann.

#### 4.1.4 Die Laufzeit-Infrastruktur

Die Laufzeit-Infrastruktur RTI stellt die Implementierung der HLA-Schnittstellenspezifikation dar und stellt den an einer Föderation beteiligten Simulationen verschiedene Dienste zur Verfügung. Diese sind in insgesamt sechs funktionale Gruppen unterteilt:

- Federation Management

Aufgabe dieser Dienstgruppe ist die Koordination von Aktivitäten, die das Föderationsmanagement betreffen. Dazu werden Dienste zum Erzeugen und Beenden von Föderationen zur Verfügung gestellt. Weitere Dienste gestatten den Föderierten den dynamischen Beitritt bzw. Austritt aus einer Föderation. Darüber hinaus werden Steuermöglichkeiten für das Anhalten, Speichern und Wiederherstellen von Föderationszuständen angeboten.

- Declaration Management

Die HLA zeichnet sich durch einen impliziten Datenaustausch aus. Das bedeutet, dass Föderierte Daten des gemeinsamen Objektmodells nicht explizit durch Angabe eines Föderiertennamens an andere Simulationsteilnehmer versenden, sondern implizit über die RTI der Föderation zur Verfügung stellen. Die Dienste des Declaration Managements ermöglichen es, den Föderierten der RTI mitzuteilen, dass sie beabsichtigen, Daten zu publizieren (publish) bzw. zu abonnieren (subscribe).

- Object Management

Object Management Dienste bewerkstelligen den tatsächlichen Daten- und Nachrichtenaustausch. Ein Föderierter verwendet Dienste dieser Art, um im FOM definierte Interaktionen zu senden bzw. zu empfangen. Des Weiteren ermöglichen diese Dienste die Erzeugung von Objektinstanzen der im FOM spezifizierten Objektklassen sowie die Modifikation der dazugehörenden Objektattribute.

- Ownership Management

Föderierte müssen, um Attribute von Objektinstanzen der im FOM spezifizierten Objektklassen verändern zu können, eine entsprechende Berechtigung besitzen. Die Dienste des Ownership Managements ermöglichen es, solche Berechtigungen an Föderierte zu vergeben bzw. bereits vergebene Berechtigungen an andere Föderierte weiterzugeben.

#### - Time Management

Wie bereits in Abschnitt 2.2.2.3 beschrieben, stellt das Zeitmanagement ein zentrales Problem verteilter Simulationen dar. Die vom Zeitmanagement zur Verfügung gestellten Dienste ermöglichen

- den Föderierten die Fortschaltung ihrer logischen Zeit in Koordination mit anderen Föderierten,
- die kontrollierte Vergabe von Ereignissen mit Zeitstempel, wobei bei konservativer Synchronisation ausgeschlossen wird, dass Föderierte ein Ereignis erhalten, dessen Zeitstempel in der logischen Vergangenheit des Föderierten liegt.

Die RTI erlaubt den Föderierten sich in verschiedenen Abstufungen am Zeitmanagement zu beteiligen. Ein Föderierter kann zeitreguliert und/oder zeitregulierend sein. Zeitreguliert bedeutet, dass das Fortschalten der logischen Zeit des Föderierten von den zeitregulierenden Föderierten der Föderation kontrolliert wird.

#### - Data Distribution Management

Die Dienste des Data Distribution Management kontrollieren die Erzeuger/Konsumenten-Beziehungen zwischen Föderierten und ermöglichen es, solche Beziehungen weiter zu verfeinern. Der Routing Spaces-Ansatz gewährleistet eine effiziente Datenübermittlung, wobei im Gegensatz zum Broadcast-Ansatz von DIS (Distributed Interactive Simulation) lediglich dort, wo sich Angebot und Nachfrage von Objektinformationen überschneiden, Kommunikation stattfindet.

Weitere Informationen zur RTI sowie eine Auswahl konkreter RTI-Implementierungen werden vom US-Verteidigungsministerium im WWW unter [HLA3] zur Verfügung gestellt.

### 4.1.5 Bestehende Probleme

Neben zahlreichen Vorzügen weist die HLA jedoch auch einige Unzulänglichkeiten auf. Dabei handelt es sich vor allem um

- die (noch) unzureichende Unterstützung für die Ausführung und das Management von HLA-Föderationen.
- die (noch) nicht gewährleistete echte Plug-and Play-Interoperabilität.
- diverse Mängel in der Schnittstellenspezifikation. Bei verschiedenen Diensten<sup>1</sup> fehlt die Abstimmung mit den Diensten des Zeitmanagements. Für Simulationen, die strikte Kausalität erfordern, sind solche Dienste daher nur bedingt einsetzbar.
- die (noch) unzureichende Werkzeugunterstützung für den i.a. sehr aufwändigen Entwicklungsprozess komplexer Simulationsföderationen.

---

<sup>1</sup> beispielsweise Diensten aus der Dienstgruppe des Ownership Managements

- die (noch) geringe Anzahl HLA-konformer Modellierungswerkzeuge. Die meisten existierenden Modellierungswerkzeuge wurden lange vor der Einführung der HLA entwickelt und sind daher nicht HLA-fähig. Eine nachträgliche Adaption von bestehenden Simulatoren an die HLA ist zwar grundsätzlich möglich (siehe dazu auch [StK198] und [Stra99]), birgt jedoch auch einige Risiken in sich. Insbesondere bei fehlender oder unvollständiger Dokumentation stellt sich die Frage, ob der dazu erforderliche Aufwand angemessen ist.
- die (noch) fehlenden Modellbausteinbibliotheken für bestimmte Anwendungsbereiche.
- die (noch) unzureichende Erschließung ziviler Anwendungsfelder. Da die HLA aus dem Umfeld militärischer Trainingssimulationen hervorgegangen ist, muss die Frage beantwortet werden, ob die HLA auch unter den speziellen Randbedingungen des zivilen Simulationssektors anwendbar ist. Ergebnisse grundlegender Arbeiten zu dieser Problematik sind in [Stra01b] zu finden.
- die (noch) unzureichende Grundlagenforschung zur Kopplung von sehr heterogenen Modellen, die sich auf unterschiedlichen Abstraktionsebenen bewegen und/oder verschiedenen Modellierungsparadigmen folgen. Hier ist noch völlig offen, ob die HLA in ihrer jetzigen Form für diesen Zweck überhaupt geeignet ist.
- die (noch) bestehenden Unzulänglichkeiten der RTI. Während Corba einen transparenten Datenaustausch vornimmt, enthält sich die RTI auf Datenebene jeglicher Interpretation oder Konvertierung und sorgt damit für zusätzlichen Aufwand bei plattformübergreifenden Anwendungen.

## 4.2 Komponentenorientierter Aufbau von Modellierungswerkzeugen

In den nun folgenden Betrachtungen, die sich grundsätzlich auf alle Arten von Modellierungswerkzeugen beziehen, werden zunächst im Rahmen einer Ist-Analyse bestehender Modellierungswerkzeuge die Nachteile des heute noch weitgehend vorherrschenden monolithischen Werkzeugaufbaus herausgearbeitet. Anschließend wird eine komponentenorientierte Softwarearchitektur für Modellierungswerkzeuge vorgeschlagen und es werden die darin vorkommenden Komponenten sowie deren Beziehungen untereinander aus fachlicher Sicht beschrieben. Abschließend wird auf eine mögliche software- bzw. systemtechnische Realisierung der Facharchitektur eingegangen und es werden Möglichkeiten zur Einbindung existierender Werkzeuge bzw. Werkzeugteile aufgezeigt.

### 4.2.1 Klassische Modellierungswerkzeuge

Bei den heute etablierten Modellierungswerkzeugen handelt es sich meist um hochkomplexe monolithische Softwaresysteme, die in vielen Fällen einen langen evolutionären Entwicklungsprozess hinter sich haben. Die Softwarearchitektur, die man üblicherweise bei diesen Werkzeugen vorfindet, ist in Abb. 4.2.1.1 dargestellt. Sie besteht im Wesentlichen aus zwei eng miteinander verwobenen Hauptbestandteilen: Analysemodul(e) mit Algorithmen zur simulativen

und/oder analytischen Modellauswertung und einer meist sehr umfangreichen graphischen Benutzeroberfläche.

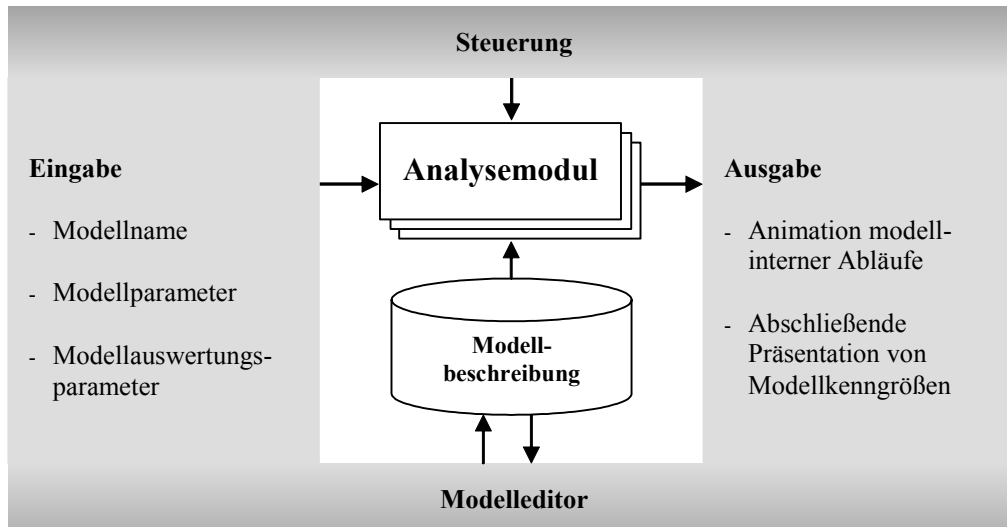


Abb. 4.2.1.1: Monolithischer SW-Aufbau klassischer Modellierungswerkzeuge

Zur Modellerstellung stellt die Graphikoberfläche dem Modellierer einen Modelleditor zur Verfügung, der es ermöglicht, Modelle zu editieren und anschließend in einem bestimmten Modellbeschreibungsformat abzuspeichern. Modelle, die in diesem Beschreibungsformat vorliegen, stellen wiederum die Eingabe für die zur Verfügung stehenden Modellauswertungsalgorithmen dar. Neben dem Editieren von Modellen erlaubt die Graphikoberfläche dem Modellierer, Modellexperimente zu spezifizieren, diese kontrolliert auszuführen und im Anschluss daran die berechneten Ergebnisse darzustellen. Häufig wird auch die Möglichkeit geboten, Zustandsänderungen simulierter Prozesse zur Laufzeit zu animieren.

Wie bereits erwähnt, blicken viele Altwerkzeuge auf eine lange Entwicklungsgeschichte zurück. Den ältesten Softwareteil eines solchen Werkzeugs stellen i.a. die Algorithmen zur Modellauswertung dar, die meist in einer prozeduralen Programmiersprache wie C, Fortran oder Pascal implementiert wurden, da zu Entwicklungsbeginn objektorientierte Programmiersprachen noch nicht zur Verfügung standen. Aufgrund ihrer hohen Effizienz und Bewährtheit wurde die prozedurale Implementierung der Modellauswertungsalgorithmen in der Regel bis heute unverändert beibehalten. Die graphische Benutzeroberfläche wurde dagegen häufig unter Verwendung moderner objektorientierter Graphikbibliotheken reimplementiert, um dem Werkzeug ein zeitgemäßes und konkurrenzfähiges Erscheinungsbild zu geben. Die Kopplungsschnittstelle zwischen Graphikoberfläche und (den) Analysemodul(en) basiert nur in sehr seltenen Fällen auf einem Komponentenstandard, sondern stellt meist ein komplexes und hochgradig plattformabhängiges Schnittstellengeflecht dar. Solche komplexen von mehreren Entwicklungsgenerationen implementierten Systeme sind naturgemäß schwierig zu warten, zu modifizieren und zu erweitern. Die vielfältigen Nachteile, die sich aus einem solchen Softwareaufbau ergeben,

werden im Folgenden am Beispiel von Petrinetz-basierten Modellierungswerkzeugen<sup>2</sup> im Detail aufgezeigt.

Heute existieren ca. 100 verschiedene Petrinetz-Werkzeuge. Eine detaillierte Auflistung dieser Werkzeuge findet sich im World Wide Web unter [PN1]. Dort wird auch die Gelegenheit geboten, diese Werkzeuge bezüglich ihres Funktionsumfangs miteinander zu vergleichen. Führt man einen solchen Vergleich durch, stellt man eine große Implementierungsvielfalt fest hinsichtlich

- graphischer Modelleditoren

76 Werkzeuge bieten zur Erstellung von Petrinetzmodellen graphische Editoren an. Die übrigen Werkzeuge stellen lediglich einen Texteditor zur Verfügung.

- Tokenspiel-Animationen

50 Werkzeuge bieten dem Benutzer die Möglichkeit, Markierungswechsel graphisch animiert zu verfolgen.

- Strukturanalysen

52 Werkzeuge bieten Algorithmen zur Strukturanalyse von Petrinetzen (Platz-, Transitionsinvarianten, Lebendigkeit, etc.) an.

- Verhaltensanalyse

39 Werkzeuge stellen Algorithmen zur Analyse des dynamischen Verhaltens von zeitbehafteten Petrinetzen (Mittelwertberechnungen im stationären Zustand, transiente Verhaltensanalyse) zur Verfügung.

Das breite Spektrum vorhandener Werkzeuge ist grundsätzlich zu begrüßen, da dem Modellierer vielfältige Möglichkeiten zum Experimentieren mit den verschiedensten Petrinetzarten<sup>3</sup> geboten werden. Von großem Nachteil sind jedoch der vorherrschend monolithische Werkzeugaufbau sowie die Vielfalt an unterschiedlichen Modellbeschreibungsformaten. Dadurch wird die gleichzeitige Nutzung des Funktionsumfangs mehrerer Petrinetz-Werkzeuge innerhalb einer Modellierungsstudie erheblich erschwert oder ganz verhindert. So ist es beispielsweise nicht ohne weiteres möglich, ein Petrinetzmodell mit dem Modelleditor eines bestimmten Petrinetz-Werkzeugs zu editieren und anschließend zum Auswerten des Modells Analysemodule eines anderen Petrinetz-Werkzeugs zu verwenden. Auch der Austausch bzw. die Kombination unterschiedlicher Werkzeugteile ist aufgrund uneinheitlicher Schnittstellen und unzureichender Dokumentation meist ein äußerst schwieriges und langwieriges Unterfangen mit unsicherem Ausgang.

Das veraltete Architekturkonzept monolithischer Komplettlösungen im Bereich der Modellierungswerkzeuge stellt sowohl für den Entwickler als auch für den Anwender einen äußerst un-

---

<sup>2</sup> Hierzu hätte man durchaus auch Modellierungswerkzeuge heranziehen können, die auf anderen Modellierungstechniken wie beispielsweise erweiterten Warteschlangennetzen basieren, da hier die Situation sehr ähnlich aussieht.

<sup>3</sup> Standard-, Zeit-, stochastische, gefärbte Petrinetze

befriedigenden Zustand dar. Neben den oben beschriebenen Problemen werden dadurch vor allem Weiterentwicklungen und die Integration innovativer Neuerungen erheblich erschwert. Auch die heute zunehmend lauter werdenden Forderungen nach Web- und HLA-Fähigkeit<sup>4</sup> sowie Möglichkeiten zum kollaborativen Arbeiten an einem Modell sind unter dieser Voraussetzung nur sehr schwer zu erfüllen. Dazu bedarf es eines offenen Architekturstandards sowie standardisierter Modellaustauschformate. Solche Standards bilden die Themenschwerpunkte des nun folgenden Abschnitts.

## 4.2.2 Entwurf einer komponentenorientierten Softwarearchitektur für Modellierungswerkzeuge

Die Softwarearchitektur<sup>5</sup> legt auf einer hohen Abstraktionsstufe die Struktur und den Aufbau eines Softwaresystems aus einzelnen Teilelementen fest und beschreibt die Schnittstellen dieser Elemente und deren Zusammenwirken zur Realisierung des Gesamtsystems. Darüber hinaus werden auch nichtfunktionale Anforderungen festgelegt, wie beispielsweise Leistungsfähigkeit, Wartbarkeit, Veränderbarkeit, Wiederverwendbarkeit und Einfachheit. Die Bedeutung solcher Eigenschaften ist nicht zu unterschätzen, da deren gewünschte Ausprägung den Aufbau eines Software-Systems wesentlich beeinflussen kann.

Bei der Festlegung der Softwarearchitektur werden häufig grundlegende Entwurfsentscheidungen getroffen, die in ihren Auswirkungen eingehend betrachtet werden müssen, da sie für die nachfolgende Realisierungsphase bindend sind. Eine tragfähige Softwarearchitektur ist von entscheidender Bedeutung, weil sie den "Bauplan" des Gesamtsystems darstellt und somit eine gemeinsame Basis und Anleitung

- zum Verständnis des Gesamtsystems,
- zur Identifikation und Nutzung allgemeiner und wiederverwendbarer Teilelemente,
- zur Durchführung der einzelnen Entwicklungsschritte oder -phasen sowie
- zur Wartung und zur möglichen Weiterentwicklung des Systems

bietet.

Mit dem Ziel der Komplexitätsbeherrschung wird die Softwarearchitektur in der Regel nicht durch ein einzelnes großes Architekturmodell beschrieben, das sämtliche relevanten Architektur Aspekte des zu erstellenden Systems enthält. Vielmehr wird die Gesamtarchitektur durch verschiedene Teilmodelle beschrieben, die erst zusammengefügt ein vollständiges Bild des zu entwickelnden Systems ergeben. Jedes Teilmodell beschreibt die Architektur aus einer speziellen Sichtweise und beschränkt sich auf die Beschreibung der dort relevanten Architektur Aspekte.

---

<sup>4</sup> [StK198], [LaSU99] und [Stra99] beschäftigen sich im Detail mit der Problemstellung, existierende Modellierungswerkzeuge Web- und (vor allem) HLA-tauglich zu machen

<sup>5</sup> eine umfassende Übersicht über existierende Definitionen zum Begriff "Softwarearchitektur" ist im WWW unter [SWA1] zu finden



Zu Beginn des Entwicklungsprozesses wird zunächst ein initiales fachliches Modell entwickelt, das hauptsächlich die funktionalen Aspekte des zu erstellenden Softwaresystems beschreibt. Die Modellierungselemente sind die Anwendungsprozesse, die mit dem System unterstützt werden sollen, sowie die Aktivitäten, die von den zukünftigen Anwendern des Systems ausgeführt werden sollen. Zur übersichtlichen Darstellung komplexer fachlicher Zusammenhänge werden heute in zunehmendem Maße Beschreibungsmethoden aus der UML (Unified Modeling Language) [Burk99] eingesetzt. Ein wesentliches Ergebnis dieses ersten Arbeitsschrittes ist die so genannte Facharchitektur, die aus Sicht des Anwenders die im zu erstellenden Softwaresystem vorkommenden Fachkomponenten sowie deren Beziehungen untereinander beschreibt.

Die Facharchitektur wird im nächsten Entwicklungsschritt in eine softwaretechnische Architektur überführt, die hauptsächlich implementierungsspezifische Aspekte des Systems beschreibt. Soll das zu entwickelnde Softwaresystem komponentenorientiert aufgebaut werden, müssen zunächst die einzelnen Softwarekomponenten und deren Beziehungen untereinander herausgearbeitet sowie deren Einbettung in das zugrunde liegende Komponentenmodell festgelegt werden. In Abhängigkeit des zur Systemimplementierung vorgesehenen Programmierparadigmas wird die softwaretechnische Architektur nun weiter verfeinert. Entscheidet man sich beispielsweise für eine objektorientierte Implementierung der Softwarekomponenten, sind die verwendeten Klassen und Objekte sowie ihre wechselseitigen Beziehungen und Interaktionen zu beschreiben, was heute häufig unter Verwendung bzw. in Anlehnung an objektorientierte Analyse- und Designmuster geschieht.

Im letzten Entwicklungsschritt wird schließlich bestimmt, in welcher systemtechnischen Landschaft die in der softwaretechnischen Architektur definierten Komponenten ablaufen sollen. Hier werden typischerweise die einzusetzenden Hardware-, Telekommunikations- und Middlewaresysteme in ihrem Zusammenspiel und unter Berücksichtigung ihrer räumlichen Verteilung festgelegt, wobei i.a. auf bereits existierende Infrastrukturen aufgesetzt wird, da Architekturprojekte in aller Regel nicht auf der grünen Wiese begonnen werden.

Abschließend sei angemerkt, dass die Zerlegung einer Gesamtarchitektur in verschiedene Teilmodelle nicht vollkommen unproblematisch ist, da folgende Aspekte zu berücksichtigen sind:

- Die Teilmodelle müssen zusammen ein vollständiges Bild ergeben, d.h. in ihrer Gesamtheit alle relevanten Aspekte des zu erstellenden Softwaresystems abdecken.
- Die Teilmodelle sollten möglichst redundanzfrei sein. Jeder modellierte Aspekt sollte möglichst genau einem Teilmodell zugeordnet sein (Orthogonalität).
- Die einzelnen Teilmodelle müssen zueinander konsistent sein und dürfen sich nicht widersprechen.
- Bestehende Zusammenhänge zwischen unterschiedlichen Teilmodellen müssen zusätzlich beschrieben werden.

In den nun folgenden Abschnitten wird, basierend auf der oben beschriebenen Vorgehensweise, zunächst einmal eine Facharchitektur für Modellierungswerkzeuge herausgearbeitet. Anschließend wird aufgezeigt, wie diese Facharchitektur auf eine zeitgemäße software- bzw. sys-

temtechnische Architektur abgebildet werden kann. Abschließend wird auf Möglichkeiten zur Einbindung existierender Werkzeuge bzw. Werkzeugteile eingegangen.

### 4.2.3 Werkzeugkomponenten aus fachlicher Sicht

Analysiert man die beim Umgang mit Modellierungswerkzeugen ablaufenden Anwendungsprozesse, stellt man fest, dass der Anwender im Wesentlichen mit drei verschiedenen Fachkomponenten in Berührung kommt. Dabei handelt es sich wie in Abb. 4.2.3.1 dargestellt um Modelleditoren, Komponenten zur Auswertung des dynamischen Modellverhaltens und Spezialkomponenten, wobei jede dieser Komponenten in sehr unterschiedlichen Ausprägungen vorkommen kann. Im Gegensatz zur monolithischen Komplettlösung aus Abb. 4.2.1.1 besitzt jede der in Abb. 4.2.3.1 dargestellten Fachkomponenten eine eigene graphische Benutzeroberfläche, was das Modellierungswerkzeug insgesamt überschaubarer macht. Darüber hinaus werden dadurch Modifikationen und Erweiterungen stark vereinfacht. Auf die vorteilhaften Eigenschaften der in Abb. 4.2.3.1 dargestellten Werkzeugarchitektur wird noch einmal ausführlich in Abschnitt 6.1 eingegangen. In dem dort beschriebenen Forschungsprojekt wurde, ausgehend von diesem Architekturkonzept, ein Web-basiertes Werkzeug zur dynamischen Prozessoptimierung realisiert.

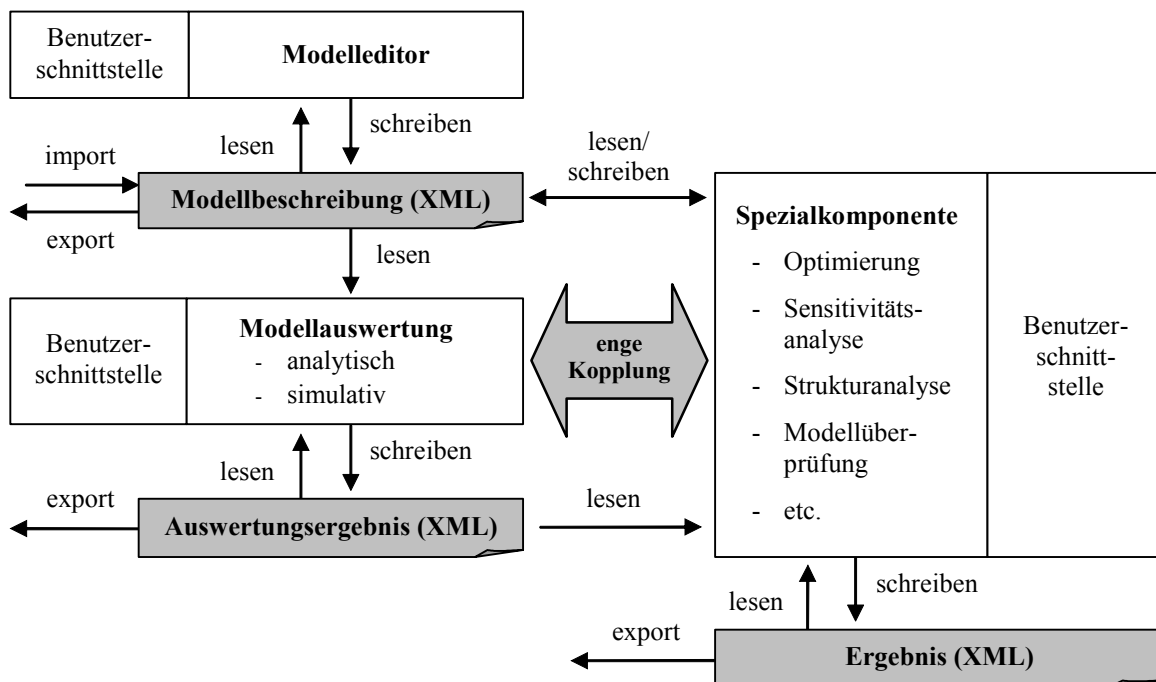


Abb. 4.2.3.1: Architektur eines Modellierungswerkzeugs aus fachlicher Sicht

Im Folgenden werden die in Abb. 4.2.3.1 vorkommenden Fachkomponenten näher beschrieben und es wird auf deren Abhängigkeiten untereinander eingegangen.

### 4.2.3.1 Modelleditoren

Modelleditoren ermöglichen es dem Anwender, Modelle neu zu erstellen bzw. bereits vorhandene Modelle zu modifizieren. Als Ergebnis des Editiervorgangs wird eine Modellbeschreibung erzeugt, die wiederum als Eingabe für die Auswertungs- bzw. Spezialkomponenten dient. Der grundsätzliche Aufbau einer Modellbeschreibung ist in Abb. 4.2.3.1.1 dargestellt. Das Editieren von Modellen kann in graphischer und/oder textueller Form erfolgen. Bei netz- und graphenbasierten Modellierungstechniken gehören heute Graphikeditoren zum Standard. Um von einer Auswertungs- bzw. Spezialkomponente eingelesen und bearbeitet werden zu können, muss die graphische Modellrepräsentation in eine textuelle Repräsentation überführt werden.

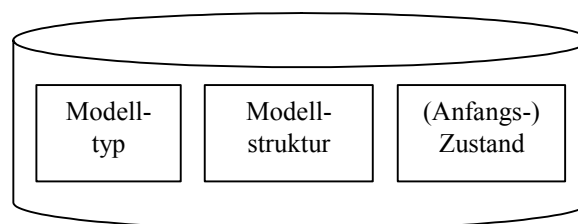


Abb. 4.2.3.1.1 Grundsätzlicher Aufbau einer Modellbeschreibung

### 4.2.3.2 Komponenten zur Modellauswertung

Komponenten zur Modellauswertung beinhalten Verfahren zur stationären und/oder transienten Analyse des dynamischen Modellverhaltens. Untersuchungen dieser Art können entweder analytisch unter Verwendung mathematischer Methoden oder simulativ durch Nachspielen der im Modell ablaufenden Vorgänge vorgenommen werden. Die graphische Benutzerschnittstelle einer Auswertungskomponente sollte dem Modellierer eine möglichst komfortable Durchführung von Modellauswertungsexperimenten ermöglichen, wobei grundsätzlich folgende Arbeitsschritte unterstützt werden müssen:

- Vorbereitung des Modellauswertungsexperiments

In diesem Arbeitsschritt wählt der Modellierer zunächst über die Benutzerschnittstelle aus den vorhandenen Modellbeschreibungen das auszuwertende Modell aus. Anschließend sind Werte für die veränderlichen Modellparameter festzulegen. Dies kann entweder über den Modelleditor oder die Graphikoberfläche der Modellauswertungskomponente erfolgen. Abschließend muss das von der Auswertungskomponente angebotene Auswertungsverfahren geeignet parametrisiert werden. Bei simulativer Modellauswertung sind hier beispielsweise die Simulationsdauer sowie der Umfang der aufzuzeichnenden Simulationsdaten festzulegen.

- Kontrollierte Durchführung der Modellauswertung

Zu diesem Zweck sollte die Benutzerschnittstelle dem Modellierer das Starten, Anhalten, Weiterführen und Beenden von Modellauswertungen ermöglichen. Heute werden zuneh-

mend auch im Detaillierungsgrad variable Animationen der modellinternen Abläufe verlangt, da diese wesentlich zum Systemverständnis beitragen können.

- Ausgabe von Modellauswertungsergebnissen

Aufgabe dieses Teils der Benutzerschnittstelle ist es, dem Modellierer Zwischen- und Endergebnisse der Modellauswertung in anschaulicher Form zu präsentieren. Darüber hinaus sollte die Möglichkeit bestehen, die angezeigten Ergebnisse in einer Ergebnisdatei abzuspeichern.

### 4.2.3.3 Spezialkomponenten

Modellauswertungskomponenten ermöglichen es dem Modellierer, das dynamische Modellverhalten bei von ihm ausgewählten Einstellungen der Modellparameter zu analysieren. Allein durch solche punktuellen Analysen lassen sich bei der heutigen Modellkomplexität die gesteckten Modellierungsziele in der Regel jedoch nicht mehr erreichen. Dazu bedarf es weitergehender Untersuchungen, bei denen grundsätzlich folgende Methoden zum Einsatz kommen können:

- Analytische Methoden, die gänzlich ohne punktuelle Modellauswertungsexperimente auskommen und ausschließlich auf der Modellbeschreibung aufsetzen (beispielsweise Verfahren zur Strukturanalyse von Petrinetzen).
- Empirische Methoden, die versuchen, durch wiederholte Modellauswertungsexperimente mit systematischer Parametervariation das Untersuchungsziel zu erreichen (beispielsweise direkte Optimierungsverfahren).

In der in Abb. 4.2.3.1 dargestellten Werkzeugarchitektur werden diese Methoden in Form von so genannten Spezialkomponenten angeboten. Mögliche Analyseziele solcher Spezialkomponenten sind

- Bestimmung optimaler Modellparametereinstellungen (auf die Konzeption, Realisierung und Bewertung einer Spezialkomponente zur Modelloptimierung wird detailliert in Kapitel 5 eingegangen)
- Bestimmung empfindlicher Modellparameter
- Überprüfung der Modellstruktur
- Prüfung der hinreichenden Übereinstimmung von Modell und Originalsystem

Die Benutzerschnittstelle einer Spezialkomponente muss es dem Anwender zunächst einmal ermöglichen, aus den vorhandenen Modellbeschreibungen das zu untersuchende Modell auszuwählen. Handelt es sich um eine Spezialkomponente, die auf empirischen Methoden basiert, muss zusätzlich die Möglichkeit geboten werden, die zu variierenden Modellparameter sowie die Bereichsgrenzen, in denen sich die Variationen bewegen sollen, festzulegen. Abschließend muss eine geeignete Modellauswertungskomponente zur Durchführung der bei der empirischen Analyse anfallenden Modellauswertungsexperimente ausgewählt werden. Um empirische Modelluntersuchungen dieser Art zu automatisieren, müssen die dazu eingesetzten Spezialkompo-

nenten mit Modellauswertungskomponenten in geeigneter Weise gekoppelt werden (siehe dazu auch Abschnitt 4.2.3.4).

Spezialkomponenten haben in letzter Zeit enorm an Bedeutung gewonnen, da sie viel versprechende Möglichkeiten bieten, um aus komplexen Modellen in akzeptabler Zeit die zum Erreichen weitgesteckter Modellierungsziele erforderlichen Informationen zu gewinnen. Dies ist deshalb von entscheidender Bedeutung, weil sich erst dadurch der i.a. sehr hohe Modellerstellungsaufwand rechtfertigen lässt. Da anzunehmen ist, dass die Entwicklung im Bereich der Spezialkomponenten rasch voranschreitet, sollten Modellierungswerkzeuge möglichst so aufgebaut sein, dass sich Neuerungen dieser Art problemlos und schnell integrieren lassen.

#### 4.2.3.4 Abhängigkeiten zwischen den Fachkomponenten

Zwischen den in Abb. 4.2.3.1 dargestellten Fachkomponenten lassen sich zwei Arten von Abhängigkeiten unterscheiden:

- Indirekte Abhängigkeiten auf Basis von Dokumentenaustausch (lose Kopplung)
- Direkte Abhängigkeiten auf Basis von (entfernten) Prozedur- bzw. Methodenaufrufen (enge Kopplung)

Modelleditoren und Modellauswertungskomponenten brauchen prinzipiell nur lose miteinander gekoppelt zu werden, da das Editieren von Modellen völlig unabhängig und asynchron von der Modellauswertung vorgenommen werden kann. Einzige Voraussetzung für die Zusammenarbeit beider Komponenten ist, dass die vom Modelleditor erzeugte Modellbeschreibung von der Modellauswertungskomponente auch interpretiert werden kann.

Bei Spezialkomponenten, die auf rein analytischen Methoden basieren, genügt ebenfalls eine lose Kopplung mit den Modelleditoren, da hier zur Modellanalyse ausschließlich auf die Modellbeschreibung zurückgegriffen wird. Werden in einer Spezialkomponente jedoch empirische Methoden verwendet, die zum Erreichen des Untersuchungsziels Modellauswertungsexperimente benötigen, müssen zur Durchführung dieser Experimente Modellauswertungskomponenten (entfernt) aufgerufen werden. Zur Realisierung einer solchen engen Kopplung zwischen Komponenten auf Client/Server-Basis stehen heute mächtige Middleware-Systeme wie beispielsweise CORBA, RMI oder DCOM/COM+ zur Verfügung. Bei den zwischen Spezial- und Auswertungskomponente ausgetauschten Daten handelt es sich im Einzelnen um

- Parameter, welche die Arbeitsweise des aufgerufenen Modellauswertungsverfahrens beeinflussen. Bei simulativer Modellauswertung kann es sich dabei beispielsweise um die Simulationsdauer oder den Umfang der aufzuzeichnenden Simulationsdaten handeln. Diese Einstellungen sind über die Graphikoberfläche von der das Modellauswertungsverfahren aufrufenden Spezialkomponente festzulegen.
- die Rückmeldung der Modellauswertungskomponente an die Spezialkomponente über die Terminierung der Modellauswertung.

Daten, die das zu analysierende Modell betreffen, werden wie in Abb. 4.2.3.1 dargestellt, dokumentenbasiert über die Modellbeschreibung bzw. die von der Modellauswertungskomponente erzeugte Ergebnisdatei zur Verfügung gestellt. Dazu muss die Spezialkomponente vor

dem Aufruf der Modellauswertungskomponente die Modellparametereinstellung, unter der die Modellauswertung durchgeführt werden soll, in die Modellbeschreibung eintragen. Des Weiteren muss nach erfolgter Modellauswertung die Modellauswertungskomponente eine Datei erzeugen, in der die berechneten Modellauswertungsergebnisse eingesehen werden können. Eine solche Ergebnisdatei wird immer dann vorausgesetzt, wenn die in der Spezialkomponente verwendete Experimentierstrategie die Modellparametereinstellung für das nächste Experiment aus dem Auswertungsergebnis des vorangegangenen Experiments berechnet. Dies ist bei den meisten Experimentierstrategien der Fall, unter anderem auch bei den in Kapitel 5 beschriebenen direkten Verfahren zur Parameteroptimierung von Simulationsmodellen.

#### 4.2.3.5 Beschreibung der Modellierungsdaten

Die Integration der in Abb. 4.2.3.1 dargestellten Werkzeugkomponenten basiert neben engen Kopplungsmechanismen vor allem auch auf dem Austausch von Dokumenten. Im Einzelnen handelt es sich dabei um die vom Modelleditor erzeugten Modellbeschreibungen sowie um die von den Modellauswertungs- bzw. Spezialkomponenten erzeugten Ergebnisdateien. Um den Dokumentenaustausch zwischen den Werkzeugkomponenten möglichst einfach und flexibel zu gestalten, aber auch um einen möglichst offenen Austausch mit anderen Applikationen zu ermöglichen, ist ein geeignetes Beschreibungsformat von entscheidender Bedeutung.

Wie bereits in Abschnitt 3.2 beschrieben etabliert sich zunehmend die eXtensible Markup Language (XML) als Universalformat zur Speicherung von Dokumenten und für den Datenaustausch zwischen heterogenen Anwendungen. Auch zur Beschreibung von Modellierungsdaten, die zwischen den Komponenten der in Abb. 4.2.3.1 dargestellten Werkzeugarchitektur ausgetauscht werden, ist XML aufgrund seiner in Abschnitt 2.4.3.3 beschriebenen Eigenschaften hervorragend geeignet. Die großen internationalen Standardisierungsorganisationen haben diese Vorzüge bereits erkannt und sind heute dabei, zu den verschiedensten Modellierungstechniken einheitliche XML-basierte Austauschformate zu erarbeiten. Die ISO (International Organisation for Standardisation) beispielsweise treibt unter anderem die Standardisierung eines Modellaustauschformats für erweiterte Petrinetze voran. Der aktuelle Stand dieser noch laufenden Arbeiten kann im World Wide Web unter [PN2] eingesehen werden. Ausgehend davon wurde im Rahmen des in Abschnitt 6.1 beschriebenen Forschungsprojekts ein XML-basiertes Austauschformat für stochastische Petrinetze entwickelt. Eine ausführliche Beschreibung dieses Formats findet sich in [Schi02].

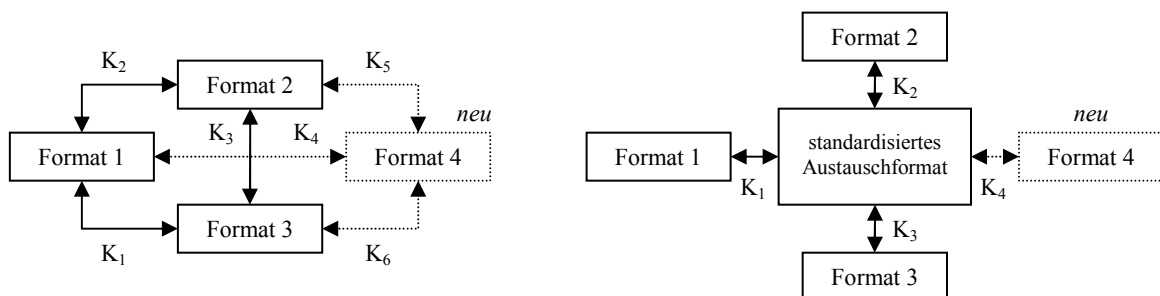


Abb. 4.2.3.5.1: Gegenüberstellung des Konvertierungsaufwands ohne und mit standardisiertem Austauschformat

Ein XML-basiertes Austauschformat ermöglicht nicht nur Interoperabilität zwischen Modellierungswerkzeugen, sondern vereinfacht auch die Integration existierender Altwerkzeuge, die auf derselben Modellierungstechnik basieren aber jeweils unterschiedliche Modellbeschreibungsformate verwenden. Wie in Abb. 4.2.3.5.1 angedeutet, steigt bei Verwendung eines standardisierten Modellaustauschformats die Anzahl der zur Werkzeugintegration benötigten Konverter  $K_i$ ,  $i \in \mathbb{N}$  mit zunehmender Werkzeuganzahl lediglich linear an. Ohne standardisiertes Modellaustauschformat würde die Zahl der benötigten Konverter dagegen quadratisch<sup>6</sup> ansteigen.

#### 4.2.4 Mögliche Realisierung durch technische Komponenten

Abb. 4.2.4.1 zeigt eine mögliche software- bzw. systemtechnische Realisierung der in Abb. 4.2.3.1 dargestellten Facharchitektur für Modellierungswerkzeuge. Dabei handelt es sich um eine moderne 4-schichtige Client/Server Architektur, bestehend aus einer Präsentations-, Zugriff-, Fachlogik- und Datenhaltungsschicht.

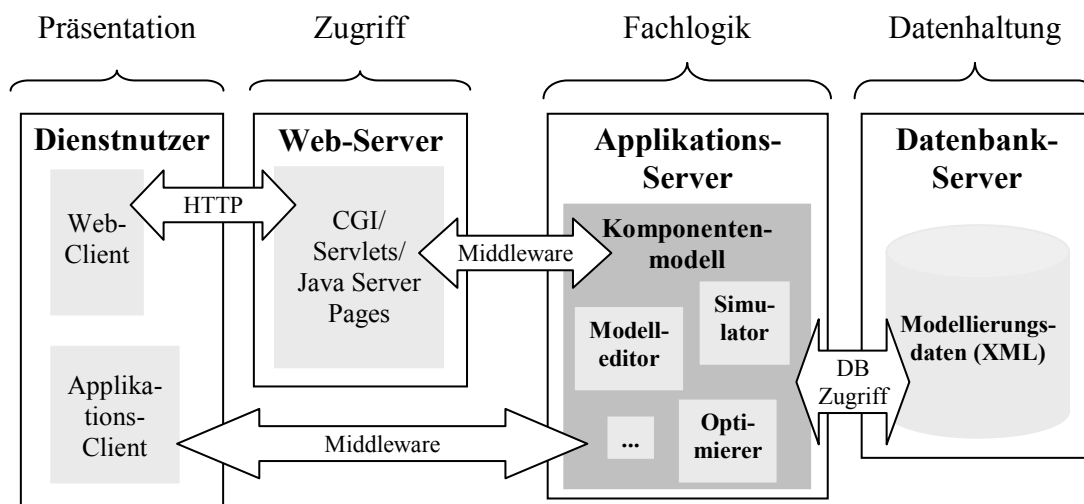


Abb. 4.2.4.1: Mögliche Realisierung durch technische Komponenten

Der Anwender kann entweder über herkömmliche Applikationsklienten oder über das World Wide Web auf die verschiedenen vom Applikationsserver zur Verfügung gestellten Werkzeugkomponenten zugreifen. Eine flexible Präsentationsschicht ist heute von entscheidender Bedeutung, insbesondere dann, wenn kollaboratives Arbeiten mehrerer Personen an einem Modell ermöglicht werden soll. Unterstützungen dieser Art werden zunehmend gefordert, da es sich bei der Erstellung komplexer Modelle in der Regel um interdisziplinäre Aufgabenstellungen handelt, die von mehreren Experten aus unterschiedlichen Bereichen bearbeitet werden. Daher sollte die Präsentationsschicht neben Mechanismen zur Synchronisation gemeinsamen Arbei-

<sup>6</sup> um  $n$  unterschiedliche Modellbeschreibungsformate ineinander zu überführen, werden  $(n^2-n)/2$  bidirektionale Konverter benötigt

tens auch unterschiedliche auf die jeweiligen Bedürfnisse und den Kenntnisstand der beteiligten Experten zugeschnittene Sichten auf die Werkzeugkomponenten anbieten.

Zur Realisierung der auf dem Applikationsserver zur Verfügung gestellten Werkzeugkomponenten eignen sich grundsätzlich die heute existierenden de facto Komponentenstandards wie beispielsweise (Enterprise) JavaBeans, (D)COM/COM+ oder das Corba Component Model<sup>7</sup>, wobei die Entscheidung für ein konkretes Komponentenmodell letztendlich stark von der bestehenden Systemlandschaft abhängen wird. Zur Realisierung der vierten Architekturschicht, welche die persistente Speicherung von Modellierungsdaten ermöglichen soll, können gängige Dateiverwaltungs- bzw. Datenbanksysteme eingesetzt werden.

## 4.2.5 Einbindung existierender Werkzeuge

Im Folgenden werden Möglichkeiten vorgestellt, wie sich Altwerkzeuge bzw. Altwerkzeugteile nachträglich in die in Abb. 4.2.3.1 dargestellte komponentenorientierte Werkzeugarchitektur einbinden lassen. Grundvoraussetzung für eine solche Einbindung ist natürlich, dass von den betreffenden Werkzeugen dieselbe Modellierungstechnik unterstützt wird. Gerade bei Altwerkzeugen, die sich über einen langen Zeitraum evolutionär entwickelt und in der Praxis bewährt haben, macht eine (teilweise) Integration durchaus Sinn, weil dadurch Entwicklungsaufwand vermieden und bereits (aufwändig) erstellte Modelle weiterverwendet werden können. Aus den beiden in Abschnitt 4.2.3.4 beschriebenen Arten von Abhängigkeiten zwischen den Fachkomponenten der Werkzeugarchitektur ergeben sich grundsätzlich folgende Integrationsmöglichkeiten

- (lose) Integration kompletter Altwerkzeuge auf Basis von Dokumentenaustausch

Dazu wird lediglich ein bidirektionaler Konverter benötigt, der das vom Altwerkzeug unterstützte proprietäre Modellbeschreibungsformat in das vom komponentenorientierten Werkzeug verwendete XML-basierte Modellbeschreibungsformat umwandelt (und umgekehrt). Durch einen solchen Konverter können Modellbeschreibungen flexibel zwischen den betreffenden Modellierungswerkzeugen ausgetauscht werden. Von großem Vorteil ist, dass diese Art der Integration keinerlei Änderungen am Altwerkzeug erfordert.

- (enge) Integration von Altwerkzeugteilen auf Basis von (entfernten) Prozedur- bzw. Methodenaufrufen

Die lose Integration durch Dokumentenaustausch ermöglicht zwar den flexiblen Austausch von Modellbeschreibungen, der direkte Zugriff auf bestimmte Altwerkzeugteile<sup>8</sup> ist jedoch nicht ohne weiteres möglich, da deren Schnittstellen i.a. nicht mit denen der komponentenorientierten Werkzeugarchitektur übereinstimmen. Um dennoch den direkten Zugriff zu ermöglichen, müssen die Schnittstellen der Altwerkzeugteile durch so genannte Wrapper<sup>9</sup> entsprechend gekapselt werden. Deren Realisierung kann, je nach Umfang der erforderlichen

---

<sup>7</sup> das Corba Component Model stellt heute zwar den konzeptionell ausgereiftesten Komponentenstandard dar, eine konkrete Implementierung dieses Standards ist bislang jedoch noch nicht verfügbar

<sup>8</sup> beispielsweise der Aufruf eines bewährten Simulatorenkerns durch eine Spezialkomponente

<sup>9</sup> Wrapper werden häufig auch als Adapter bezeichnet



Schnittstellenanpassungen, unter Umständen einigen Aufwand erfordern. Das zu kapselnde Altwerkzeugteil lässt sich jedoch meist (nahezu) unverändert übernehmen.



## *Kapitel 5*

# **Konzeption, Realisierung und Bewertung einer Optimierungskomponente**

---

Heute existieren mächtige Techniken und Werkzeuge, die den Modellierer bei der Konzeption und Implementierung von Simulationsmodellen in umfassender Weise unterstützen. Strategien zum zielgerichteten Experimentieren mit bereits vorhandenen Simulationsmodellen, die es dem Modellierer ermöglichen, Aufschluss über das meist sehr komplexe Modellverhalten zu bekommen, werden von den meisten Simulationswerkzeugen jedoch nur in sehr unzureichender Form oder gar nicht angeboten. Daher müssen die zum Erreichen der Modellierungsziele notwendigen Experimentabfolgen heute noch vorwiegend manuell vom Modellierer geplant und durchgeführt werden, wobei der Erfolg im Wesentlichen von dessen Kenntnisstand und Intuition abhängt. Bei der heutigen Modellkomplexität lässt sich das Modellverhalten durch diese manuelle Art des Experimentierens jedoch nicht mehr in ausreichendem Maße erschließen, was in der Regel dazu führt, dass die gesteckten Modellierungsziele nicht erreicht werden können. Um diesen höchst unbefriedigenden Zustand zu verbessern, müssen effiziente Experimentierstrategien entwickelt und in die zur Verfügung stehenden Modellierungswerkzeuge integriert werden. Dadurch ließe sich der Nutzen und Stellenwert der Modellierung beträchtlich erhöhen und der meist sehr hohe Aufwand zur Erstellung von Simulationsmodellen besser rechtfertigen.

Aus den in Abschnitt 4.2 genannten Gründen bietet es sich an, Experimentierstrategien in Form von separaten Spezialkomponenten zu realisieren, die wie in Abbildung 4.2.3.1 dargestellt eigenständige und austauschbare Teile eines Modellierungswerkzeugs bilden. Zu den möglichen Aufgabenstellungen solcher Komponenten könnten zählen

- die Bestimmung empfindlicher Modellparameter,
- die Prüfung der hinreichenden Übereinstimmung von Modell und Originalsystem,
- die Ermittlung optimaler Modellparametereinstellungen.

Dieses Kapitel befasst sich im Detail mit der Konzeption, Realisierung und Bewertung einer Spezialkomponente zur effizienten Parameteroptimierung von Simulationsmodellen. In Abbildung 5.1 ist ein Modellierungswerkzeug dargestellt, das eine solche Optimierungskomponente anbietet. Die Aufgabe dieser Komponente besteht darin, dem Modellierer effiziente und möglichst einfach zu handhabende Strategien zur automatischen Steuerung der zur Modelloptimierung notwendigen Experimentabfolgen zur Verfügung zu stellen.

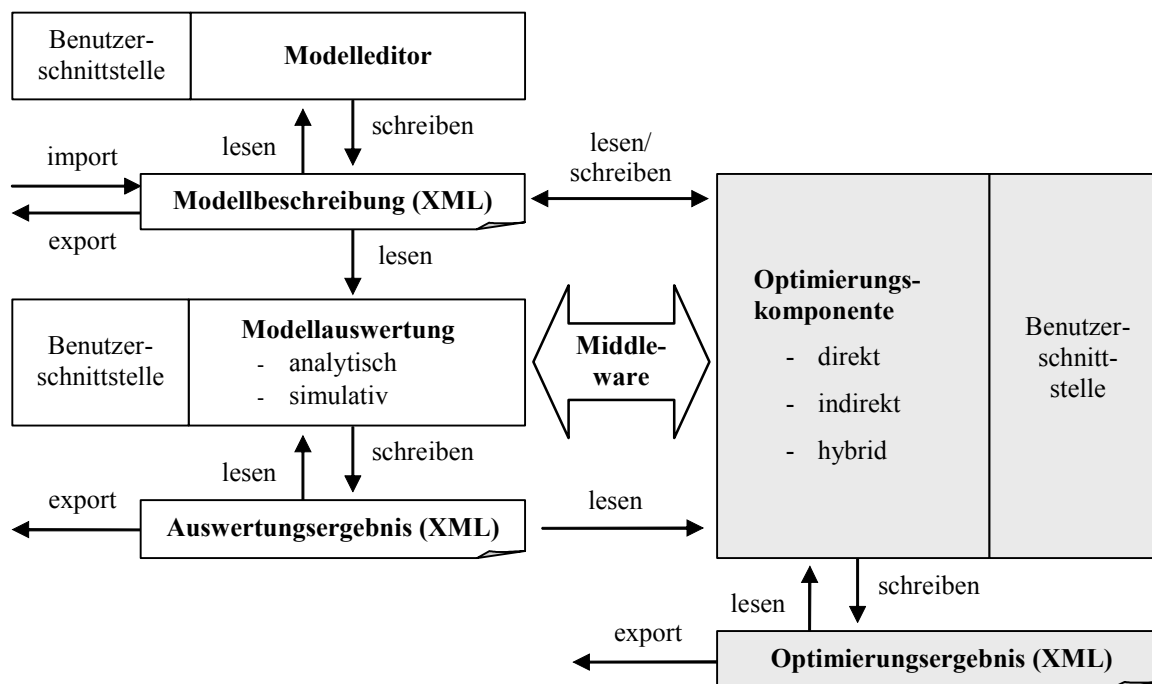


Abb. 5.1: Komponentenorientiertes Modellierungswerkzeug mit einer integrierten Spezialkomponente zur Modelloptimierung

Wie bereits in Abschnitt 2.3 beschrieben, lassen sich die zur Optimierung modellbasierter Zielfunktionen existierenden Verfahren in folgende Klassen einteilen:

- indirekte (analytische) Optimierungsverfahren, die eine mathematische Beschreibung der zu optimierenden Zielfunktion voraussetzen,
- universell einsetzbare direkte (empirische) Optimierungsverfahren, die zur Steuerung des Optimierungsprozesses ausschließlich Zielfunktionswerte verwenden,
- hybride Optimierungsverfahren, die Mischformen aus direkten und indirekten Verfahren darstellen.

Indirekte Optimierungsverfahren haben gegenüber direkten Methoden den Vorteil, dass keine aufwändigen Modellauswertungsexperimente benötigt werden. Leider ist die klassische mathematische Vorgehensweise zur Modelloptimierung jedoch nur in einigen wenigen speziellen Fällen möglich, da es insbesondere bei diskreten ereignisorientierten Simulationsmodellen i.a. große Schwierigkeiten bereitet, eine mathematische Beschreibung des Optimierungsproblems zu finden. Zur Optimierung komplexer Simulationsmodelle bleibt daher im Normalfall keine andere Wahl, als auf universell anwendbare direkte Optimierungsverfahren zurückzugreifen. Die heute zur Verfügung stehenden direkten Optimierungsmethoden sind für diese Aufgabe jedoch nur bedingt geeignet, da sie eine viel zu hohe Anzahl von Modellauswertungen benötigen, um zu einem zufriedenstellenden Ergebnis zu kommen. Bei sehr rechenintensiver Mo-

dellauswertung<sup>1</sup> kann eine Modelloptimierung mit herkömmlichen direkten Verfahren wie beispielsweise Genetischen Algorithmen oder dem Simulated Annealing mehrere Tage oder sogar Wochen in Anspruch nehmen. Um direkte Optimierungsverfahren für die Modelloptimierung zu qualifizieren, müssen also primär Maßnahmen zur Reduktion der Anzahl benötigter Modellauswertungen getroffen werden. Der nun folgende Abschnitt befasst sich ausführlich mit dieser Problematik.

## 5.1 Effiziente Verfahren zur direkten Parameteroptimierung von Simulationsmodellen

Die in Abschnitt 2.3.3 beschriebenen direkten Optimierungsverfahren gehen in der Regel recht verschwenderisch mit Zielfunktionsevaluierungen um<sup>2</sup>. Berechnete Zielfunktionswerte werden meist nur einmal verwendet und anschließend sofort wieder verworfen, wobei es durchaus vorkommen kann, dass im Optimierungsverlauf zu einem bestimmten Suchpunkt dessen Zielfunktionswert mehrmals berechnet wird. Bei einfach zu berechnenden Zielfunktionen kann diese Vorgehensweise noch toleriert werden. Bei aufwändigen modellbasierten Zielfunktionen sollte jedoch jede überflüssige Zielfunktionsevaluierung unbedingt vermieden werden. Dies lässt sich sehr einfach durch entsprechendes Durchsuchen der Optimierungstrajektorie<sup>3</sup> bewerkstelligen, die zu diesem Zweck abgespeichert werden muss. Weitere Einsparungen an Zielfunktionsevaluierungen lassen sich durch den kombinierten Einsatz globaler und lokaler Optimierungsverfahren erreichen. Der durch ein solches hybrides Suchverfahren erzielte Effizienzgewinn kann wiederum gewinnbringend in dessen mehrfache Anwendung auf ein und dasselbe Optimierungsproblem reinvestiert werden. Dadurch erhöhen sich mit jedem durchgeführten Optimierungslauf die Chancen, das globale Optimum aufzufinden. Insgesamt bietet die Mehrfachausführung hybrider Optimierungsverfahren eine Reihe von guten Möglichkeiten, die in der Optimierungstrajektorie enthaltenen Informationen gewinnbringend zur Beschleunigung des Optimierungsprozesses einzusetzen. In den folgenden Abschnitten werden die oben genannten Lösungsansätze zur Weiterentwicklung direkter Optimierungsverfahren näher beschrieben.

### 5.1.1 Atomare direkte Optimierungsverfahren

Den Ausgangspunkt für die in den folgenden Abschnitten vorgenommenen Weiterentwicklungen bilden die in Abschnitt 2.3.3 beschriebenen direkten Verfahren zur lokalen und globalen Optimierung. Untersucht man deren algorithmischen Aufbau, lassen sich folgende gemeinsame Bestandteile feststellen:

---

<sup>1</sup> Die Rechenzeiten für die Auswertung praxisrelevanter Simulationsmodelle bewegen sich häufig im Minutenbereich, können aber bei entsprechender Komplexität durchaus auch Stunden oder im Extremfall Tage ausmachen.

<sup>2</sup> Dies trifft insbesondere auf die probabilistisch arbeitenden Verfahren zur globalen Optimierung zu.

<sup>3</sup> Die Optimierungstrajektorie umfasst alle während des Optimierungsverlaufs erzeugten Suchpunkte und deren Zielfunktionswerte.

- Optimierungsalgorithmus A

Durch den iterativ arbeitenden Optimierungsalgorithmus werden das Optimierungsprinzip und die zur Optimierung eingesetzten Suchoperatoren festgelegt. Direkte Optimierungsverfahren basieren meist auf einem naturalogen Optimierungsprinzip. Evolutionäre Algorithmen beispielsweise funktionieren nach Prinzipien der biologischen Evolution, das Simulated Annealing beruht auf einem physikalischen Abkühlungsprozess und lokale Hill-Climbing Strategien ahmen die Vorgehensweise eines Bergsteigers nach.

- Datenstruktur D

Die Datenstruktur dient zur Repräsentation der Lösungsalternativen des zu bearbeitenden Optimierungsproblems. Der Optimierungsalgorithmus A manipuliert und modifiziert die in der Datenstruktur D vorliegenden Lösungsalternativen durch Anwendung seiner Suchoperatoren.

- Terminierungsbedingung T

Durch die Terminierungsbedingung werden die Kriterien festgelegt, die ausschlaggebend sind für den Abbruch des Optimierungsalgorithmus A.

- Kontrollparameter P

Mit den Kontrollparametern lassen sich der Optimierungsalgorithmus A, die Datenstruktur D sowie die Terminierungsbedingung T bezüglich ihrer Arbeitsweise manipulieren.

In Abb. 5.1.1.1 sind die Einzelbestandteile A, D, T, P und deren gegenseitige Beeinflussung untereinander graphisch dargestellt. Darüber hinaus ist der iterative Optimierungsablauf angedeutet. Um das Verfahren zu starten, müssen eine oder mehrere Startlösungen ( $L_{\text{start}}$ ) ausgewählt und die Kontrollparameter des Verfahrens eingestellt werden (kp). Nach erfolgter Terminierung liefert das Verfahren eine Optimierungstrajektorie  $o_T$  (sämtliche während der Optimierung erzeugten Suchpunkte und deren Zielfunktionswerte) und ein Optimierungsergebnis  $o_E$  (Suchpunkt(e) aus  $o_T$  mit dem besten Zielfunktionswert). Neben der meist zufällig erzeugten Menge von Startlösungen wird der Optimierungsverlauf entscheidend durch die gewählten Kontrollparametereinstellungen kp beeinflusst.

Optimierungsverfahren, welche die in Abb. 5.1.1.1 dargestellte algorithmische Grundstruktur aufweisen, werden im Folgenden als "atomar" bezeichnet und mit aos abgekürzt. Sie bilden die Grundbausteine für die in den nächsten Abschnitten vorgestellten hybriden Optimierungsmethoden. Je nach zugrunde liegendem Optimierungsprinzip sind atomare Verfahren eher zur lokalen oder globalen Suche geeignet. Um lokale von globalen Verfahren unterscheiden zu können, werden die Kürzel aos<sub>l</sub> und aos<sub>g</sub> eingeführt.

Wie bereits in Abschnitt 2.3.3 beschrieben, sind die heute verfügbaren atomaren Optimierungsverfahren nur bedingt geeignet, wenn es darum geht, das globale Optimum aufwändig zu berechnender simulationsbasierter Zielfunktionen zu ermitteln. Lokale Hill-Climbing Verfahren sind zwar in der Lage, eine Extremstelle sehr effizient und mit einer benutzerdefinierten Genauigkeit zu lokalisieren. Um eine global-optimale Lösung zu finden, müssen diese Verfahren jedoch in deren Einzugsbereich gestartet werden. Diese Einschränkung weisen globale Optimierungsverfahren aufgrund ihrer probabilistischen Suchoperatoren nicht auf. Hier muss je-

doch mit einem hohen Optimierungsaufwand (Anzahl benötigter Zielfunktionsevaluierungen) gerechnet werden, der vor allem durch die hohe Ineffektivität der probabilistischen Suchoperatoren in der Schlussphase der Optimierung bedingt ist. Ein weiterer Nachteil globaler Optimierungsverfahren liegt in der nicht gesicherten Ergebnisqualität.

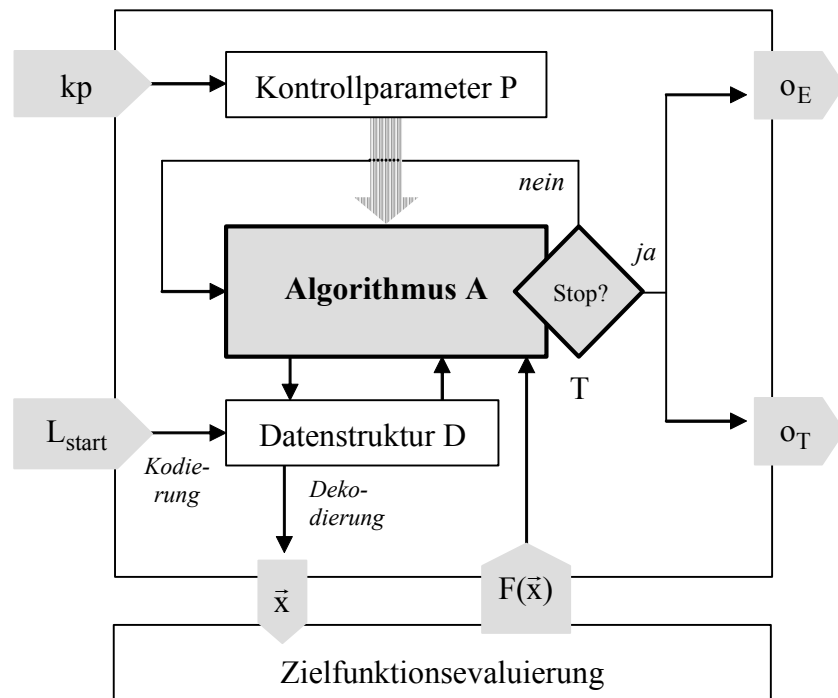


Abb. 5.1.1.1: Aufbau einer atomaren Optimierungsstrategie aos

Dass jedes atomare Optimierungsverfahren bestimmte Unzulänglichkeiten aufweist, ist nicht weiter verwunderlich, da nach dem No Free Lunch Theorem (siehe Abschnitt 2.3.2) kein direktes Optimierungsverfahren existieren kann, das allen anderen in allen Belangen überlegen ist. Im Folgenden wird aufgezeigt, dass es dennoch möglich ist, ausgehend von atomaren Optimierungsverfahren effizient und - falls die in Abschnitt 2.3.2 beschriebenen vereinfachenden Annahmen erfüllt sind - auch erfolgreich Modelloptimierung zu betreiben. Der dazu entwickelte Lösungsansatz basiert auf dem kombinierten Einsatz zweier komplementärer atomarer Optimierungsmethoden, die sich in ihren Sucheigenschaften gegenseitig ergänzen. Zur Realisierung eines solchen hybriden Optimierungsverfahrens hat es sich als sehr vorteilhaft herausgestellt, ein probabilistisches globales mit einem deterministischen lokalen Verfahren zu kombinieren. Der innerhalb eines solchen Verfahrens ablaufende Optimierungsprozess unterteilt sich in die beiden Phasen Vor- und Feinoptimierung [SySz94a], [SySz94b], [Syrj97]. Aufgabe der Voroptimierung ist die möglichst umfassende Exploration des gesamten Lösungsraums nach Regionen, die global-optimale Lösungen enthalten. Ausgehend vom Ergebnis der Voroptimierung besteht die Aufgabe der anschließenden Feinoptimierung darin, die optimale Lösung in einer solchen vielversprechenden Region möglichst effizient und exakt zu lokalisieren. Das daraus resultierende Gesamtverfahren, das die Stärken atomarer globaler und lokaler Optimierungsmethoden in sich vereint, wird im Folgenden als kombinierte 2-Phasen Optimierungsstrategie bezeichnet.

## 5.1.2 Kombinierte 2-Phasen Optimierung

Abb. 5.1.2.1 zeigt die Grundstruktur einer kombinierten 2-Phasen Optimierungsstrategie  $os_{2P}$ , deren Hauptbestandteile ein atomares globales Optimierungsverfahren  $aos_g$  und ein atomares lokales Verfahren  $aos_l$  bilden. Die beiden atomaren Optimierungsmethoden sind über eine Kopplungsschnittstelle miteinander verbunden, deren Aufgabe darin besteht, ausgehend vom Ergebnis des Voroptimierungsverfahrens  $aos_g$ , möglichst günstige Startbedingungen für das lokale Feinoptimierungsverfahren  $aos_l$  zu berechnen.

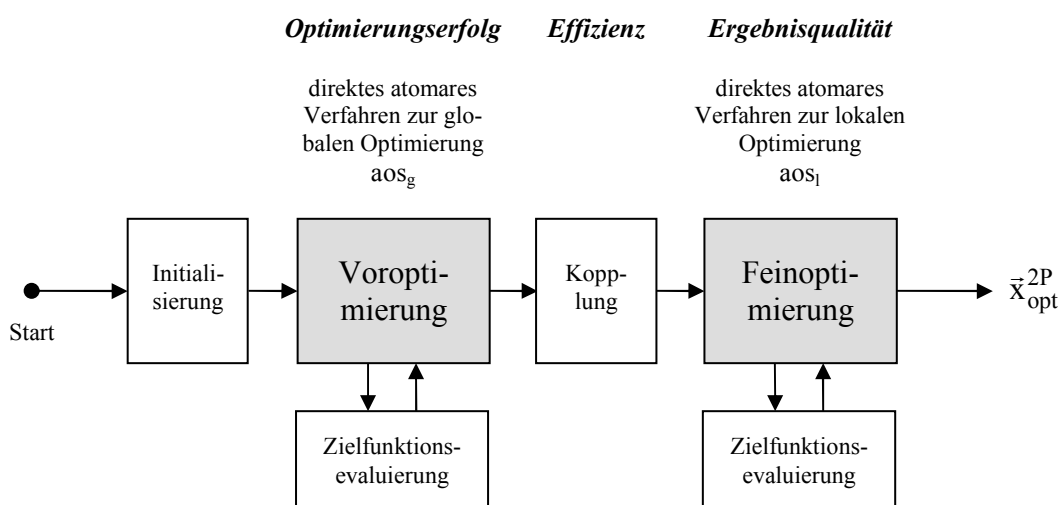


Abb. 5.1.2.1: Grundstruktur einer kombinierten 2-Phasen Optimierungsstrategie  $os_{2P}$

Bei dem 2-phasigen Optimierungsprozess aus Abb. 5.1.2.1 werden in jeder Phase unterschiedliche, voneinander klar abgegrenzte Zielsetzungen verfolgt. Das Hauptziel der Voroptimierung besteht darin, den Optimierungserfolg (Wahrscheinlichkeit für das Auffinden einer global-optimalen Lösung) zu maximieren. Aufgabe der Feinoptimierung ist es dagegen, optimale Lösungen möglichst exakt zu lokalisieren und damit für eine hohe Ergebnisqualität zu sorgen. Ausschlaggebend für die Effizienz des Gesamtverfahrens ist die erfolgreiche Bearbeitung folgender Teilproblematiken:

- Auswahl geeigneter direkter Optimierungsverfahren für die Vor- und Feinoptimierung

Zur Voroptimierung eignen sich beispielsweise Genetische Algorithmen und das Simulated Annealing, bei denen zur globalen Suche probabilistische, auf Prinzipien der Natur basierende Suchoperatoren eingesetzt werden. Als Feinoptimierungsverfahren bietet sich die Mustersuche von Hooke und Jeeves [HoJe61] an, bei der es sich um ein sehr effizientes und weit verbreitetes lokales Hill-Climbing Verfahren handelt. Die Richtung abnehmender (bzw. zunehmender) Zielfunktionswerte wird hier nach rein deterministischen Regeln ohne Zuhilfenahme von Gradientenberechnungen bestimmt.



- Parametrisierung des Vorooptimierungsverfahrens

Bei den zur Vorooptimierung eingesetzten globalen Optimierungsverfahren sind die Kontrollparameter so einzustellen, dass die explorativen Eigenschaften betont und vorzeitige Konvergenz gegen eine sub-optimale Region des Lösungsraums vermieden wird.

- "Rechtzeitiges" Umschalten von Vor- auf Feinoptimierung

Um Effizienzverluste zu vermeiden, sollte möglichst schnell nach Erreichen einer global-optimalen Region von Vor- auf Feinoptimierung umgeschaltet werden. Da kein einfach zu berechnendes Kriterium existiert, das anzeigt, ob bereits eine global-optimale Region aufgefunden wurde oder nicht, werden zur Lösung des Umschaltproblems Heuristiken eingesetzt.

- Automatische Berechnung einer vorteilhaften Kontrollparametereinstellung für das Feinoptimierungsverfahren basierend auf dem Verlauf und dem Ergebnis der Vorooptimierung

Diese Berechnung fällt in den Aufgabenbereich der zwischen Vor- und Feinoptimierung gelagerten Kopplungsschnittstelle, die zu diesem Zweck Methoden bereitstellt, die aus dem Vorooptimierungsergebnis und sämtlichen während des Vorooptimierungsprozesses erzeugten Lösungen (Vorooptimierungstrajektorie) automatisch eine möglichst günstige Kontrollparametereinstellung für das Feinoptimierungsverfahren ableiten. Dabei handelt es sich um eine geeignete Startlösung<sup>4</sup> sowie um geeignete Anfangsschrittweiten, die das Erreichen der global-optimalen Lösung innerhalb möglichst weniger Iterationen erlauben.

Eine ausführliche Beschreibung verschiedener Lösungsansätze zu den oben genannten Teilproblematiken sowie weitere Informationen und Hinweise zur algorithmischen Realisierung einer kombinierten 2-Phasen Optimierungsstrategie finden sich in [Syrj97].

### 5.1.3 Mehrstufige Optimierung

Die kombinierte 2-Phasen Optimierung hat sich im praktischen Einsatz als sehr leistungsfähig und effektiv erwiesen [SySz95a], [SySz95b]. Ihre vorteilhaften Eigenschaften stellen die Grundvoraussetzung zur Realisierung so genannter mehrstufiger Optimierungsverfahren [Syrj97] dar, deren Zielsetzung in der systematischen Ermittlung der markantesten Extremstellen eines vorgegebenen Optimierungsproblems besteht. Abb. 5.1.3.1 zeigt die algorithmische Grundstruktur eines mehrstufigen Optimierungsverfahrens, dessen Vorgehensweise im Wesentlichen auf der mehrfachen Hintereinanderausführung einer kombinierten 2-Phasen Strategie basiert.

---

<sup>4</sup> hier bietet es sich an, die beste während der Vorooptimierung ermittelte Lösung zu wählen

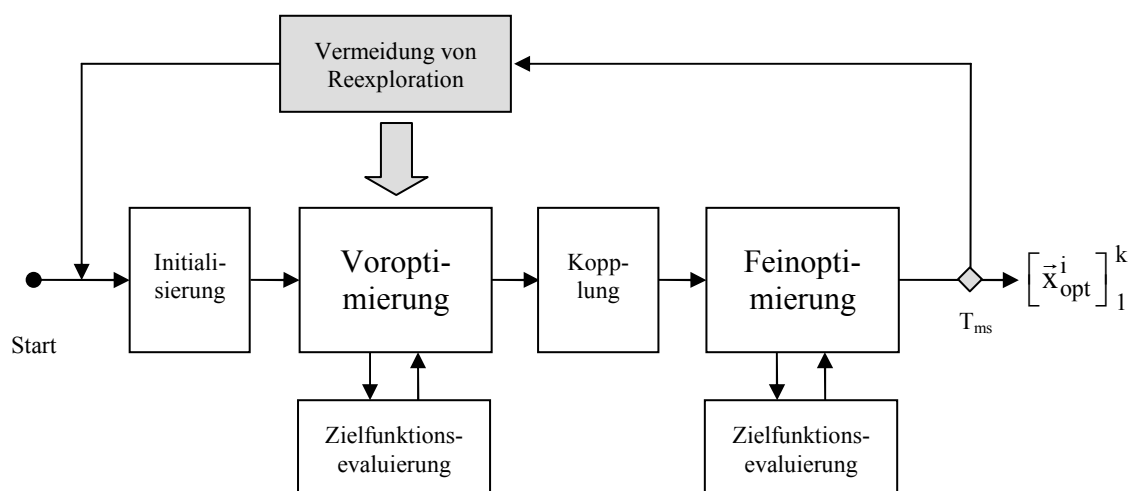


Abb. 5.1.3.1: Grundstruktur einer mehrstufigen Optimierungsstrategie  $os_{ms}$

Zentraler Bestandteil einer mehrstufigen Optimierungsstrategie  $os_{ms}$  ist eine kombinierte 2-Phasen Strategie  $os_{2p}$ , die in einen äußeren Iterationsprozess eingebettet ist, dessen Endergebnis eine Folge von optimalen Lösungen  $[\bar{x}_{opt}^i]_1^k$ ,  $k \in \mathbb{N}$  darstellt. Ein Iterationsschritt eines mehrstufigen Optimierungsverfahrens wird im Folgenden als Optimierungsstufe bezeichnet. Der mehrstufige Optimierungsprozess wird abgebrochen, wenn die Terminierungsbedingung  $T_{ms}$  erfüllt ist. Zur Terminierung der mehrstufigen Optimierung bieten sich beispielsweise folgende Abbruchkriterien an:

- Terminiere nach Erreichen einer bestimmten Anzahl von Optimierungsstufen.
- Terminiere, falls über eine bestimmte Anzahl von Optimierungsstufen kein Optimierungsfortschritt (Lösungsverbesserung) erzielt werden konnte.

Ein weiterer integraler Bestandteil der mehrstufigen Optimierung ist die Methode zur Vermeidung von Reexploration (VR), deren Aufgabe darin besteht, das wiederholte Auffinden bereits aufgefundener optimaler Lösungen in nachfolgenden Optimierungsstufen zu erschweren. Eine Möglichkeit zur Realisierung von VR besteht darin, jeder möglichen Lösung im Suchraum neben dem Zielfunktionswert einen Attraktivitätswert zuzuordnen. Auf diese Weise ist es möglich, bereits explorierte Regionen des Lösungsraums für das Voroptimierungsverfahren unattraktiv zu machen und die Voroptimierung in noch unerforschte Suchraumregionen zu lenken. Attraktivitätswerte lassen sich auf einfache Weise durch folgende Funktion berechnen:

$$aw(\bar{x}) = \prod_{i=1}^k \left[ 1 - (1 + \alpha \cdot d_i)^{-\beta} \right],$$

mit  $d_i = \sqrt{(\bar{x} - \bar{x}_{opt}^i)^2}$ ;  $\alpha, \beta$ : Skalierungsfaktoren;

$k$ : Anzahl bereits aufgefundener Optimumpunkte.

Das Beispiel aus Abb. 5.1.3.2 zeigt eine Attraktivitätsfunktion für die dritte Optimierungsstufe, wobei angenommen wird, dass in den beiden vorangegangenen Optimierungsstufen die beiden Optimumpunkte  $(3\pi/2, \pi/2)$  und  $(\pi/2, 3\pi/2)$  aufgefunden wurden ( $k=2$ ).

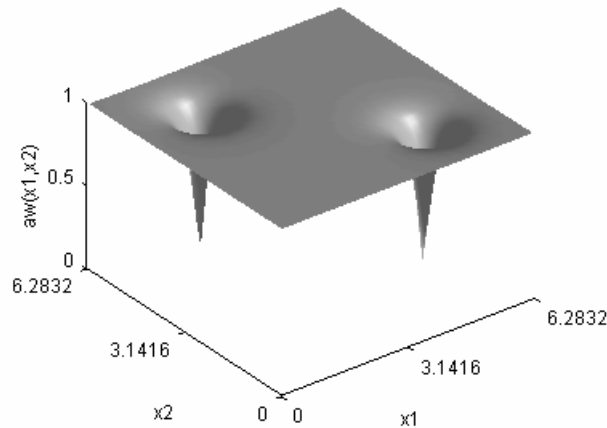


Abb. 5.1.3.2: Beispiel einer in der dritten Optimierungsstufe gültigen Attraktivitätsfunktion

Da diese Punkte nicht noch einmal lokalisiert werden sollen, wird ihnen der kleinstmögliche Attraktivitätswert (Null) zugewiesen. Davon ausgehend, wird die Attraktivität der benachbarten Punkte kontinuierlich der Maximalattraktivität 1 angenähert.

Die um die beiden Optimumpunkte gelagerten Attraktivitätspotentiale gleichen umgekehrten, rotationssymmetrischen Kegeln, deren Größe und Form durch die beiden Skalierungsfaktoren  $\alpha, \beta \in \mathbb{R}_0^+$  (hier  $\alpha=1$ ,  $\beta=6$ ) beliebig variiert werden kann. Durch die oben dargestellte Attraktivitätsfunktion wird die Attraktivität der Punkte im gesamten Lösungsraum beeinflusst, wobei jedoch der Einfluss mit wachsendem Abstand zu den bekannten Optimumpunkten mehr und mehr abnimmt. Für sehr weit entfernte Punkte ist die Auswirkung eines solchen "globalen" Attraktivitätspotentials vernachlässigbar gering. Eine detaillierte Beschreibung der Methode zur Vermeidung von Reexploration findet sich in [Syrj97].

Durch die mehrstufige Optimierung, bei der jede weitere Optimierungsstufe einen monotonen Anstieg des Optimierungserfolgs (Wahrscheinlichkeit des Auffindens einer global-optimalen Lösung) nach sich zieht, lassen sich auch sehr schwierige Problemstellungen bearbeiten, die mit einer kombinierten 2-Phasen Strategie auf Anhieb nicht zu bewältigen wären. Die mehrstufige Optimierung stellt daher eine deutlich verbesserte und systematischere Methodologie zur globalen Optimierung dar, als die bisher gebräuchliche "einstufige" Vorgehensweise.

## 5.1.4 Spezielle Maßnahmen zur Reduktion des Optimierungsaufwands

Trotz der hohen Effizienz der kombinierten 2-Phasen Strategie ist die Anzahl der Zielfunktionsbewertungen, die sich durch deren Mehrfachausführung innerhalb des in Abb. 5.1.3.1 dargestellten mehrstufigen Optimierungsverfahrens ergibt, i.a. zu hoch, um auch komplexe Simu-

lationsmodelle mit sehr rechenintensiven Auswertungsprozessen in einem vernünftigen Zeitrahmen zu optimieren. Dazu sind wirkungsvolle Beschleunigungsmaßnahmen erforderlich, wobei grundsätzlich zwischen

- Maßnahmen zur Beschleunigung der Zielfunktionsevaluierung durch Verkürzung der Modellauswertungszeit und
- Maßnahmen zur Beschleunigung des direkten Optimierungsprozesses durch Einsparung von Zielfunktionsevaluierungen

unterschieden werden kann.

Eine Beschleunigung des Simulationsprozesses kann prinzipiell durch Einsatz schnellerer Hardware und/oder effizienterer Simulatoren erreicht werden, was hier jedoch nicht weiterverfolgt werden soll. Im Folgenden werden spezielle Methoden vorgestellt, die darauf abzielen, die in den vorangegangenen Abschnitten beschriebenen direkten Optimierungsverfahren durch Einsparung von Zielfunktionsevaluierungen weiter zu beschleunigen. Der Grundgedanke dieser Methoden besteht darin, bereits gewonnene Informationen über das Optimierungsproblem besser als bisher zur Aufwandsreduktion zu verwenden.

#### **5.1.4.1 Vermeidung von Reevaluation**

Eine sehr einfache und naheliegende Möglichkeit, um bei direkten Optimierungsverfahren Zielfunktionsauswertungen einzusparen, besteht darin, nochmalige Evaluationen von bereits erzeugten und ausgewerteten Lösungen zu vermeiden. Da es bei direkten Optimierungsverfahren sehr häufig vorkommt, dass Suchpunkte während des Optimierungsprozesses mehrfach erzeugt werden, können auf diese Weise sowohl bei Vor- als auch bei Feinoptimierungsverfahren beträchtliche Einsparungseffekte erreicht werden, ohne dabei einen Genauigkeitsverlust hinnehmen zu müssen. Das wiederholte Auswerten bereits erzeugter Suchpunkte lässt sich sehr einfach durch entsprechendes Durchsuchen der Optimierungstrajektorie vermeiden, die zu diesem Zweck abgespeichert werden muss.

#### **5.1.4.2 Mehrfachstart der Feinoptimierung**

Bei diesem Ansatz wird die Feinoptimierung nicht nur von der besten vom Voroptimierungsverfahren ermittelten Lösung aus gestartet, sondern auch an Punkten, die einen etwas schlechteren Zielfunktionswert aufweisen und relativ weit entfernt voneinander liegen. Um solche viel versprechenden Startpunkte zu ermitteln, wird zunächst einmal die Voroptimierungstrajektorie analysiert und ausgehend davon heuristisch entschieden, wie oft und an welchen Punkten die Feinoptimierung gestartet werden soll. Weitere Einzelheiten zur Terminierung der mehrfachen Feinoptimierung ( $T_{mF}$ ) finden sich in [Syrj97]. Bei hochgradig multimodalen Zielfunktionen lässt sich auf diese Weise die Anzahl der benötigten Voroptimierungsläufe in der Regel beträchtlich reduzieren. In Abb. 5.1.4.2.1 wurde das in Abb. 5.1.3.1 dargestellte mehrstufige Optimierungsverfahren um den Mehrfachstart der Feinoptimierung erweitert.

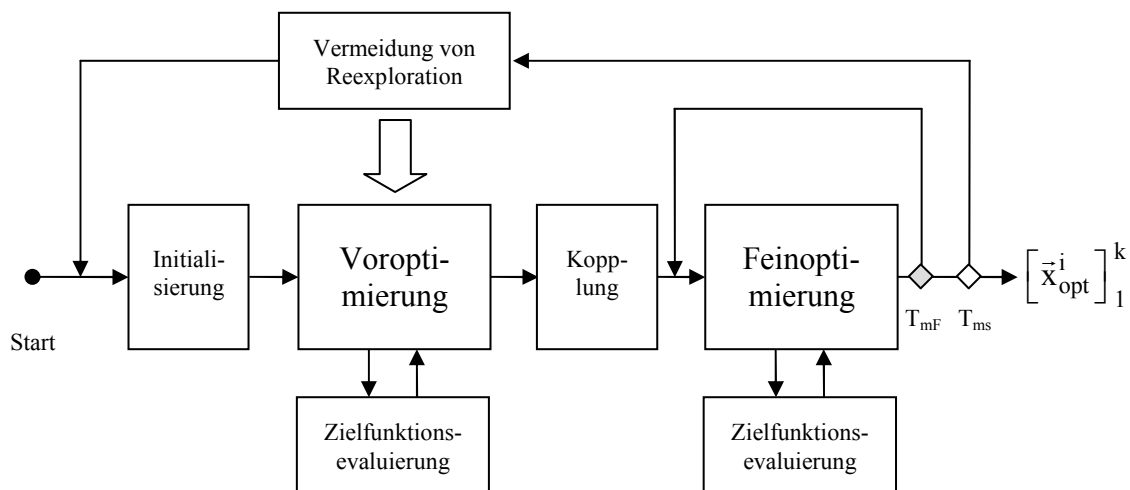


Abb. 5.1.4.2.1: Mehrfachstart der Feinoptimierung

### 5.1.4.3 Beschleunigung der Voroptimierung

Mit der Vermeidung von Reevaluation ist das bei effizienten Feinoptimierungsverfahren vorhandene Einsparungspotential bereits weitgehend ausgeschöpft. Bei der Voroptimierung dagegen bestehen, wie im Folgenden aufgezeigt wird, noch einige gute Möglichkeiten zur Aufwandseinsparung, wobei vor allem die weitgehende Unempfindlichkeit probabilistischer Voroptimierungsverfahren gegenüber ungenau berechneten Zielfunktionswerten ausgenutzt werden kann [GrFi85]. Grundsätzlich bieten sich zur Beschleunigung des Voroptimierungsprozesses folgende Vorgehensweisen an:

- Verkürzung der Simulationsdauer

Bei der Voroptimierung mit probabilistischen Suchverfahren kann aufgrund deren Unempfindlichkeit gegenüber ungenau berechneten Zielfunktionswerten eine etwas kürzere Simulationsdauer als bei der Feinoptimierung angesetzt werden. Der mit der Aufwandseinsparung verbundene Genauigkeitsverlust sollte jedoch im Vorfeld der Optimierung durch geeignete statistische Methoden (Konfidenzintervalle) abgeschätzt werden.

- Zielfunktionsapproximation

Der Grundgedanke dieses Ansatzes besteht darin, die Zielfunktion in häufig besuchten Regionen des Lösungsraums anhand der dort bereits bekannten Zielfunktionswerte zu approximieren. Auch durch diese Maßnahme lässt sich der Optimierungsprozess in vielen Fällen beträchtlich verkürzen, da Zielfunktionsapproximationen sehr viel weniger Rechenaufwand erfordern als Zielfunktionsevaluierungen durch Simulation.

Der letztgenannte Lösungsansatz, der eine sehr interessante und viel versprechende Möglichkeit zur Aufwandsreduktion darstellt, wird im Folgenden etwas detaillierter beschrieben. Die prinzipielle Vorgehensweise bei der Aufwandsreduktion durch Zielfunktionsapproximation ist in Abb. 5.1.4.3.1 dargestellt. Bei dem neu hinzugekommenen Baustein handelt es sich um ei-

nen Funktionsapproximator, der zwischen den Voroptimierungsprozess und den Prozess der Zielfunktionsevaluierung geschaltet ist.

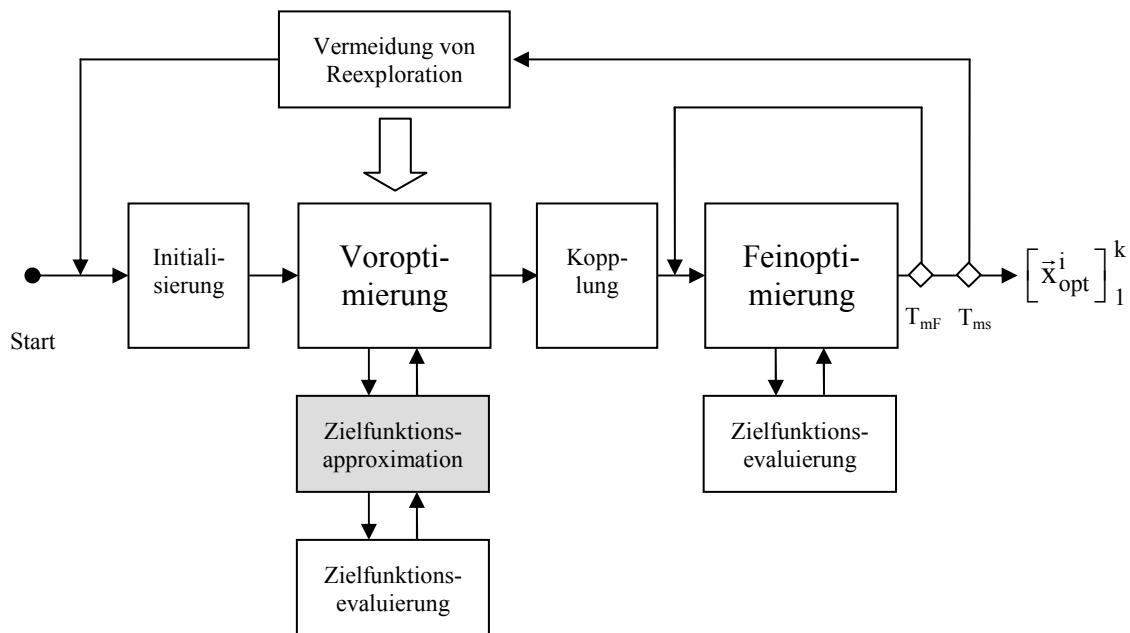


Abb. 5.1.4.3.1: Beschleunigung der Voroptimierung durch Zielfunktionsapproximation

Ein vom Optimierungsprozess erzeugter Parametervektor  $\vec{x}$  wird nicht, wie sonst üblich, sofort an die Zielfunktionsevaluierung weitergegeben, sondern zunächst einmal an den Funktionsapproximator, der anhand eines Approximationskriteriums entscheidet, ob eine Approximation durchgeführt werden kann oder nicht. Zur Entscheidungsfindung wird dabei auf die bereits berechneten und als Stützstellen abgespeicherten Zielfunktionswerte zurückgegriffen.

Falls keine Approximation vorgenommen werden kann, wird der Parametervektor  $\vec{x}$  direkt an die Zielfunktionsevaluierung weitergereicht, die daraufhin dessen Zielfunktionswert  $F(\vec{x})$  durch Auswertung des zu optimierenden Modells berechnet. Anschließend wird das Tupel  $(\vec{x}, F(\vec{x}))$  dem Funktionsapproximator als eine weitere Stützstelle übergeben und der Zielfunktionswert  $F(\vec{x})$  an den Voroptimierungsprozess weitergeleitet.

Mit zunehmendem Voranschreiten des Optimierungsprozesses nimmt die Anzahl der Stützstellen zu, womit die Wahrscheinlichkeit immer größer wird, dass eine Approximation durchgeführt werden kann. Diese wird durch einen Approximationsmechanismus vorgenommen, der, basierend auf den bereits bekannten Stützstellen, einen angenäherten Zielfunktionswert  $\tilde{F}(\vec{x})$  berechnet. Ob der approximierter Wert  $\tilde{F}(\vec{x})$  angenommen oder verworfen wird, entscheidet ein Akzeptanzkriterium. Wird  $\tilde{F}(\vec{x})$  nicht akzeptiert, muss eine weitere Zielfunktionsevaluierung durchgeführt werden. Der berechnete Zielfunktionswert wird dann sowohl dem Voroptimierungsprozess als auch dem Funktionsapproximator übergeben, der auf diese Weise "nachtrainiert" wird. Falls  $\tilde{F}(\vec{x})$  jedoch akzeptiert wird, kann eine Zielfunktionsevaluierung eingespart werden, da dann der approximierter Wert  $\tilde{F}(\vec{x})$  anstelle des eigentlichen Zielfunktionswerts

$F(\bar{x})$  an den Voroptimierungsprozess weitergereicht wird. Insbesondere dann, wenn die Voroptimierungsstrategie gegen eine bestimmte Region des Lösungsraums konvergiert, lassen sich durch Zielfunktionsapproximation hohe Einsparungseffekte erzielen. Der damit verbundene Genauigkeitsverlust kann i.a. toleriert werden, da die Aufgabe der Voroptimierung nicht darin besteht, eine optimale Lösung möglichst exakt zu lokalisieren, sondern lediglich eine Annäherung zu finden, von der aus die Feinoptimierung die optimale Lösung erreichen kann. Als besonders wirkungsvoll hat sich die Beschleunigung der Voroptimierung durch Zielfunktionsapproximation bei der mehrstufigen Optimierung herausgestellt. Hier lassen sich vor allem in höheren Optimierungsstufen große Einsparungseffekte erzielen, da in diesem fortgeschrittenen Optimierungsstadium bereits viele Regionen des Lösungsraums exploriert wurden und dementsprechend viele Stützstellen existieren, die vom Zielfunktionsapproximator genutzt werden können.

Abschließend sei angemerkt, dass dem Voroptimierungsprozess der Funktionsapproximator grundsätzlich verborgen bleibt, d.h. der Voroptimierung ist es nicht bekannt, ob der gelieferte Zielfunktionswert tatsächlich evaluiert oder nur approximiert wurde. In [SSBH96] und [Syrj97] werden verschiedene Möglichkeiten zur Realisierung eines Funktionsapproximators vorgestellt und erläutert.

### 5.1.5 Behandlung irreführender Probleme

Auch wenn die in Abschnitt 2.3.2 formulierten vereinfachenden Annahmen eingehalten werden, kann die Zielfunktionsoberfläche durchaus Strukturen aufweisen, bei denen einige naturanaloge Voroptimierungsverfahren große Schwierigkeiten haben, in den Einzugsbereich eines globalen Optimums zu gelangen. Zwei Beispiele solcher schwer zu optimierender Funktionen sind in Abb. 5.1.5.1 dargestellt. Hier ist die Ergebnisoberfläche so strukturiert, dass das Auffinden des globalen Maximums evolutionären Algorithmen wie beispielsweise Genetische Algorithmen oder Evolutionsstrategien große Schwierigkeiten bereitet. Diese Verfahren maximieren die in Abb. 5.1.5.1 dargestellten Probleme bei gleichem Optimierungsaufwand im Durchschnitt weniger erfolgreich als die reine Zufallssuche.

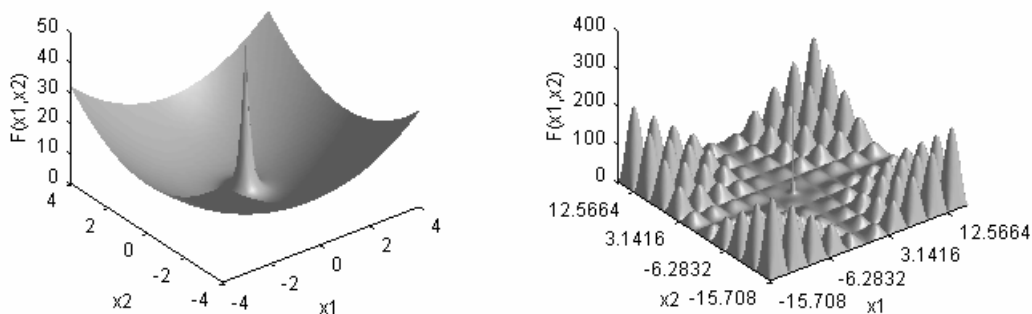


Abb. 5.1.5.1: Für evolutionäre Algorithmen irreführende Probleme

Erklären lässt sich dieses schlechte Abschneiden folgendermaßen: Da der Einzugsbereich der global-maximalen Lösung nur einen sehr kleinen Teil des zu explorierenden Lösungsraums

darstellt, ist die Wahrscheinlichkeit sehr gering, dass Punkte aus diesem im Zentrum des Lösungsraums gelegenen Bereichs bereits in der zufällig erzeugten Startpopulation enthalten sind. Bleibt der Einzugsbereich unentdeckt, konvergieren evolutionäre Algorithmen aufgrund des hohen Selektionsdrucks i.a. sehr schnell gegen eine der vier lokal-maximalen Lösungen, die sich am äußersten Rand des Lösungsraums befinden. Mit zunehmender Konvergenz gegen die äußeren Regionen wird es immer unwahrscheinlicher, durch zufällige Mutation bzw. günstiges Crossover in den gerade entgegengesetzt zur Optimierungsrichtung liegenden Einzugsbereich der global-maximalen Lösung zu gelangen. Der Erfolg evolutionärer Algorithmen hängt bei den in Abb. 5.1.5.1 dargestellten Problemen also entscheidend davon ab, dass bereits beim Erzeugen der Startpopulation der Einzugsbereich der global-maximalen Lösung entdeckt wird. Bei der reinen Zufallssuche ist dies nicht der Fall, da hier die Suchpunkte gleichverteilt und völlig unabhängig von der Vorgeschichte der Optimierung erzeugt werden.

Problemstellungen, bei denen ein Optimierungsverfahren bei gleichem Optimierungsaufwand im Durchschnitt einen schlechteren Optimierungserfolg aufweist als die reine Monte-Carlo Suche, werden im Folgenden als für das Optimierungsverfahren irreführende Probleme bezeichnet. Eine naheliegende Möglichkeit, um die Erfolgchancen bei irreführenden Problemen zu erhöhen, besteht darin, bei der mehrstufigen Optimierung nicht nur genau ein Voroptimierungsverfahren zu verwenden, sondern wie in Abb. 5.1.5.2 dargestellt, mehrere im zyklischen Wechsel.

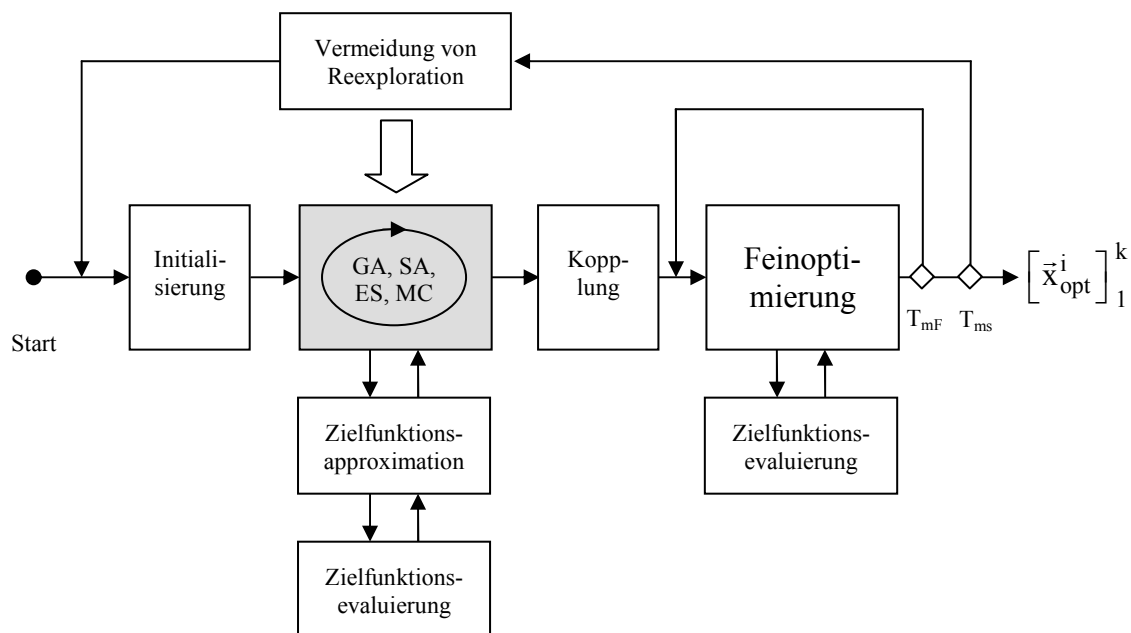


Abb. 5.1.5.2: Anwendung verschiedener Voroptimierungsstrategien im zyklischen Wechsel

Dabei sollten globale Optimierungsverfahren mit möglichst unterschiedlichen Sucheigenschaften ausgewählt werden. Als sehr geeignet hat sich eine Zusammenstellung aus Monte-Carlo Suche (MC), Simulated Annealing (SA) sowie Vertretern evolutionärer Algorithmen (GA, ES) erwiesen. Durch die Vielzahl unterschiedlicher Optimierungsprinzipien kann sich bei gutartigen Problemen die Gesamtleistung des Verfahrens gegenüber dem herkömmlichen mehrstufigen Verfahren aus Abb. 5.1.3.1 zwar etwas verschlechtern, insgesamt jedoch wird das



Risiko vermindert, ein bestimmtes Problem mit einem dafür ungeeigneten Voroptimierungsalgorithmus zu bearbeiten.

Mit der reinen Zufallssuche lassen sich die in Abb. 5.1.5.1 dargestellten Optimierungsprobleme bei gleichem Optimierungsaufwand zwar im Mittel erfolgreicher lösen als mit evolutionären Algorithmen, der zu beobachtende Erfolgsunterschied ist jedoch nicht sonderlich groß und nimmt zudem mit wachsender Problemdimension ab. Die reine Zufallssuche ist zur Lösung irreführender Probleme daher auch nicht sehr empfehlenswert. Als weitaus erfolgreicher hat sich dagegen eine geringfügig modifizierte kombinierte 2-Phasen Strategie herausgestellt, bei der die Suchrichtung der Voroptimierung einfach umgedreht wurde (inverse Voroptimierung). Bei der Maximierung der in Abb. 5.1.5.1 dargestellten Optimierungsprobleme würde das bedeuten, dass bei der Voroptimierung nicht wie sonst üblich maximiert, sondern minimiert wird. Da die global-minimalen Lösungen bei diesen Optimierungsproblemen ringförmig um den Einzugsbereich der global-maximalen Lösung angeordnet sind, ist die Wahrscheinlichkeit sehr hoch, durch eine Vorminimierung in diesen Einzugsbereich oder zumindest in dessen unmittelbare Nähe zu gelangen. Um ganz sicher zu gehen, den Einzugsbereich nicht knapp verfehlt zu haben, bietet es sich an, nach Abschluss der Vorminimierung in einem eng begrenzten Bereich um die gefundene minimale Lösung noch einmal eine Maximierung vorzunehmen. Diese braucht nicht sehr aufwändig zu sein. Einige Dutzend Zielfunktionsevaluierungen sind zu diesem Zweck in der Regel völlig ausreichend.

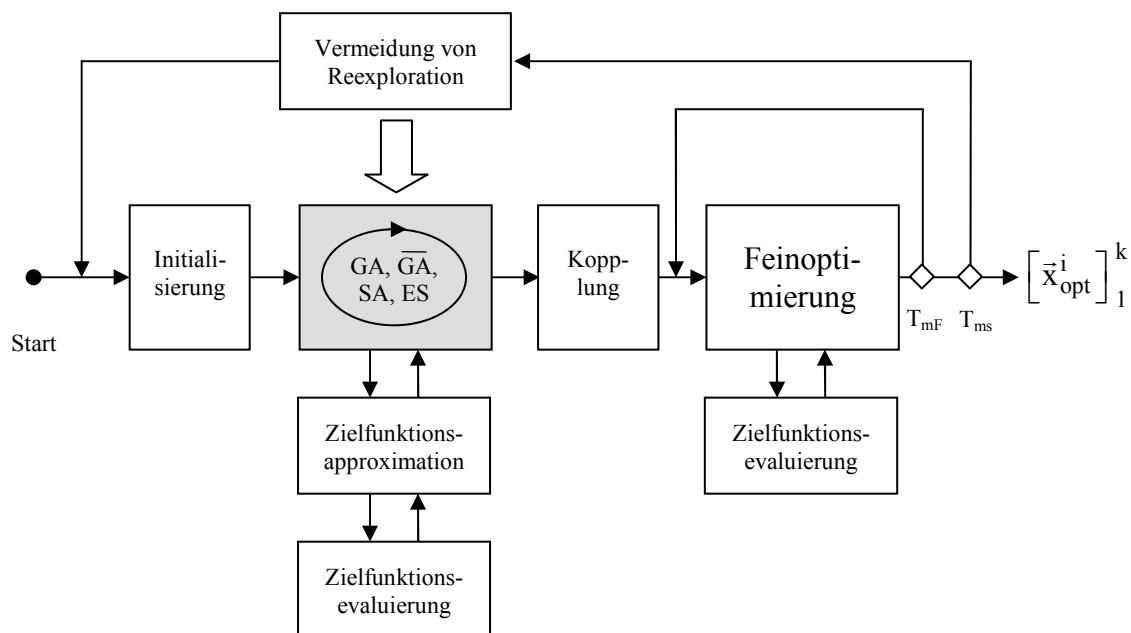


Abb. 5.1.5.3: Lösung irreführender Probleme durch inverse Voroptimierung

In Abb. 5.1.5.3 ist das in Abb. 5.1.5.2 dargestellte Optimierungsverfahren um die Methode der inversen Voroptimierung erweitert worden. Hier wird in der ersten Optimierungsstufe wie gehabt mit einem Genetischen Algorithmus vormaximiert. Im Anschluss daran wird der Genetische Algorithmus zur Vorminimierung verwendet, um bei dem vorliegenden Optimierungsproblem gezielt nach schwer auffindbaren Maximalstellen zu suchen. In den nächsten beiden Optimierungsstufen wird mit den Evolutionsstrategien und dem Simulated Annealing wieder

vormaximiert, wonach der Zyklus von neuem beginnt. Diese Vorgehensweise bei der mehrstufigen Optimierung, in regelmäßigen Abständen auch Stufen mit inverser Voroptimierung durchzuführen, hat sich als eine sehr elegante und effektive Möglichkeit erwiesen, um zu verhindern, dass irreführende Probleme als solche unerkannt bleiben.

### 5.1.6 Umgang mit stochastischem Rauschen

Bei stochastischen Simulationsmodellen sind die berechneten Ausgabegrößen (Zielfunktionswerte) mit einem Rauschen überlagert, dessen Intensität stark von der jeweils angesetzten Simulationsdauer abhängt. Verfahren zur Modelloptimierung sollten aus diesem Grund möglichst robust gegenüber ungenau berechneten Zielfunktionswerten sein. Bei vielen direkten Optimierungsverfahren, insbesondere bei globalen probabilistischen Suchverfahren wie Evolutionären Algorithmen oder dem Simulated Annealing trifft dies auch weitgehend zu, was jedoch nicht dazu verleiten sollte, stochastisches Rauschen bei der Optimierung simulationsbasierter Zielfunktionen gänzlich zu vernachlässigen. Bevor eine Modelloptimierung durchgeführt wird, sollte daher stets die Intensität des stochastischen Rauschens abgeschätzt werden, um diese bei der Auswahl und Parametrisierung der Optimierungsverfahren<sup>5</sup> sowie bei der Interpretation der Optimierungsergebnisse entsprechend berücksichtigen zu können. Bei zu starkem Rauschen muss gegebenenfalls eine Verlängerung der Simulationsdauer in Betracht gezogen werden. Weitere Einzelheiten zur Problematik des stochastischen Rauschens finden sich in [Syrj97].

## 5.2 Realisierung der Optimierungskomponente

Die in den vorangegangenen Abschnitten vorgestellten direkten Optimierungsverfahren bilden den Kern der in Abb. 5.2.1 dargestellten Optimierungskomponente. Um diese möglichst flexibel und plattformübergreifend als Spezialkomponente zur Erweiterung der Funktionalität von Modellierungswerkzeugen einsetzen zu können, wurde zur Implementierung die Programmiersprache Java verwendet. Neben den direkten Optimierungsverfahren, die in Form einer modularen Bausteinbibliothek angeboten werden, gehören zur Optimierungskomponente drei weitere wichtige Bestandteile. Dabei handelt es sich um

- eine graphische Benutzeroberfläche, deren Aufgabe darin besteht, verschiedenen Anwendergruppen ein komfortables auf deren jeweilige Bedürfnisse zugeschnittenes Arbeiten mit den implementierten Optimierungsverfahren zu ermöglichen,
- eine flexible Schnittstelle zur Ankopplung an alle Arten von Modellauswertungssoftware,
- ein separates Modul zur statistischen Aufbereitung des bei umfangreichen Optimierungsexperimenten erzeugten Datenmaterials.

---

<sup>5</sup> Bei lokalen Hill-Climbing Verfahren beispielsweise sollten bei starkem stochastischen Rauschen die Anfangsschrittweiten deutlich größer als im unverrauschten Fall eingestellt werden.

Die oben genannten Bestandteile der Optimierungskomponente werden im Folgenden näher erläutert.

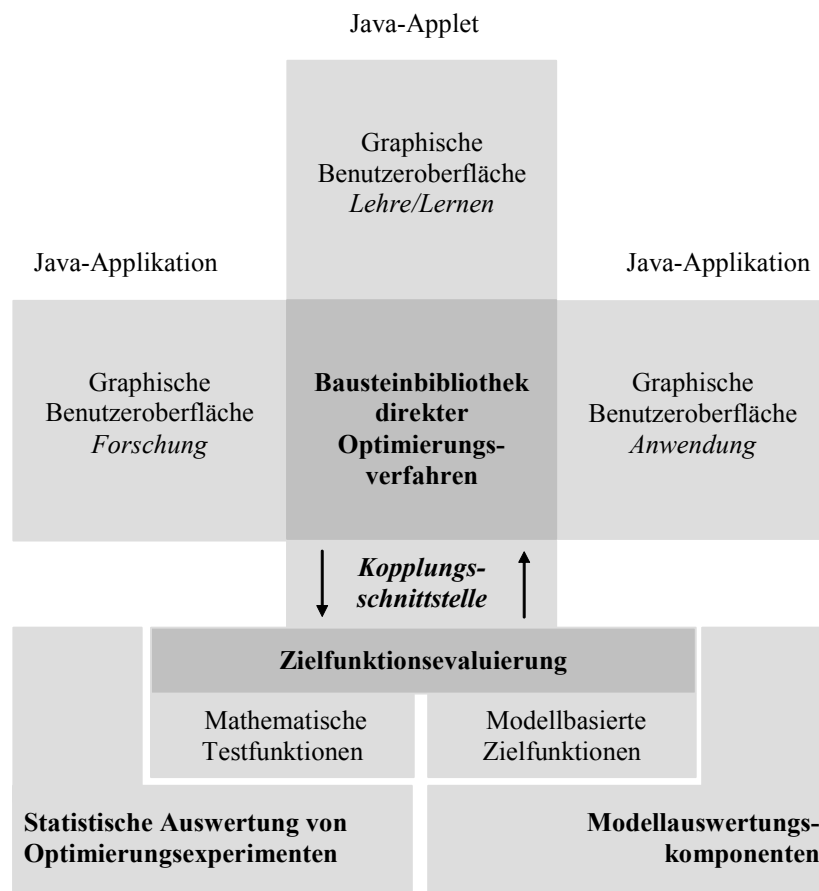


Abb. 5.2.1: Software-Architektur der Optimierungskomponente

Um einen möglichst großen Benutzerkreis zu erschließen, wurde die Graphikoberfläche in drei unterschiedlichen Varianten realisiert, die jeweils auf die speziellen Bedürfnisse einer bestimmten Zielgruppe zugeschnitten sind. Bei den berücksichtigten Zielgruppen handelt es sich um

1. Wissenschaftler, die mit den implementierten Optimierungsverfahren experimentieren und diese weiterentwickeln wollen. Zur Durchführung von Optimierungsexperimenten wird von der auf Wissenschaftler zugeschnittenen Oberflächenversion eine große Auswahl an mathematischen Testfunktionen angeboten. Einige der zur Verfügung gestellten Funktionen sind in Abschnitt 5.3.2 graphisch dargestellt. Der vorhandene Satz lässt sich auf einfache Weise um weitere Funktionen erweitern. Mathematische Testfunktionen haben gegenüber simulationsbasierten Zielfunktionen den Vorteil, dass sie um Größenordnungen schneller ausgewertet werden können. Darüber hinaus lassen sie sich bezüglich ihrer Oberflächencharakteristik beliebig gestalten. Sie sind daher hervorragend zur umfassenden empirischen Leistungsanalyse der implementierten Optimierungsverfahren geeignet. Zu diesem Zweck wird die Möglichkeit geboten, groß angelegte Optimierungsexperimente durchzuführen, bei denen ein Verfahren mehrfach mit jeweils unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators auf eine bestimmte Problemstellung angewendet wird. Zur statisti-

schen Auswertung der dabei anfallenden Daten kann auf ein speziell zu diesem Zweck entwickeltes Analysewerkzeug zurückgegriffen werden. Insgesamt wird dadurch eine umfassende empirische Leistungsanalyse der implementierten Optimierungsverfahren sowie deren Kalibrierung hinsichtlich geeigneter Kontrollparametereinstellungen ermöglicht.

2. Studenten bzw. auf dem Gebiet der direkten Optimierung unerfahrene Anwender, die den Umgang mit den implementierten Optimierungsverfahren erlernen wollen. Um tiefgehende Einblicke in die Arbeitsweise der implementierten atomaren globalen und lokalen Optimierungsverfahren zu bekommen, ermöglicht diese Oberflächenversion eine graphische Visualisierung des Optimierungsablaufs. Der Benutzer kann die Verfahren beliebig parametrisieren und auf eine Reihe zweidimensionaler mathematischer Testprobleme anwenden. Um den Zugriff auf diese Version möglichst komfortabel zu gestalten, wurde sie in Form eines Java-Applets realisiert und im WWW unter [http://goethe.ira.uka.de/~syrjakow/anim\\_env3/start\\_environment.html](http://goethe.ira.uka.de/~syrjakow/anim_env3/start_environment.html) zur Verfügung gestellt. Eine ausführliche Beschreibung dieses Java-Applets findet sich in Abschnitt 6.2.
3. Erfahrene Anwender, welche die Verfahren ihrer eigentlichen Bestimmung zuführen wollen, nämlich der Optimierung komplexer Simulationsmodelle. Zu diesem Zweck erlaubt die Graphikoberfläche, Modelloptimierungsprobleme zu spezifizieren, den Prozess der Modelloptimierung anzustoßen und kontrolliert ablaufen zu lassen sowie im Anschluss daran die erzeugten Ergebnisdaten einzusehen. Es werden verschiedene Benutzungsmodi angeboten, um einem möglichst breiten Anwenderspektrum mit unterschiedlichem Vorwissen gerecht zu werden. Unerfahrenen Benutzern wird beispielsweise die Möglichkeit geboten, sich durch einen persönlichen Assistenten (Wizard) unterstützen zu lassen. Dieser gibt Hilfestellung bei der Spezifikation des Optimierungsproblems, bei der Auswahl eines dazu geeigneten Optimierungsverfahrens sowie dessen Parametrisierung. Darüber hinaus ist es möglich, den ablaufenden Optimierungsprozess in graphisch aufbereiteter Form zu verfolgen und gegebenenfalls interaktiv zu beeinflussen.

Den algorithmischen Kern der Optimierungskomponente bildet eine erweiterbare modulare Bausteinbibliothek direkter Optimierungsmethoden. In seiner jetzigen Ausbaustufe werden dem Anwender drei atomare globale Optimierungsverfahren (Genetische Algorithmen, Simulated Annealing und reine Monte-Carlo Suche) sowie ein atomares lokales Optimierungsverfahren (Mustersuche) zur Verfügung gestellt. Daraus lassen sich verschiedene Realisierungsalternativen kombinierter 2-Phasen Strategien zusammenstellen, die wiederum zur mehrstufigen Optimierung eingesetzt werden können. Die in den Abschnitten 5.1.4 und 5.1.5 beschriebenen Erweiterungen zur Reduktion des Optimierungsaufwands sowie zur Erkennung irreführender Probleme werden ebenfalls unterstützt.

Die implementierten direkten Optimierungsverfahren sind grundsätzlich universell anwendbar und ermöglichen es dem Benutzer, nicht nur Simulationsmodelle zu optimieren, sondern auch analytische Modelle oder einfach nur mathematische Funktionen. Zur Ankopplung an alle Arten von (Modell-)Auswertungssoftware wurde die Optimierungskomponente mit einer flexiblen Kopplungsschnittstelle ausgestattet. Eine genaue Auflistung der über diese Schnittstelle ausgetauschten Daten findet sich in [Syrj97]. Um Sprach- und Plattformunabhängigkeit zu gewährleisten, wurde zur Realisierung der Kopplungsschnittstelle der in Abschnitt 2.5.3 beschriebene Middleware-Standard CORBA verwendet.

Den vierten Bestandteil der Optimierungskomponente stellt ein separates Modul zur statistischen Aufbereitung des bei umfangreichen Optimierungsexperimenten erzeugten Datenmateri-

als dar. Hauptintention dieses Moduls ist es, die empirische Leistungsanalyse der implementierten Optimierungsverfahren zu unterstützen. Dazu werden aus den abgespeicherten Ergebnisdatensätzen durchgeführter Optimierungsexperimente Leistungskenngrößen zur Quantifizierung der Güte des betreffenden Optimierungsverfahrens berechnet. Eine Zusammenstellung dieser Kenngrößen befindet sich im folgenden Abschnitt. Darüber hinaus werden Ergebnisse einiger umfangreicher Optimierungsexperimente zu verschiedenen mathematischen Problemstellungen vorgestellt und erläutert.

## 5.3 Bewertung der Optimierungsverfahren

Den Schwerpunkt der nun folgenden Betrachtungen bildet eine empirische Leistungsanalyse der entwickelten direkten Optimierungsverfahren. Empirische Untersuchungen sind zwar sehr rechenaufwändig, häufig jedoch die einzige Möglichkeit, um das Leistungsverhalten direkter Optimierungsverfahren zu erschließen, da zur erfolgreichen Durchführung theoretischer Analysen die zur Verfügung stehende Methodologie aus heutiger Sicht noch nicht ausreicht. Bei den in Abschnitt 5.1 beschriebenen direkten Optimierungsverfahren erschweren vor allem die vielfältigen Einflussgrößen, die probabilistischen Suchoperatoren sowie der hybride Verfahrensaufbau fundierte theoretische Leistungsuntersuchungen.

### 5.3.1 Leistungskenngrößen

Als Grundlage zur Charakterisierung der Leistung und Effizienz direkter Optimierungsverfahren werden zunächst folgende Basiskenngrößen eingeführt:

- Optimierungsaufwand  $o_A$

Der Optimierungsaufwand gibt die Anzahl der während des Optimierungsprozesses durchgeführten Zielfunktionsevaluierungen an.

- Optimierungstrajektorie  $o_T$

Die Optimierungstrajektorie stellt die Menge aller während des Optimierungsprozesses erzeugten Lösungsalternativen einschließlich ihrer Zielfunktionswerte dar.

- Optimierungsergebnis  $o_E$

Das Optimierungsergebnis ist eine ein- oder mehrelementige Menge von Lösungen, die den besten während des Optimierungsprozesses ermittelten Zielfunktionswert aufweisen.

Die oben aufgeführten Kenngrößen lassen sich grundsätzlich bei jedem Optimierungslauf auf einfache Weise empirisch bestimmen. Experimente zur Ermittlung dieser Größen sind jedoch nur dann aussagekräftig und unter gleichen Randbedingungen wiederholbar, wenn das bearbeitete Optimierungsproblem, das verwendete Optimierungsverfahren, dessen Kontrollparametereinstellung und die Startlösung(en) angegeben werden. Bei probabilistischen Optimierungsverfahren, bei denen ein Optimierungslauf ein Zufallsexperiment mit ungewissem Ausgang

darstellt, ist außerdem die verwendete Initialisierung des Pseudo-Zufallszahlengenerators ausschlaggebend. In diesem Fall müssen zur fundierten empirischen Bestimmung des Leistungsverhaltens mehrere gleichartige Optimierungsexperimente mit jeweils unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators durchgeführt und analysiert werden. Dazu werden, basierend auf den oben genannten Basiskenngrößen, folgende Leistungskenngrößen definiert:

- Trefferwahrscheinlichkeit

Zur Quantifizierung des Optimierungserfolgs probabilistisch vorgehender direkter Optimierungsverfahren wird die  $\bar{x}^*$ -Trefferwahrscheinlichkeit  $p_{\bar{x}^*, \varepsilon}$  eingeführt, die wie folgt definiert ist: Wahrscheinlichkeit  $P(\bar{X} = \bar{x}^*, D \leq \varepsilon)$ , mit der ein direktes Optimierungsverfahren als Ergebnis  $o_E$  einen Punkt  $\bar{x}_{opt}$  zurückliefert, für dessen Abstand  $d$  vom (globalen oder lokalen) Optimumpunkt  $\bar{x}^*$  gilt:

$$d(\bar{x}^*, \bar{x}_{opt}) \leq \varepsilon.$$

Das Eintreten des Ereignisses  $\{\bar{X} = \bar{x}^*, D \leq \varepsilon\}$  wird im Folgenden  $\bar{x}^*$ -Treffer genannt. Als Abstandsnorm bietet sich bei der reellwertigen Parameteroptimierung der euklidische Abstand an.

- Treffergenauigkeit

Zur Quantifizierung der Ergebnisqualität direkter Optimierungsverfahren wird die  $\bar{x}^*$ -Treffergenauigkeit  $d_{\bar{x}^*}$  eingeführt, die wie folgt definiert ist: Handelt es sich bei einem Ergebnispunkt  $\bar{x}_{opt} \in o_E$  einer direkten Optimierungsstrategie um einen  $\bar{x}^*$ -Treffer, wird der Abstand  $d(\bar{x}^*, \bar{x}_{opt})$  als  $\bar{x}^*$ -Treffergenauigkeit bezeichnet.

- Brutto- und Netto-Optimierungsaufwand

Um den zur direkten Optimierung benötigten Rechenaufwand zu quantifizieren, werden die beiden Leistungskenngrößen Brutto- und Netto-Optimierungsaufwand eingeführt:

- Brutto-Optimierungsaufwand  $o_A^{brutto}$

Als Brutto-Optimierungsaufwand wird die Anzahl der während des Optimierungsprozesses mit einer direkten Optimierungsstrategie erzeugten Suchpunkte bezeichnet, wobei wiederholt erzeugte Suchpunkte mitgezählt werden.

- Netto-Optimierungsaufwand  $o_A^{netto}$

Da die benötigte Rechenzeit bei der Optimierung simulationsbasierter Zielfunktionen größtenteils von der Simulationsdauer bestimmt wird, ist vor allem die Anzahl der Suchpunkte von Interesse, die tatsächlich durch Simulation evaluiert werden. Neben dem Brutto-Optimierungsaufwand wird daher die Leistungskenngröße Netto-Optimierungsaufwand eingeführt. Der Netto-Optimierungsaufwand umfasst sämtliche Zielfunktions-

evaluierungen, die während des Optimierungsprozesses mit einer direkten Optimierungsstrategie durchgeführt wurden, wobei Mehrfachevaluierungen eines Suchpunktes mitgezählt werden. Damit entspricht der Netto-Optimierungsaufwand exakt der Basiskenngröße Optimierungsaufwand  $o_A$ . Der Brutto-Optimierungsaufwand ist stets größer oder gleich dem Netto-Optimierungsaufwand. Unterschiede ergeben sich dann, wenn durch Maßnahmen zur Reduktion des Optimierungsaufwands wie beispielsweise Vermeidung von Reevaluation oder Zielfunktionsapproximation Zielfunktionsevaluierungen eingespart werden können. In solchen Fällen übersteigt die Anzahl der erzeugten Suchpunkte die der tatsächlich evaluierten.

Die Tatsache, dass die Anwendung probabilistischer Optimierungsverfahren auf ein Optimierungsproblem ein Zufallsexperiment mit ungewissem Ausgang darstellt, macht die Quantifizierung der Leistungsfähigkeit dieser Verfahren durch die oben aufgeführten Kenngrößen zu einer anspruchsvollen und meist auch sehr zeitaufwändigen Aufgabe. Da sich die zu bestimmenden Leistungskenngrößen erst bei wachsender Anzahl von Versuchswiederholungen statistisch stabilisieren (Prinzip der großen Zahlen), sind zu ihrer empirischen Ermittlung umfangreiche Mehrfachexperimente<sup>6</sup> durchzuführen. Die dabei anfallenden Optimierungsdaten werden in Form von Datensätzen abgespeichert, um anschließend statistisch ausgewertet und auf die oben genannten Leistungskenngrößen reduziert werden zu können.

### 5.3.2 Testprobleme

Zur empirischen Bewertung direkter Optimierungsverfahren können als Testprobleme sowohl Simulationsmodelle als auch mathematische Testfunktionen herangezogen werden. Mathematische Funktionen haben gegenüber simulationsbasierten Zielfunktionen jedoch zwei entscheidende Vorteile: Ihre Extremstellen lassen sich meist auf analytische Weise im Voraus bestimmen und die i.a. sehr kurze Evaluierungsdauer ermöglicht die Durchführung einer statistisch relevanten Anzahl von Optimierungsexperimenten in relativ kurzer Zeit. Die im Folgenden beschriebenen Leistungsuntersuchungen basieren daher ausschließlich auf mathematischen Testproblemen. Ergebnisse zu Optimierungsexperimenten mit simulationsbasierten Zielfunktionen finden sich in [Syrj97] und [Info99].

Die Sucheigenschaften der in diesem Beitrag beschriebenen Optimierungsalgorithmen sollen anhand von vier mathematischen Testproblemen mit sehr unterschiedlichem Schwierigkeitsgrad demonstriert werden. Alle vier Problemstellungen weisen hochgradig nicht-lineare multimodale Zielfunktionen mit einem globalen und mehreren lokalen Optimumpunkten auf.

In Tabelle 5.3.2.1 ist das erste Testproblem  $(L_1^n, F_1^n)$  spezifiziert. Hier besteht die Aufgabe darin, eine trigonometrische Zielfunktion zu optimieren, die für alle Problemdimensionen  $n \in \mathbb{N}$  genau einen globalen Maximumpunkt  $\bar{x}^* = (7.98, 7.98, \dots, 7.98)$  aufweist. Beachtet werden sollte, dass die Anzahl der lokalen Maximumpunkte von  $(L_1^n, F_1^n)$  mit wachsendem  $n$  exponentiell ansteigt.

---

<sup>6</sup> Als Mehrfachexperiment wird die mehrfache Ausführung eines fest parametrisierten Optimierungsverfahrens zu einem Optimierungsproblem mit jeweils unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators bezeichnet.

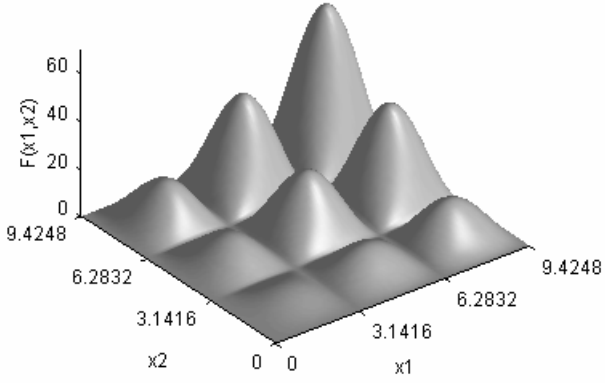
Spezifikation des Optimierungsproblems	Ergebnisoberfläche von $F_1^2$ über $L_1^2$
<p>Lösungsraum:</p> $L_1^n = \{\vec{x} \in \mathbb{R}^n \mid 0 \leq x_i \leq 3 \cdot \pi; i \in \{1, \dots, n\}\}$ <p>Zielfunktion:</p> $F_1^n(\vec{x}) = \left  \prod_{i=1}^n x_i \cdot \sin(x_i) \right $	

Tabelle 5.3.2.1: Mathematisches Testproblem ( $L_1^n, F_1^n$ )

Bei der Zielfunktion des zweiten Testproblems ( $L_2^n, F_2^n$ ) aus Tabelle 5.3.2.2 handelt es sich um eine Benchmarkfunktion, die sehr häufig zur Bewertung von direkten Optimierungsverfahren herangezogen wird.

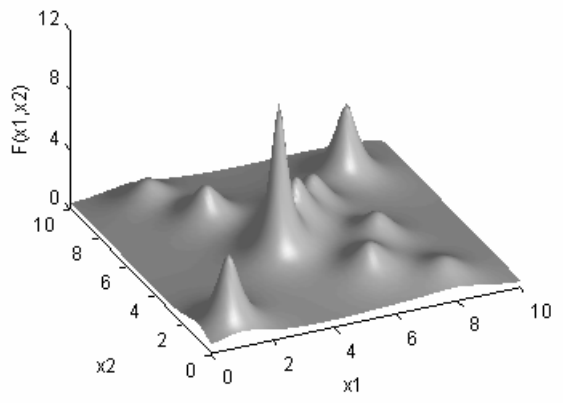
Spezifikation des Optimierungsproblems	Ergebnisoberfläche von $F_2^2$ über $L_2^2$
<p>Lösungsraum:</p> $L_2^n = \{\vec{x} \in \mathbb{R}^n \mid 0 \leq x_i \leq 10; i \in \{1, \dots, n\}\}$ <p>Zielfunktion:</p> $F_2^n(\vec{x}) = \sum_{j=1}^{10} \frac{1}{c_j + \sum_{i=1}^n (x_i - a_{ij})^2};$ <p>mit <math>a_{ij}</math> aus Tabelle 5.3.2.3 und <math>c_j</math> aus Tabelle 5.3.2.4</p>	

Tabelle 5.3.2.2: Mathematisches Testproblem ( $L_2^n, F_2^n$ )



Funktionen dieser Bauart werden auch als Shekel-Funktionen bezeichnet. Für alle Problemdimensionen  $n \in \mathbb{N}$  weist  $(L_2^n, F_2^n)$  genau einen globalen Maximumpunkt  $\bar{x}^* = (4, 4, \dots, 4)$  und neun lokale Maximumpunkte auf. Die Koordinaten  $a_{ij}; i, j \in \{1, \dots, 10\}$  der zehn Maximumpunkte von  $(L_2^n, F_2^n)$  sind in Tabelle 5.3.2.3 zusammengefasst.

$a_{ij}$	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10
1j	4	1	8	5	6	3	6	7	2	8
2j	4	1	8	5	6	7	2	3.6	9	1
3j	4	1	8	5	6	3	6	7	2	8
4j	4	1	8	5	6	7	2	3.6	9	1
5j	4	1	8	5	6	3	6	7	2	8
6j	4	1	8	5	6	7	2	3.6	9	1
7j	4	1	8	5	6	3	6	7	2	8
8j	4	1	8	5	6	7	2	3.6	9	1
9j	4	1	8	5	6	3	6	7	2	8
10j	4	1	8	5	6	7	2	3.6	9	1

Tabelle 5.3.2.3: Koordinaten der Maximumpunkte von Testproblem  $(L_2^n, F_2^n)$

Durch die Konstanten  $c_j; j \in \{1, \dots, 10\}$  aus Tabelle 5.3.2.4 werden die Zielfunktionswerte der Maximumpunkte von  $(L_2^n, F_2^n)$  festgelegt.

$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	$c_7$	$c_8$	$c_9$	$c_{10}$
0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.5	0.6	0.7

Tabelle 5.3.2.4: Konstanten zur Festlegung der Maxima von Testproblem  $(L_2^n, F_2^n)$

Das dritte Testproblem  $(L_3^n, F_3^n)$  aus Tabelle 5.3.2.5 stellt bei höherer Problemdimension ein quasi unlösbares Problem dar. Die Zielfunktion  $F_3^n$  ist eine Kombination aus einer Sinus-Funktion und einer Shekel-Funktion mit genau einem Optimumpunkt. Für alle Problemdimensionen weist  $(L_3^n, F_3^n)$  genau einen globalen Maximumpunkt  $\bar{x}^* = (4, 4, \dots, 4)$  auf. Im Gegensatz zu  $(L_2^n, F_2^n)$  bleibt die Anzahl der lokalen Maximumpunkte bei wachsendem  $n$  jedoch nicht konstant, sondern steigt exponentiell an. Betrachtet man den Verlauf der Ergebnisoberfläche von  $F_3^2$  über  $L_3^2$  wird sehr schnell deutlich, dass  $(L_3^n, F_3^n)$  mit wachsender Problemdimension  $n$  zunehmend der sprichwörtlichen Suche nach der Nadel im Heuhaufen gleicht.

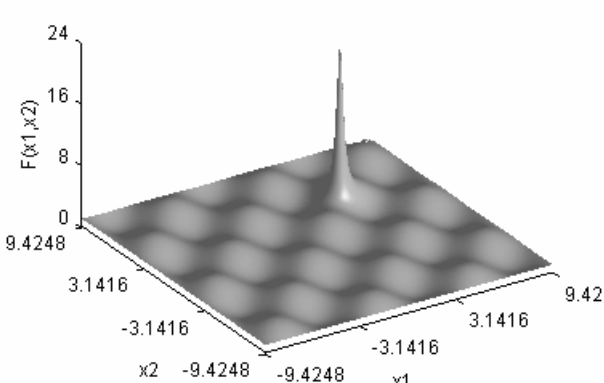
Spezifikation des Optimierungsproblems	Ergebnisoberfläche von $F_3^2$ über $L_3^2$
<p>Lösungsraum:</p> $L_3^n = \left\{ \begin{array}{l} \bar{x} \in \mathbb{R}^n \mid -3 \cdot \pi \leq x_i \leq 3 \cdot \pi; \\ i \in \{1, \dots, n\} \end{array} \right\}$ <p>Zielfunktion:</p> $F_3^n(\bar{x}) = 1 + \prod_{i=1}^n \sin(x_i) + (0.05 + \sum_{i=1}^n (x_i - 4)^2)^{-1}$	

Tabelle 5.3.2.5: Mathematisches Testproblem  $(L_3^n, F_3^n)$

Beim vierten Testproblem  $(L_4^n, F_4^n)$  aus Tabelle 5.3.2.6 handelt es sich wie bereits in Abschnitt 5.1.5 beschrieben um ein für Evolutionäre Algorithmen irreführendes Problem. Die Zielfunktion  $F_4^n$  stellt eine Kombination aus einer quadratischen Funktion und einer Shekel-Funktion mit genau einer Spitze dar. Für alle Problemdimensionen weist  $(L_4^n, F_4^n)$  genau einen globalen Maximumpunkt  $\bar{x}^* = (0, 0, \dots, 0)$  auf.

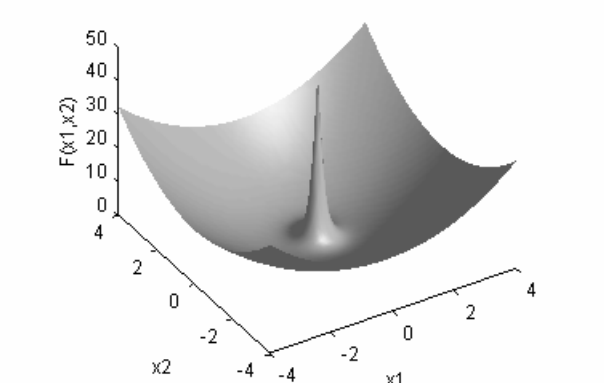
Spezifikation des Optimierungsproblems	Ergebnisoberfläche von $F_4^2$ über $L_4^2$
<p>Lösungsraum:</p> $L_4^n = \left\{ \bar{x} \in \mathbb{R}^n \mid -a_i \leq x_i \leq a_i; i \in \{1, \dots, n\} \right\}$ <p>Zielfunktion:</p> $F_4^n(\bar{x}) = \sum_{i=1}^n x_i^2 + \frac{1}{0.02 + \sum_{i=1}^n x_i^2}$	

Tabelle 5.3.2.6: Mathematisches Testproblem  $(L_4^n, F_4^n)$

Die Werte der Konstanten  $a_i; i \in \{2, \dots, 4\}$ , welche die Zielfunktionswerte der vier lokalen Maximumpunkte von  $(L_4^n, F_4^n)$  festlegen, sind in Tabelle 5.3.2.7 zusammengefasst.

$a_2$	$a_3$	$a_4$
4	2.5	2

Tabelle 5.3.2.7: Werte der Konstanten  $a_i; i \in \{2, \dots, 4\}$  von Testproblem  $(L_4^n, F_4^n)$

### 5.3.3 Optimierungsergebnisse

Die im Folgenden vorgestellten Optimierungsergebnisse stammen aus verschiedenen Experimentreihen zu einem mehrstufigen Optimierungsverfahren  $os_{ms}$ . Um einen umfassenden Überblick über das Leistungsverhalten von  $os_{ms}$  zu bekommen, wurden innerhalb einer Experimentreihe Mehrfachexperimente zu verschiedenen Dimensionen eines der in Abschnitt 5.3.2 beschriebenen Testprobleme durchgeführt. Um statistisch relevante Ergebnisse zu erhalten, wurden innerhalb eines Mehrfachexperiments stets 200 Optimierungsläufe mit  $os_{ms}$  durchgeführt, die jeweils mit unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators gestartet und jeweils nach genau 20 Optimierungsstufen abgebrochen wurden. Insgesamt wurden innerhalb eines Mehrfachexperiments also 4000 Optimumpunkte lokalisiert.

Innerhalb des untersuchten mehrstufigen Optimierungsverfahrens  $os_{ms}$  wurde eine kombinierte 2-Phasen Strategie  $os_{2p}$  eingesetzt. Zur Voroptimierung wurde, falls nicht anders angegeben, ein Genetischer Algorithmus (GA) verwendet. Die Feinoptimierung wurde mit der Mustersuche (MS) von Hooke und Jeeves durchgeführt.

Neben der zu optimierenden Problemstellung sind für das Leistungsverhalten einer Optimierungsstrategie auch deren Kontrollparametereinstellungen von großer Bedeutung. Um nicht zu tief ins Detail gehen zu müssen, werden im Folgenden nur die wichtigsten Kontrollparametereinstellungen von  $os_{ms}$  angegeben. Die Parametrisierung des innerhalb der kombinierten 2-Phasen Strategie  $os_{2p}$  zur Voroptimierung verwendeten Genetischen Algorithmus wurde in Anlehnung an Standardeinstellungen aus der Literatur [SCED89] vorgenommen. Für die Crossoverwahrscheinlichkeit, die Mutationswahrscheinlichkeit sowie die Länge eines Parametersegments im Gesamtbinärvektor wurden in allen Experimentreihen, unabhängig von der Problemdimension, jeweils dieselben in Tabelle 5.3.3.1 aufgeführten Werte verwendet. Für die den Voroptimierungsaufwand maßgeblich beeinflussenden Parameter  $mc$  (Monte-Carlo Initialisierung<sup>7</sup>),  $pg$  (Populationsgröße) und  $t$  (Kontrollparameter des Terminierungskriteriums

<sup>7</sup> Bei der Monte Carlo Initialisierung werden zunächst  $mc$  Suchpunkte zufällig im Lösungsraum erzeugt. Im Anschluss daran werden daraus die  $pg$  besten Suchpunkte ermittelt, welche letztendlich die Startpopulation des Genetischen Algorithmus bilden.

$T_{VO}$ <sup>8</sup>), wurden dagegen auf die in den einzelnen Experimentreihen bearbeiteten Problemstellungen angepasste Einstellungen verwendet. Die entsprechenden Werte werden daher zu jeder Experimentreihe separat angegeben.

GA-Kontrollparameter	Wert
Crossoverwahrscheinlichkeit $c_o$	0.75
Mutationswahrscheinlichkeit $\mu$	0.075
Länge eines Parametersegments im Gesamtbinärvektor $l_i, i \in \{1, \dots, n\}$	10 Bit

*Tabelle 5.3.3.1: In allen Experimentreihen verwendete von der Problemdimension unabhängige Kontrollparametereinstellungen des Genetischen Algorithmus*

Die Ergebnisqualität, die von der zur Feinoptimierung eingesetzten Mustersuche in sämtlichen Experimentreihen gefordert wurde, lag bei einer Abweichung  $\varepsilon$  von maximal 0.01 vom jeweiligen Optimumpunkt pro Koordinatenrichtung. Auf die weiteren Kontrollparametereinstellungen von  $os_{ms}$  (beispielsweise die Einstellungen der von VR verwendeten Skalierungsparameter) wird ausführlich in [Syrj97] eingegangen.

Im Folgenden werden die Ergebnisse von zehn umfangreichen Experimentreihen  $E_i; i \in \{1, 2, \dots, 10\}$  mit  $os_{ms}$  zu den in Abschnitt 5.3.2 vorgestellten Testproblemen beschrieben und einander gegenübergestellt. Das mehrstufige Optimierungsverfahren  $os_{ms}$  wurde dabei stets zur Maximierung eingesetzt.

### 5.3.3.1 Ergebnisse zu Testproblem $(L_1^n, F_1^n)$

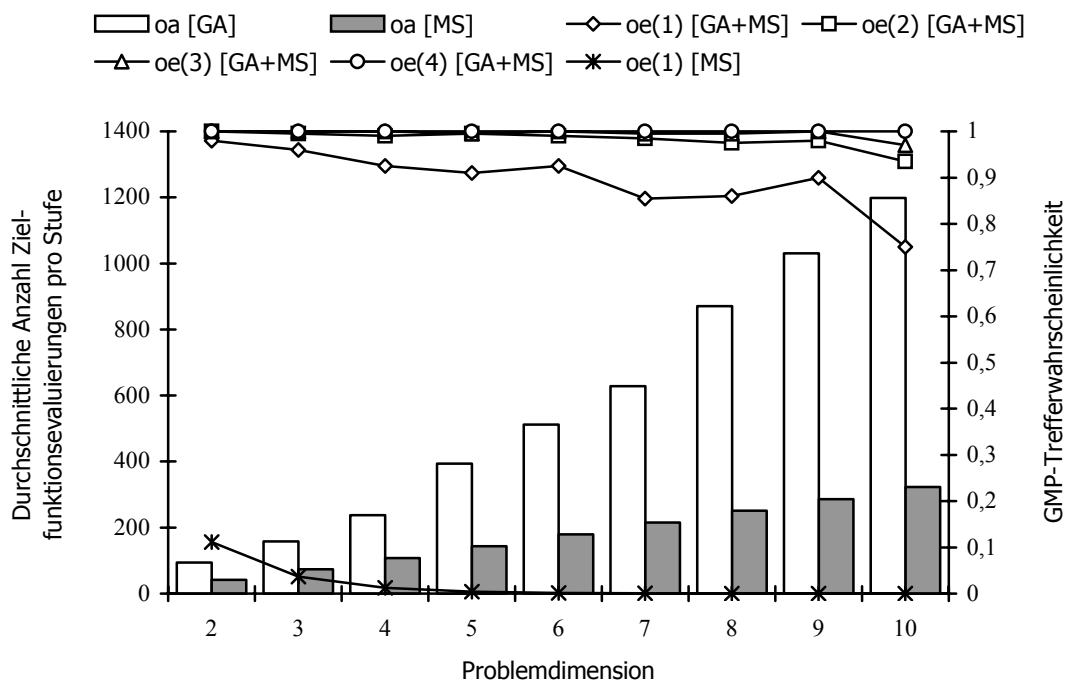
In Experimentreihe  $E_1$  wurde mit dem oben beschriebenen mehrstufigen Optimierungsverfahren  $os_{ms}$  das Testproblem  $(L_1^n, F_1^n); n \in \{2, \dots, 10\}$  maximiert. Die in  $E_1$  verwendeten Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus sind in Tabelle 5.3.3.2 zusammengefasst. Die jeweiligen Werte wurden mit steigender Problemdimension erhöht, um den Voroptimierungsaufwand an die exponentiell anwachsende Größe des Lösungsraums anzupassen.

<sup>8</sup> Das Umschalten von Vor- auf Feinoptimierung erfolgt, wenn das Terminierungskriterium  $T_{VO}$  erfüllt ist. Dies war in den durchgeführten Experimentreihen genau dann der Fall, wenn der Genetische Algorithmus die Zielfunktion nicht um 5% innerhalb der letzten  $t$  Generationen verbessern konnte.

n	2	3	4	5	6	7	8	9	10
mc	40			60			80		
pg	20			30			40		
t	2	3	4	5	6	7	8		

Tabelle 5.3.3.2: Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus in Experimentreihe  $E_1$

Diagr. 5.3.3.1 zeigt die wesentlichen Ergebnisse von Experimentreihe  $E_1$ . Zu jeder der neun betrachteten Problemdimensionen  $n \in \{2, \dots, 10\}$  wurde jeweils ein Mehrfachexperiment durchgeführt. Wie bereits erwähnt umfasste ein solches Mehrfachexperiment 200 mit jeweils unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators gestartete Optimierungsläufe mit  $os_{ms}$ , wobei jeder Optimierungslauf nach genau 20 Optimierungsstufen abgebrochen wurde.



Diagr. 5.3.3.1: In Experimentreihe  $E_1$  ermittelter Optimierungserfolg und -aufwand

Auf der x-Achse von Diagr. 5.3.3.1 ist die Problemdimension  $n$  aufgetragen. Die Säulen des Verbunddiagramms stellen den durchschnittlichen Netto-Optimierungsaufwand (durchschnittliche Anzahl durchgeführter Zielfunktionsevaluierungen) pro Optimierungsstufe dar. Die weiß gefärbten Säulen zeigen den Netto-Optimierungsaufwand des Genetischen Algorithmus (Voroptimierungsaufwand). Die grau gefärbten Säulen stehen für den von der Mustersuche benötigten Netto-Optimierungsaufwand (Feinoptimierungsaufwand). Die linienförmig dargestellten

Kurven geben den Optimierungserfolg nach der  $s$ -ten Optimierungsstufe  $oe(s)$ ;  $s \in \{1,2,3,4\}$  an, der wie folgt definiert ist:

$$oe(s) = \sum_{i=1}^s p_{\bar{x}^*, \varepsilon}^i.$$

$s$  bezeichnet die Optimierungsstufe,  $p_{\bar{x}^*, \varepsilon}^i$  stellt die empirische Wahrscheinlichkeit für das erstmalige Auffinden des globalen Maximumpunkts<sup>9</sup>  $\bar{x}^*$  in Optimierungsstufe  $i$  mit Genauigkeit  $\varepsilon$  dar:

$$p_{\bar{x}^*, \varepsilon}^i = \frac{\text{Anzahl von } (\bar{x}^*, \varepsilon)\text{-Erstlokalisierungen in Optimierungsstufe } i}{\text{Insgesamt durchgeführte Optimierungsläufe in Optimierungsstufe } i}.$$

Die Kurve  $oe(1)$  [GA+MS] aus Diagr. 1 zeigt, dass  $os_{ms}$  bereits nach der ersten Optimierungsstufe, also nach einmaliger Ausführung von  $os_{2p} = \text{GA+MS}$ , einen sehr hohen Optimierungserfolg aufweist, der nur gering mit steigender Problemdimension abfällt. Offensichtlich handelt es sich bei  $(L_1^n, F_1^n)$  also um ein für den Genetischen Algorithmus sehr leicht zu bearbeitendes Problem. Dies liegt vor allem daran, dass die Zielfunktionswerte der lokalen Maximumpunkte von  $(L_1^n, F_1^n)$  monoton in Richtung des globalen Maximums ansteigen (vergleiche dazu auch die in Tabelle 5.3.2.1 dargestellte Ergebnisoberfläche von  $F_1^2$  über  $L_1^2$ ) und somit dem Genetischen Algorithmus den Weg zum Optimierungsziel weisen. Des Weiteren verfügen die Extremstellen von  $(L_1^n, F_1^n)$  jeweils über einen ausgeprägten Einzugsbereich und die Ergebnisverläufe der Zielfunktion über den Einzugsbereichen stellen keine nadelförmigen Spitzen dar. Damit sind zwei wesentliche vereinfachende Annahmen aus Abschnitt 2.3.2 erfüllt. Eine weitere wichtige vereinfachende Annahme, die darin besteht, dass die zu optimierende Zielfunktion eine überschaubare Anzahl von Extremstellen aufweisen sollte, ist jedoch nur bei niedriger Problemdimension erfüllt, da die Anzahl der lokalen Maximalstellen bei  $(L_1^n, F_1^n)$  mit wachsender Problemdimension exponentiell ansteigt. Da jedoch die lokalen Maxima in Richtung des globalen Maximums monoton ansteigen, wird der Optimierungserfolg des Genetischen Algorithmus durch deren exponentiell anwachsende Anzahl nicht allzu sehr beeinträchtigt.

Die Kurve  $oe(1)$  [MS] zeigt den Optimierungserfolg nach der ersten Optimierungsstufe, der sich einstellt, wenn auf die Voroptimierung mit dem Genetischen Algorithmus verzichtet und die Mustersuche von einem zufällig ausgewählten Startpunkt aus gestartet wird. Aufgrund der mit wachsender Problemdimension exponentiell ansteigenden Anzahl lokaler Maximumpunkte sinken in diesem Fall die Erfolgsaussichten, den globalen Maximumpunkt  $\bar{x}^*$  von Testproblem  $(L_1^n, F_1^n)$  zu finden, sehr schnell auf einen Wert nahe bei Null.

Die drei Kurven  $oe(2)$  [GA+MS],  $oe(3)$  [GA+MS] und  $oe(4)$  [GA+MS] zeigen, dass jede weitere mit  $os_{2p} = \text{GA+MS}$  durchgeführte Optimierungsstufe einen monotonen Anstieg des Optimierungserfolgs nach sich zieht. Nach Abschluss der dritten Optimierungsstufe liegt der bis

<sup>9</sup> in den Ergebnisdigrammen wird der globale Maximumpunkt mit GMP abgekürzt

dahin erreichte Optimierungserfolg nahezu beim Maximalwert 1 für alle betrachteten Problemdimensionen.

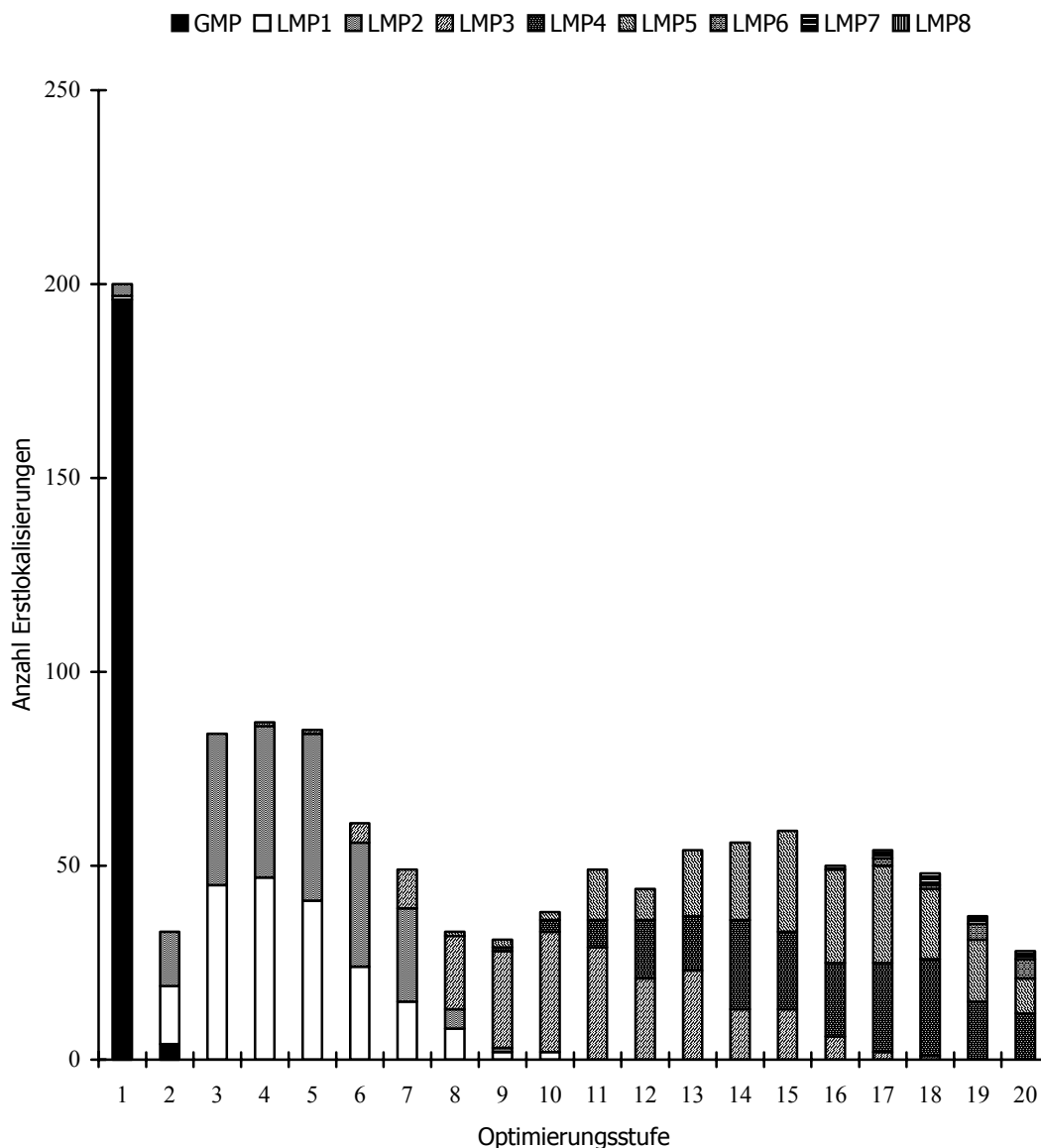
Aus der Kurve  $o_a$  [MS] ist ersichtlich, dass der Feinoptimierungsaufwand linear mit wachsendem  $n$  ansteigt. Da die Exploration des gesamten Lösungsraums eine sehr viel schwierigere Aufgabe darstellt als die lokale Optimierung in einem eng begrenzten Suchraumausschnitt, trifft dies natürlich nicht mehr auf den Voroptimierungsaufwand  $o_a$  [GA] zu. Hier ist zu berücksichtigen, dass mit wachsender Problemdimension die Größe des zu durchsuchenden Lösungsraums exponentiell ansteigt.

Zur Qualität der von der Feinoptimierung erzeugten Optimierungsergebnisse ist zu sagen, dass die vom Anwender spezifizierte Approximationsgenauigkeit  $\varepsilon=0.01$  von der Mustersuche stets eingehalten wurde. Auf ein spezielles Diagramm zur Darstellung des Verlaufs der Approximationsgüte der Feinoptimierungsergebnisse wurde daher verzichtet.

Den Optimierungserfolg mit jeder durchgeführten Optimierungsstufe monoton zu steigern, ist nicht das einzige Ziel der mehrstufigen Optimierung. Darüber hinaus wird angestrebt, dass möglichst heterogene Folgen von Optimumpunkten erzeugt werden. Dies zu gewährleisten, fällt in den Aufgabenbereich der Methode zur Vermeidung von Reexploration (VR), welche erschweren soll, dass bereits aufgefundene Optimumpunkte in nachfolgenden Optimierungsstufen nochmals lokalisiert werden.

Um aufzuzeigen, dass bei der mehrstufigen Optimierung mit  $os_{ms}$  durch Anwendung von VR tatsächlich heterogene Folgen von Optimumpunkten erzeugt werden, wurde in Diagr. 5.3.3.2 zu jeder durchgeführten Optimierungsstufe  $s \in \{1, 2, \dots, 20\}$  die Anzahl der Erstlokalisierungen der aufzufindenden Maximumpunkte aufgetragen. Bei dem zugrunde liegenden Optimierungsproblem handelt es sich um die in Experimentreihe  $E_1$  untersuchte 2-dimensionale Problemstellung  $(L_1^2, F_1^2)$ , deren in Tabelle 5.3.2.1 dargestellte Ergebnisoberfläche insgesamt neun Maximumpunkte aufweist. Diagr. 5.3.3.2 zeigt sehr eindrucksvoll, dass die markantesten Extremstellen von  $(L_1^2, F_1^2)$  aufgrund der Voroptimierung mit dem Genetischen Algorithmus bereits in sehr frühen Optimierungsstufen gefunden werden. In Übereinstimmung mit den Ergebnissen aus Diagr. 5.3.3.1 wird der globale Maximumpunkt GMP in 196 der insgesamt 200 durchgeführten Optimierungsexperimente bereits in der ersten Optimierungsstufe lokalisiert. In den verbleibenden 4 Optimierungsexperimenten der ersten Optimierungsstufe werden die beiden lokalen Maximumpunkte LMP1 und LMP2 mit dem zweitbesten Zielfunktionswert aufgefunden. Die weiteren Erstlokalisierungen dieser Maximumpunkte erfolgen bis einschließlich Optimierungsstufe 10. Ab Optimierungsstufe 4 sind Erstlokalisierungen des lokalen Maximumpunkts LMP3 zu verzeichnen, der den drittbesten Zielfunktionswert aufweist. Die Erstlokalisierungen der lokalen Maximumpunkte LMP4 und LMP5 mit dem viertbesten Zielfunktionswert beginnen ab Optimierungsstufe 8. Die beiden lokalen Maximumpunkte LMP6 und LMP7 mit dem fünftbesten Zielfunktionswert werden ab Optimierungsstufe 16 erstmalig aufgefunden. Der lokale Maximumpunkt LMP8 mit dem schlechtesten Zielfunktionswert bleibt dagegen unentdeckt. Um dieses sehr kleine lokale Maximum aufzufinden, reichen 20 Optimierungsstufen offensichtlich nicht aus.

Abschließend sei darauf hingewiesen, dass im Falle einer Deaktivierung von VR die Lokalisierung des globalen Maximumpunkts GMP nicht nur die erste, sondern sämtliche Optimierungsstufen dominieren würde.



Diagr. 5.3.3.2: Anzahl von Erstlokalisierungen der neun Maximumpunkte von Testproblem  $(L_1^2, F_1^2)$  in den Optimierungsstufen  $s \in \{1, \dots, 20\}$

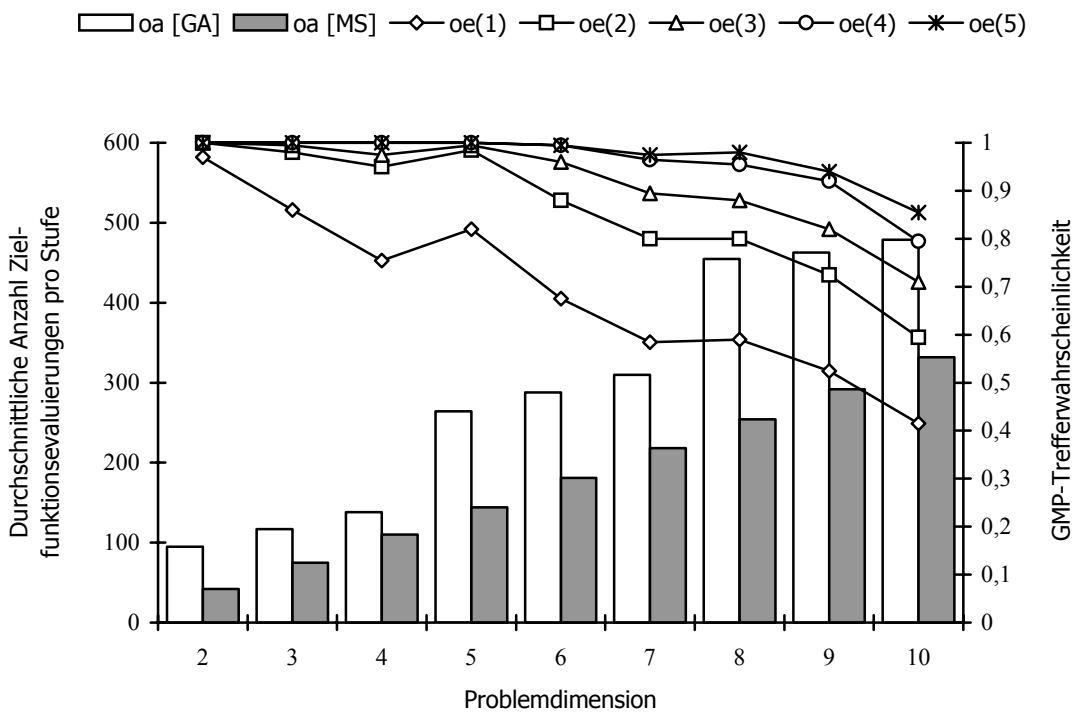
In Experimentreihe  $E_2$  wurde mit dem mehrstufigen Optimierungsverfahren  $os_{ms}$  ebenfalls das Testproblem  $(L_1^n, F_1^n)$ ;  $n \in \{2, \dots, 10\}$  maximiert. Im Unterschied zu  $E_1$  wurde jedoch bei wachsender Problemdimension mit einem weitaus geringeren Voroptimierungsaufwand gearbeitet. Dies wurde erreicht, indem die beiden Kontrollparameter  $mc$  und  $pg$  nicht wie in Experimentreihe  $E_1$  mit wachsender Problemdimension erhöht, sondern dimensionsunabhängig auf die in Tabelle 5.3.3.3 dargestellten Werte gesetzt wurden. Nur der die Terminierung der Voroptimierung beeinflussende Parameter  $t$  wurde bei Problemdimension 5 und 8 deutlich angehoben, um einen zu frühen Abbruch des Genetischen Algorithmus zu vermeiden.



n	2	3	4	5	6	7	8	9	10
mc	40								
pg	20								
t	2			4			6		

Tabelle 5.3.3.3: Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus in Experimentreihe  $E_2$

In Diagr. 5.3.3.3 sind die wesentlichen Ergebnisse von Experimentreihe  $E_2$  zusammengefasst. Auch hier wurde zu jeder der neun betrachteten Problemdimensionen  $n \in \{2, \dots, 10\}$  jeweils ein Mehrfachexperiment durchgeführt. Im Vergleich zu Diagr. 5.3.3.1 nimmt der Optimierungserfolg mit wachsender Problemdimension aufgrund des geringeren Voroptimierungsaufwands deutlich ab. Ab Problemdimension 9 reicht der gegenüber Experimentreihe  $E_1$  stark reduzierte Voroptimierungsaufwand nicht mehr aus, um nach 5 Optimierungsstufen den globalen Maximpunkt  $\bar{x}^*$  mit einer Wahrscheinlichkeit nahe bei Eins zu lokalisieren.



Diagr. 5.3.3.3: In Experimentreihe  $E_2$  ermittelter Optimierungserfolg und -aufwand

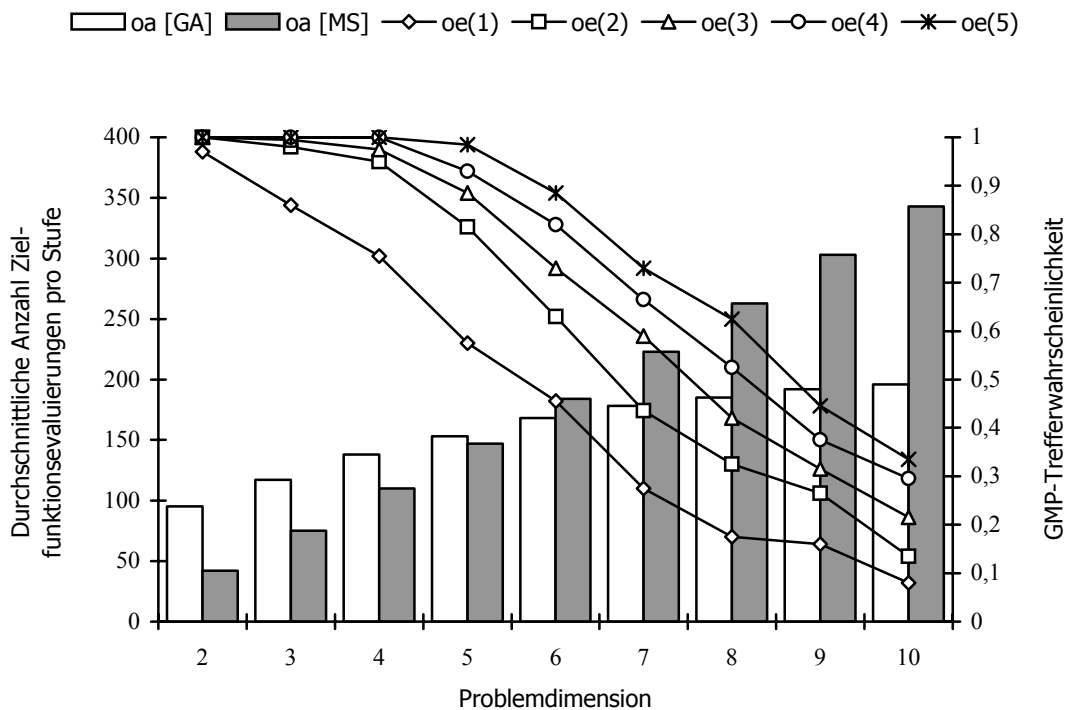
In Experimentreihe  $E_3$  wurde mit dem mehrstufigen Optimierungsverfahren  $os_{ms}$  das Testproblem  $(L_1^n, F_1^n)$  ein drittes Mal maximiert, wobei auch hier zu jeder der neun betrachteten Problemdimensionen  $n \in \{2, \dots, 10\}$  jeweils ein Mehrfachexperiment durchgeführt wurde. Diesmal wurden jedoch nicht nur die Kontrollparameter  $mc$  und  $pg$ , sondern auch der Parameter  $t$  dimensionsunabhängig auf die in Tabelle 5.3.3.4 dargestellten Werte gesetzt. Mit wachsender

Problemdimension kommt es dadurch zu einem noch stärkeren Rückgang des Voroptimierungsaufwands als in Experimentreihe  $E_2$ .

n	2	3	4	5	6	7	8	9	10
mc	40								
pg	20								
t	2								

Tabelle 5.3.3.4: Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus in Experimentreihe  $E_3$

Die daraus resultierenden Auswirkungen auf den Optimierungserfolg zeigt Diagr. 5.3.3.4, in dem die wesentlichen Ergebnisse von Experimentreihe  $E_3$  zusammengefasst sind. Im Vergleich zu Diagr. 5.3.3.3 ist hier schon ab Dimension 6 ein drastischer Einbruch des Optimierungserfolgs zu verzeichnen. Bei den Problemdimensionen 9 und 10 ist der Optimierungserfolg nach der fünften Optimierungsstufe bereits deutlich unter 0.5 abgesunken.



Diagr. 5.3.3.4: In Experimentreihe  $E_3$  ermittelter Optimierungserfolg und -aufwand

Die in Diagr. 5.3.3.3 und 5.3.3.4 dargestellten Optimierungsergebnisse sind ein deutlicher Hinweis darauf, dass die den Voroptimierungsaufwand beeinflussenden Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus nicht unabhängig von der Problemdimension eingestellt

werden sollten. Auch bei extrem gutartigen Optimierungsproblemen wie  $(L_1^n, F_1^n)$  muss stets auf ausreichend hohen Voro Optimierungsaufwand geachtet werden, um den Optimierungserfolg nicht zu gefährden.

### 5.3.3.2 Ergebnisse zu Testproblem $(L_2^n, F_2^n)$

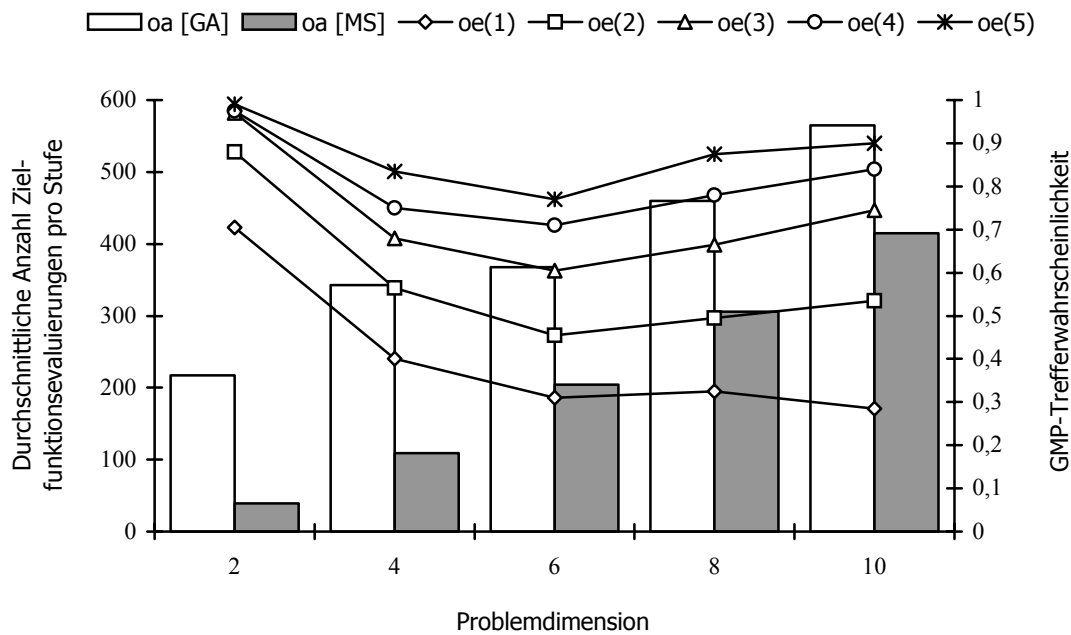
In Experimentreihe  $E_4$  bestand die Optimierungsaufgabe darin, Testproblem  $(L_2^n, F_2^n)$ ;  $n \in \{2, 4, 6, 8, 10\}$  zu maximieren. Zu jeder der fünf betrachteten Problemdimensionen wurde jeweils ein Mehrfachexperiment durchgeführt. Analog zu den drei vorangegangenen Experimentreihen setzte sich ein solches Mehrfachexperiment aus 200 Optimierungsläufen mit  $os_{ms}$  zusammen, wobei jeder Optimierungslauf mit  $os_{ms}$  nach genau 20 Optimierungsstufen abgebrochen wurde. Die in Experimentreihe  $E_4$  verwendeten Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des zur Voro Optimierung eingesetzten Genetischen Algorithmus sind in Tabelle 5.3.3.5 zusammengefasst. Auch hier wurden zunächst die Einstellungen mit steigender Problemdimension erhöht, um den Voro Optimierungsaufwand an die exponentiell anwachsende Größe des Lösungsraums anzupassen.

n	2	4	6	8	10
mc	60	80			
pg	30	40			
t	4		5	7	9

Tabelle 5.3.3.5: Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus in Experimentreihe  $E_4$

Diagr. 5.3.3.5 zeigt die wesentlichen Ergebnisse von Experimentreihe  $E_4$ . Der stark abfallende Verlauf von Kurve  $oe(1)$  zeigt deutlich, dass Testproblem  $(L_2^n, F_2^n)$  dem zur Voro Optimierung eingesetzten Genetischen Algorithmus weitaus mehr Schwierigkeiten bereitet, als das in den drei vorangegangenen Experimentreihen bearbeitete Testproblem  $(L_1^n, F_1^n)$ . Dies liegt vor allem an den für direkte Optimierungsverfahren sehr unschönen Eigenschaften der Shekel-Funktion. Diese weist zwar für alle Problemdimensionen eine feste Anzahl von Maximalstellen<sup>10</sup> auf, mit wachsender Problemdimension verformt sich die Zielfunktionsoberfläche über diesen Maximalstellen jedoch mehr und mehr zu steil nach oben ragenden Nadeln, die sich nur noch in ihrer Höhe unterscheiden. Dadurch wird es für das Voro Optimierungsverfahren extrem schwierig, den Einzugsbereich des globalen Maximumpunkts aufzufinden, weil sich die Zielfunktionsverläufe über den Einzugsbereichen der Maximalstellen in ihrer Form kaum noch voneinander unterscheiden.

<sup>10</sup> die hier maximierte Shekel-Funktion hat zehn Maximalstellen (siehe dazu auch Tabelle 5.3.2.2)



Diagr. 5.3.3.5: In Experimentreihe  $E_4$  ermittelter Optimierungserfolg und -aufwand

Bei der 2-dimensionalen Shekel-Funktion arbeitet der Genetische Algorithmus noch sehr zufriedenstellend, weil in diesem Fall die Zielfunktionsoberfläche einen Gebirgsverlauf mit deutlich voneinander unterscheidbaren Erhebungen darstellt (siehe dazu auch Tabelle 5.3.2.2) und damit die vereinfachenden Annahmen aus Abschnitt 2.3.2 erfüllt sind. Ab Problemdimension 6 ist  $oe(1)$  jedoch bereits auf ein Niveau abgefallen, das ungefähr der Auffindwahrscheinlichkeit des globalen Maximumpunkts  $\bar{x}^*$  mit der an einem zufällig ausgewählten Punkt gestarteten Mustersuche entspricht. Weiterführende Experimente mit  $os_{2p}$  haben gezeigt, dass bei schwierigen Problemstellungen wie  $(L_2^n, F_2^n)$  auch durch eine beträchtliche Steigerung des Voroptimierungsaufwands<sup>11</sup> keine wesentliche Verbesserung von  $oe(1)$  mehr zu erreichen ist. In solchen Fällen hat es sich als sehr viel effektiver erwiesen,  $os_{2p}$  nicht einmalig und mit sehr hohem Voroptimierungsaufwand, sondern innerhalb eines mehrstufigen Optimierungsprozesses mehrmals hintereinander mit jeweils moderatem Voroptimierungsaufwand auszuführen. Die vier Kurven  $oe(i)$ ;  $i \in \{2,3,4,5\}$  belegen den Erfolg dieser Vorgehensweise. Hier wurde mit jeder weiteren Optimierungsstufe ein deutlicher Anstieg des Optimierungserfolgs erreicht. Nach der fünften Optimierungsstufe ist die Wahrscheinlichkeit des Auffindens von  $\bar{x}^*$  bereits wieder auf über 0,75 für alle betrachteten Problemdimensionen angestiegen. Diese sehr vorteilhafte Eigenschaft der mehrstufigen Optimierung ist vor allem auf die Methode zur Vermeidung von Reexploration zurückzuführen. Diese bewirkt, dass in jeder Optimierungsstufe mit hoher Wahrscheinlichkeit eine noch nicht betretene Region des Lösungsraums durchsucht wird.

<sup>11</sup> beispielsweise durch eine Erhöhung des Kontrollparameters  $t$  des Terminierungskriteriums  $T_{VO}$

Der niedrige Voroptimierungsaufwand  $oa$  [GA] ist ein weiteres Anzeichen dafür, dass es sich bei  $(L_2^n, F_2^n)$  um ein nicht-triviales Testproblem handeln muss. Insbesondere bei den höheren Problemdimensionen  $n \in \{4, 6, 8, 10\}$  fällt der Voroptimierungsaufwand im Vergleich zu  $E_1$  bei ähnlicher Parametrisierung von  $mc$ ,  $pg$  und  $t$  deutlich geringer aus. Dieses Verhalten ist auf die sich mehr und mehr zusammenschnürenden Spitzen von  $(L_2^n, F_2^n)$  zurückzuführen. Diese bewirken, dass das Terminierungskriterium  $T_{VO}$  sehr häufig bereits nach  $t$  erzeugten Populationen erfüllt ist. Dies ist genau dann der Fall, wenn der Genetische Algorithmus nicht in der Lage ist, die Zielfunktion innerhalb der ersten  $t$  erzeugten Populationen um 5% zu verbessern.

Die durchschnittliche Anzahl der von der Feinoptimierung benötigten Zielfunktionsevaluierungen  $oa$  [MS] sowie die erreichte Approximationsgüte fallen bei Experimentreihe  $E_4$  ähnlich aus wie in den drei vorangegangenen Experimentreihen. Obwohl sich  $(L_2^n, F_2^n)$  als sehr schwierig für die Voroptimierung herausgestellt hat, bereitet diese Problemstellung der zur Feinoptimierung eingesetzten Mustersuche offensichtlich keinerlei Schwierigkeiten.

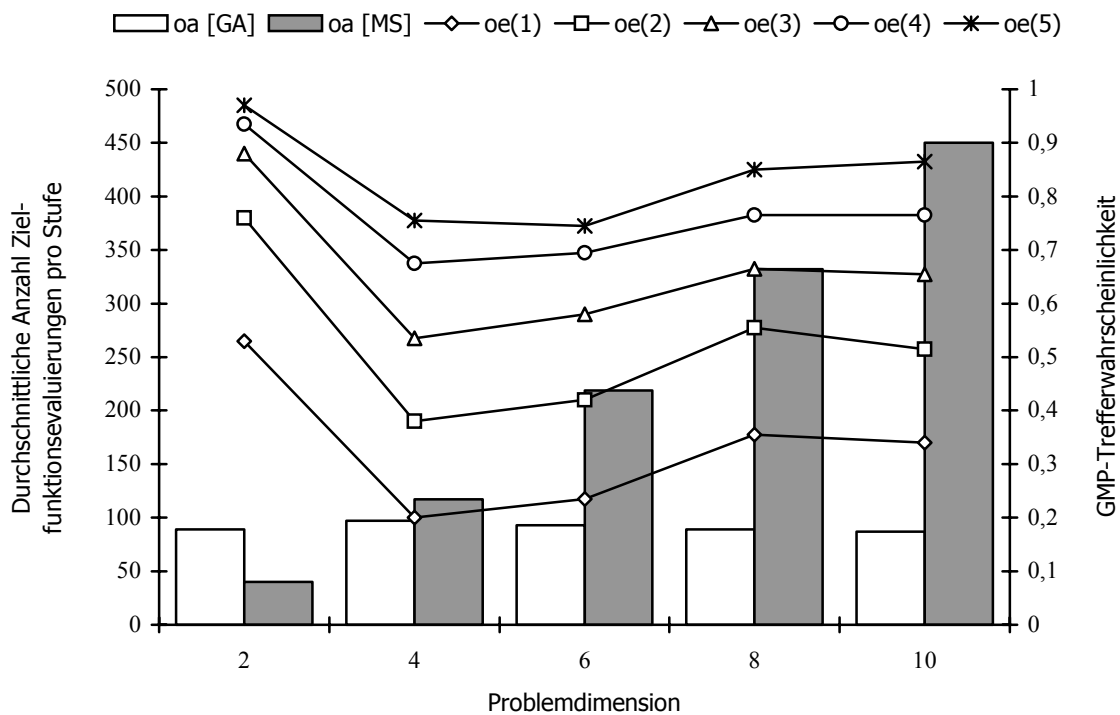
VR arbeitet ebenfalls erfolgreich und stellt sicher, dass heterogene Folgen von Optimumpunkten erzeugt werden. Nach 10 Optimierungsstufen hat  $os_{ms}$  bereits durchschnittlich 7 der insgesamt 10 verschiedenen Maximumpunkte von  $(L_2^n, F_2^n)$  lokalisiert, wobei die markanten Extremstellen bereits in den frühen Optimierungsstufen aufgefunden werden.

In Experimentreihe  $E_5$  wurde mit dem mehrstufigen Optimierungsverfahren  $os_{ms}$  ebenfalls das Testproblem  $(L_2^n, F_2^n)$ ;  $n \in \{2, 4, 6, 8, 10\}$  maximiert. Im Unterschied zu  $E_4$  wurde jedoch bei wachsender Problemdimension mit einem weitaus geringeren Voroptimierungsaufwand gearbeitet. Dies wurde erreicht, indem die Kontrollparameter  $mc$ ,  $pg$  und  $t$  nicht wie in  $E_4$  mit wachsender Problemdimension erhöht, sondern dimensionsunabhängig auf die in Tabelle 5.3.3.6 dargestellten Werte gesetzt wurden.

n	2	4	6	8	10
mc	40				
pg	20				
t	2				

*Tabelle 5.3.3.6: Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus in Experimentreihe  $E_5$*

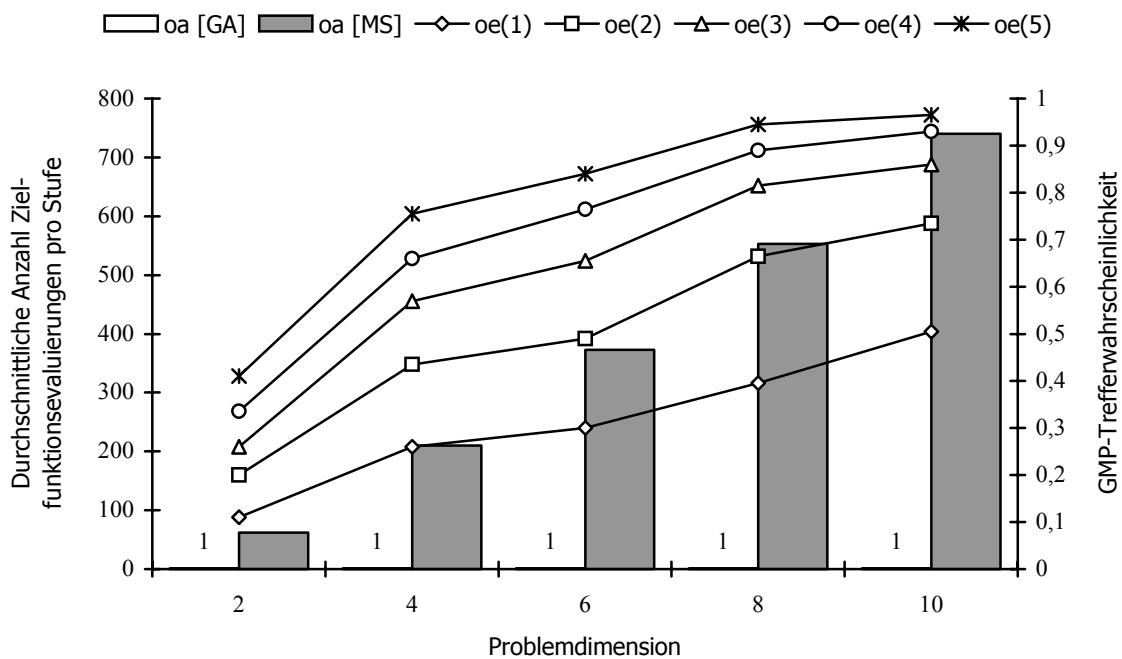
In Diagr. 5.3.3.6 sind die Ergebnisse von Experimentreihe  $E_5$  zusammengefasst. Auch hier wurde zu jeder der fünf betrachteten Problemdimensionen  $n \in \{2, 4, 6, 8, 10\}$  jeweils ein Mehrfachexperiment durchgeführt. Trotz des drastisch verringerten Voroptimierungsaufwands ändert sich der Optimierungserfolg bei hoher Problemdimension im Vergleich zu den in Diagr. 5.3.3.5 dargestellten Ergebnissen aus Experimentreihe  $E_4$  nur unwesentlich. Bei der Voroptimierung von  $(L_2^n, F_2^n)$  scheint der Genetische Algorithmus also bei hoher Problemdimension weitgehend wirkungslos zu bleiben.



Diagr. 5.3.3.6: In Experimentreihe  $E_5$  ermittelter Optimierungserfolg und -aufwand

In Experimentreihe  $E_6$  wurde das Testproblem  $(L_2^n, F_2^n)$ ;  $n \in \{2, 4, 6, 8, 10\}$  ein weiteres Mal mit dem mehrstufigen Optimierungsverfahren  $os_{ms}$  maximiert. Im Unterschied zu  $E_5$  wurde jedoch diesmal gänzlich auf den Genetischen Algorithmus zur Voroptimierung verzichtet und als Ersatz dafür die Mustersuche jeweils an einem zufällig ausgewählten Punkt gestartet.

Die Ergebnisse von Experimentreihe  $E_6$  sind in Diagr. 5.3.3.7 zusammengefasst. Auf den ersten Blick überraschend steigt der Optimierungserfolg in sämtlichen Optimierungsstufen mit wachsender Problemdimension streng monoton an und übertrifft ab Problemdimension 6 sogar die mit dem Genetischen Algorithmus in Experimentreihe  $E_4$  und  $E_5$  erreichten Erfolgskurven. Dieses Verhalten ist dadurch zu erklären, dass sich bei Problemstellung  $(L_2^n, F_2^n)$  der Einzugsbereich des globalen Maximumpunkts mit wachsender Problemdimension zunehmend vergrößert. Bei Problemdimension 10 macht dieser Einzugsbereich bereits etwa 50 Prozent des Gesamtlösungsraums aus. Wird in diesem Fall die Mustersuche an einem zufällig gewählten Punkt des Lösungsraums gestartet, liegt die Wahrscheinlichkeit, den globalen Maximumpunkt zu finden also ungefähr bei 0.5 (vergleiche dazu auch den Wert der in Diagr. 5.3.3.7 dargestellten Kurve  $oe(1)$  bei Problemdimension 10). Im Gegensatz zur Mustersuche hebt sich für den Genetischen Algorithmus der Einzugsbereich des globalen Maximumpunkts jedoch nicht sonderlich von den Einzugsbereichen der übrigen 9 Maximumpunkte ab, da sämtliche Zielfunktionsverläufe über diesen Einzugsbereichen Quasihyperbenen auf gleichem Höhenniveau darstellen, aus denen jeweils nadelförmige Spitzen herausragen.



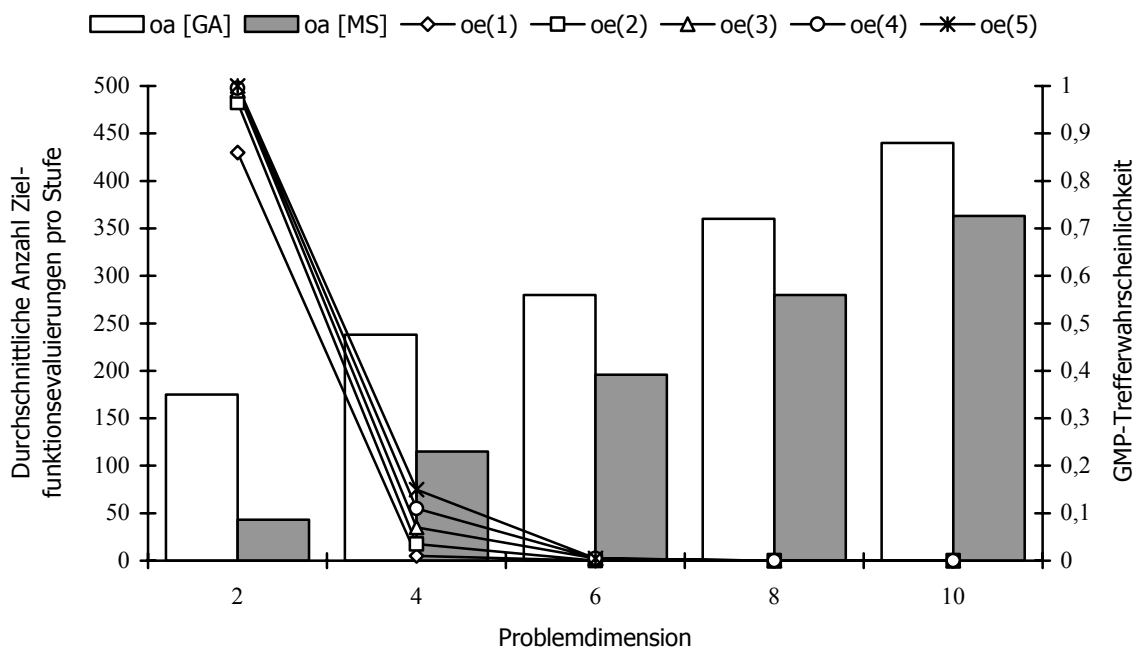
Diagr. 5.3.3.7: In Experimentreihe  $E_6$  ermittelter Optimierungserfolg und -aufwand

Die Ergebnisse aus den drei Experimentreihen  $E_4$ ,  $E_5$  und  $E_6$  zeigen deutlich, dass der Erfolg direkter naturanaloger Voroptimierungsverfahren stark davon abhängt, ob die vereinfachenden Annahmen aus Abschnitt 2.3.2 erfüllt sind. Im 2-dimensionalen Fall der Problemstellung  $(L_2^n, F_2^n)$ , bei der sämtliche dieser Annahmen zutreffen, arbeiten direkte naturanaloge Voroptimierungsverfahren effizient und erfolgreich. Wird auch nur eine dieser Annahmen verletzt, wie das bei  $(L_2^n, F_2^n)$  mit wachsender Problemdimension zunehmend der Fall ist, kann das dazu führen, dass direkte naturanaloge Voroptimierungsverfahren stark in ihrer Wirkungsweise beeinträchtigt oder sogar irreführt werden.

### 5.3.3.3 Ergebnisse zu Testproblem $(L_3^n, F_3^n)$

In Experimentreihe  $E_7$  wurde mit  $(L_3^n, F_3^n)$ ;  $n \in \{2, 4, 6, 8, 10\}$  ein Testproblem maximiert, bei dem bei hoher Problemdimension zwei der vereinfachenden Annahmen aus Abschnitt 2.3.2 nicht mehr erfüllt werden. Die hier verwendeten Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus entsprechen den in Tabelle 5.3.3.5 dargestellten Einstellungen aus Experimentreihe  $E_4$ . Analog zu den Experimentreihen  $E_4$ ,  $E_5$  und  $E_6$  wurde zu jeder der fünf betrachteten Problemdimensionen  $n \in \{2, 4, 6, 8, 10\}$  jeweils ein Mehrfachexperiment durchgeführt, das sich aus 200 Optimierungsläufen mit  $os_{ms}$  zusammensetzte, die jeweils mit unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators gestartet wurden. Jeder Optimierungslauf mit  $os_{ms}$  wurde nach genau 20 Optimierungsstufen abgebrochen.

Wie in Tabelle 5.3.2.5 dargestellt weist die Ergebnisoberfläche von  $F_3^n$  über  $L_3^n$  genau einen globalen Maximumpunkt auf. Dieser ist von einer Vielzahl lokaler Maximumpunkte umgeben, die alle den gleichen Zielfunktionswert besitzen, welcher gegenüber dem globalen Maximum relativ klein ausfällt. Mit wachsender Problemdimension verformt sich der Ergebnisverlauf der Zielfunktion über dem globalen Maximumpunkt zunehmend zu einer steil nach oben ragenden Nadel, während die Anzahl der lokalen Maximumpunkte exponentiell ansteigt. Damit sind zwei der vereinfachenden Annahmen aus Abschnitt 2.3.2 nicht mehr erfüllt. Die in Diagr. 5.3.3.8 zusammengefassten Ergebnisse von Experimentreihe  $E_7$  zeigen, dass unter diesen Voraussetzungen eine effiziente Lösung von  $(L_3^n, F_3^n)$  bei den höheren Problemdimensionen für das mehrstufige Optimierungsverfahren  $os_{ms}$  nicht mehr möglich ist.



Diagr. 5.3.3.8: In Experimentreihe  $E_7$  ermittelter Optimierungserfolg und -aufwand

Bei der Durchführung von Experimentreihe  $E_7$  wurden bezüglich des durchschnittlichen Netto-Optimierungsaufwands und der erzielten Approximationsgüte keine wesentlichen Unterschiede zu den in Experimentreihe  $E_4$ ,  $E_5$  und  $E_6$  erzielten Optimierungsergebnissen festgestellt. Beim Optimierungserfolg ergab sich jedoch, wie aus Diagr. 5.3.3.8 deutlich hervorgeht, ein drastischer Rückgang. Lediglich bei Problemdimension 2 arbeitet  $os_{ms}$  erfolgreich, da in diesem Fall die vereinfachenden Annahmen aus Abschnitt 2.3.2 noch alle erfüllt sind (siehe dazu auch die in Tabelle 5.3.2.5 dargestellte Ergebnisoberfläche von  $F_3^2$  über  $L_3^2$ ). Ab Dimension 4 jedoch fällt der Optimierungserfolg drastisch gegen Null ab. Wegen der im Vergleich zu  $(L_2^n, F_2^n)$  sehr hohen Anzahl lokaler Maximumpunkte bringt hier auch die Durchführung weiterer Optimierungsstufen keine wesentliche Verbesserung des Optimierungserfolgs.  $(L_3^n, F_3^n)$  stellt somit ein Optimierungsproblem dar, bei dem  $os_{ms}$  im hochdimensionalen Fall offensicht-



lich versagt. Dass es solche Optimierungsprobleme gibt, ist nicht weiter verwunderlich, da es sich bei der globalen Parameteroptimierung um eine NP-harte Problemstellung handelt. Für den praktischen Verfahrenseinsatz bleibt zu hoffen, dass konstruierte Problemstellungen wie  $(L_3^n, F_3^n)$  nicht allzu häufig auftreten.

### 5.3.3.4 Ergebnisse zu Testproblem $(L_4^n, F_4^n)$

In den drei folgenden Experimentreihen  $E_8$ ,  $E_9$  und  $E_{10}$  wurde mit  $(L_4^n, F_4^n)$ ;  $n \in \{2,3,4\}$  eine Problemstellung maximiert, bei der zwar die vereinfachenden Annahmen aus Abschnitt 2.3.2 weitgehend erfüllt sind, die sich aber wie bereits in Abschnitt 5.1.5 beschrieben für Evolutionäre Algorithmen als sehr schwierig (irreführend) herausgestellt hat. Analog zu den vorangegangenen Experimentreihen wurde zu jeder betrachteten Problemdimension jeweils ein Mehrfachexperiment durchgeführt, das sich aus 200 Optimierungsläufen mit  $os_{ms}$  zusammensetzte, die jeweils mit unterschiedlicher Initialisierung des Pseudo-Zufallszahlengenerators gestartet wurden. Jeder Optimierungslauf mit  $os_{ms}$  wurde nach genau 20 Optimierungsstufen abgebrochen.

In Experimentreihen  $E_8$  wurde innerhalb von  $os_{ms}$  wie bisher ein Genetischer Algorithmus zur Voroptimierung eingesetzt. Für dessen Kontrollparameter  $mc$ ,  $pg$  und  $t$  wurden die in Tabelle 5.3.3.5 dargestellten Einstellungen verwendet.

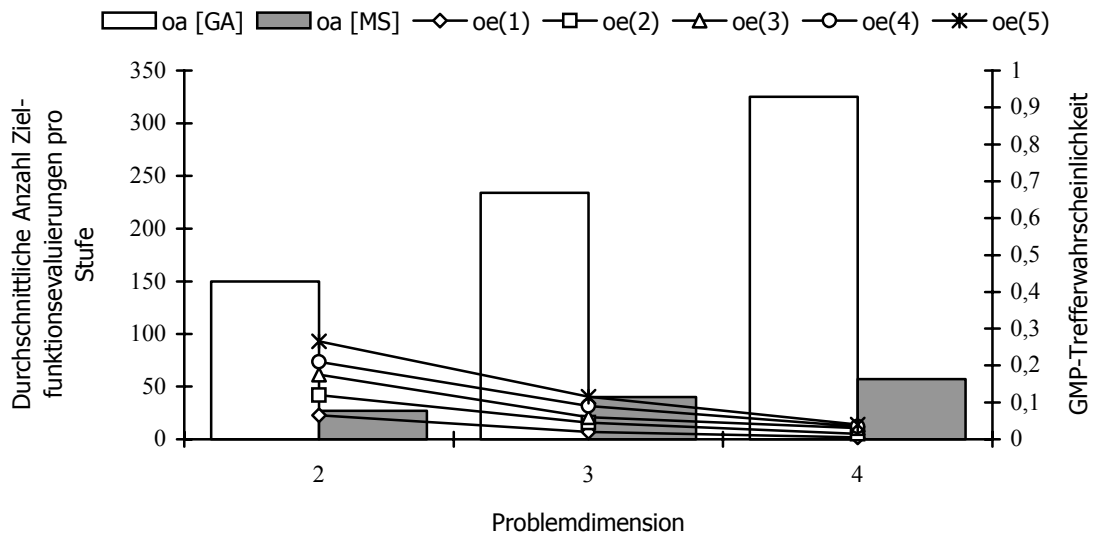
n	2	3	4
mc	40		
pg	20		
t	4	6	8

*Tabelle 5.3.3.7: Einstellungen der Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus in Experimentreihe  $E_8$*

Die in Diagr. 5.3.3.9 zusammengefassten Ergebnisse von Experimentreihe  $E_8$  zeigen, dass bei der hier vorliegenden irreführenden Problemstellung der Genetische Algorithmus offensichtlich große Schwierigkeiten hat, in den Einzugsbereich des globalen Maximumpunkts zu gelangen. Der Optimierungserfolg fällt hier bereits bei der 2-dimensionalen Problemstellung sehr niedrig aus und fällt mit wachsender Problemdimension weiter gegen Null ab. Auch durch weitere Optimierungsstufen kann der Optimierungserfolg nur unwesentlich gesteigert werden.

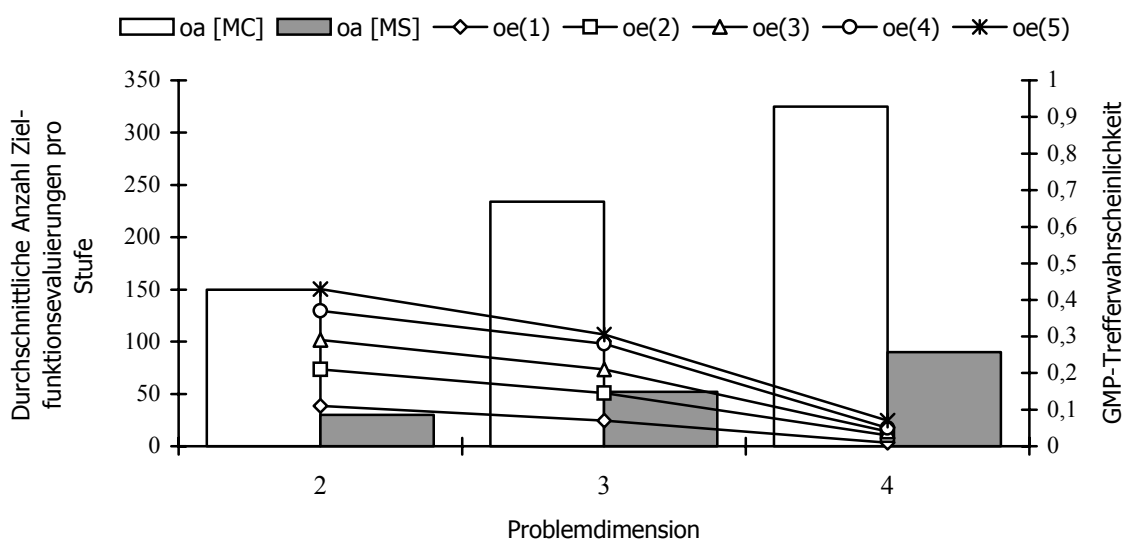
Um empirisch nachzuweisen, dass es sich bei  $(L_4^n, F_4^n)$  tatsächlich um ein für Genetische Algorithmen irreführendes Problem handelt, wurde in Experimentreihen  $E_9$  anstelle des Genetischen Algorithmus mit reiner Zufallssuche voroptimiert. Die Anzahl der dabei im Lösungsraum zufällig erzeugten Suchpunkte entsprach exakt dem Netto-Optimierungsaufwand des Genetischen

Algorithmus aus Experimentreihe  $E_8$ . Analog zu  $E_8$  wurde auch hier zu jeder der drei betrachteten Problemdimensionen  $n \in \{2,3,4\}$  jeweils ein Mehrfachexperiment durchgeführt.



Diagr. 5.3.3.9: In Experimentreihe  $E_8$  ermittelter Optimierungserfolg und -aufwand

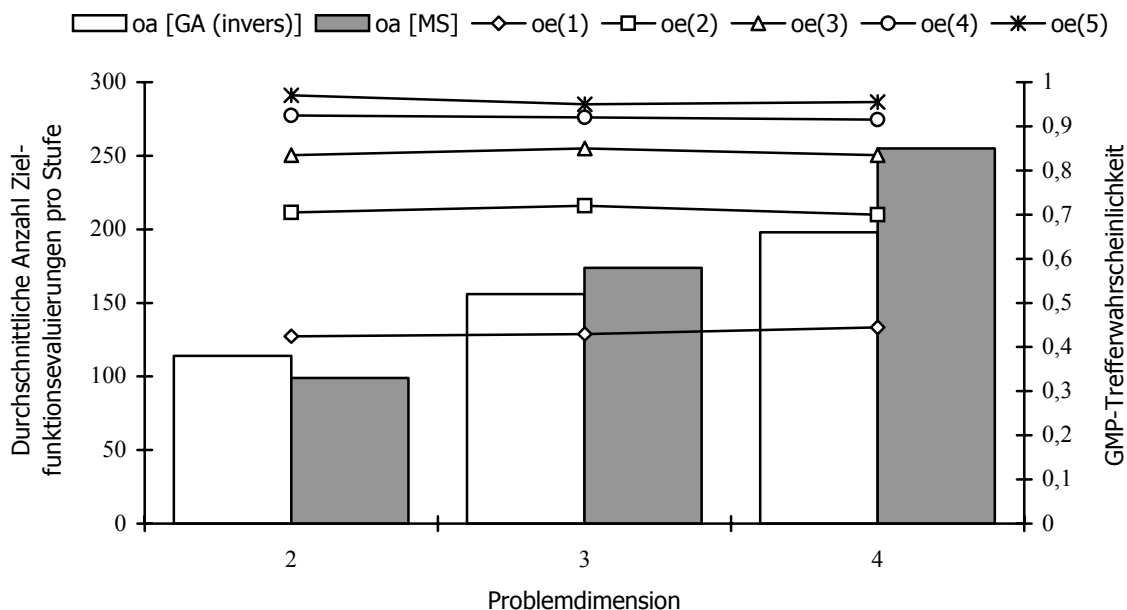
Die Ergebnisse von Experimentreihe  $E_9$  sind in Diagr. 5.3.3.10 zusammengefasst und bestätigen, dass  $(L_4^n, F_4^n)$  ein für Genetische Algorithmen irreführendes Problem darstellt. Wie nachzuweisen war, arbeitet die reine Zufallsuche bei exakt demselben Voroptimierungsaufwand im Durchschnitt erfolgreicher als der in Experimentreihe  $E_8$  zur Voroptimierung eingesetzte Genetische Algorithmus. Der zu beobachtende Erfolgsunterschied ist jedoch nicht sehr groß und nimmt mit wachsender Problemdimension zunehmend ab.



Diagr. 5.3.3.10: In Experimentreihe  $E_9$  ermittelter Optimierungserfolg und -aufwand

In Experimentreihe  $E_{10}$  wurde schließlich noch untersucht, welcher Optimierungserfolg sich bei Problemstellung  $(L_4^n, F_4^n)$  durch inverse Vorooptimierung mit dem Genetischen Algorithmus einstellt. Für die Kontrollparameter  $mc$ ,  $pg$  und  $t$  des Genetischen Algorithmus wurden wie in Experimentreihe  $E_8$  die in Tabelle 5.3.3.7 dargestellten Einstellungen verwendet. Analog zu den beiden vorangegangenen Experimentreihen  $E_8$  und  $E_9$  wurde zu jeder der drei betrachteten Problemdimensionen  $n \in \{2, 3, 4\}$  jeweils ein Mehrfachexperiment durchgeführt.

Die Ergebnisse von Experimentreihe  $E_{10}$  sind in Diagr. 5.3.3.11 zusammengefasst. Bei exakt derselben Parametrisierung wie in Experimentreihe  $E_8$  findet der zur Vorminimierung eingesetzte Genetische Algorithmus den Einzugsbereich des globalen Maximumpunkts bereits in der ersten Optimierungsstufe mit einer Wahrscheinlichkeit von knapp unter 0.5. Dieses gegenüber  $E_8$  und  $E_9$  sehr gute Ergebnis lässt sich dadurch erklären, dass der Genetische Algorithmus bei der Vorminimierung gegen den kreisförmigen Bereich global-minimaler Lösungen um den Einzugsbereich des globalen Maximumpunkts konvergiert und in diesem schließlich mit einer Wahrscheinlichkeit von etwa 0.5 terminiert. Die Kurven  $oe(i)$ ,  $i \in \{1, \dots, 5\}$  aus Diagr. 5.3.3.11 zeigen außerdem, dass der Optimierungserfolg bei wachsender Problemdimension gehalten werden kann. Darüber hinaus wird mit jeder weiteren Optimierungsstufe eine deutliche Steigerung des Optimierungserfolgs erzielt, bis schließlich nach der fünften Optimierungsstufe die Wahrscheinlichkeit für das Auffinden des globalen Maximumpunkts bereits sehr nahe bei 1 liegt.



Diagr. 5.3.3.11: In Experimentreihe  $E_{10}$  ermittelter Optimierungserfolg und -aufwand

Die in Diagr. 5.3.3.11 dargestellte Erfolgskurve  $oe(1)$  kann noch deutlich angehoben werden, wenn nach Abschluss der Vorminimierung in einem eng begrenzten Bereich um die gefundene minimale Lösung noch einmal eine Maximierung vorgenommen wird. Bei Problemstellung  $(L_4^n, F_4^n)$  lässt sich dadurch mit dem relativ geringen Aufwand von einigen Dutzend Zielfunktionsevaluierungen die Erfolgskurve  $oe(1)$  fast auf den Maximalwert 1 anheben.

Abschließend sei angemerkt, dass von den in Abschnitt 5.1.4 beschriebenen Maßnahmen zur Reduktion des Optimierungsaufwands bei der Durchführung der oben beschriebenen zehn Experimentreihen ausschließlich die in Abschnitt 5.1.4.1 beschriebene Methode zur Vermeidung von Reevaluation angewendet wurde. Durch Mehrfachstart der Feinoptimierung sowie Verwendung approximierter Zielfunktionswerte bei der Voroptimierung lässt sich der Optimierungsaufwand von  $os_{ms}$  noch weiter reduzieren. Quantitative Ergebnisse diesbezüglich finden sich in [SSBH96] und [Syrj97].

## 5.4 Zusammenfassung

Im Folgenden werden noch einmal die wesentlichen Eigenschaften der in diesem Kapitel beschriebenen direkten Verfahren zur effizienten Parameteroptimierung von Simulationsmodellen zusammengefasst. Diese Verfahren stellen Weiterentwicklungen bereits existierender auf Prinzipien der Natur basierender Strategien zur globalen und lokalen Suche dar. Die Grundidee zu deren Weiterentwicklung basiert auf der Unterteilung des Optimierungsprozesses in die beiden Phasen Vor- und Feinoptimierung. Aufgabe der Voroptimierung ist die möglichst umfassende Exploration des gesamten Suchraums nach Regionen, die mit großer Wahrscheinlichkeit global-optimale Lösungen enthalten. Ausgehend vom Ergebnis der Voroptimierung besteht die Aufgabe der anschließenden Feinoptimierung darin, die optimale Lösung in einer solchen viel versprechenden Region möglichst effizient und exakt zu lokalisieren. Das daraus resultierende Gesamtverfahren, das die Stärken der globalen und lokalen Optimierung in sich vereint, stellt einen deutlichen Fortschritt gegenüber den bereits existierenden atomaren, auf genau einem Optimierungsprinzip basierenden Methoden dar. Im Folgenden werden noch einmal die Eigenschaften zusammengefasst, welche die kombinierte 2-Phasen Optimierung gegenüber den bisherigen Ansätzen zur direkten Optimierung auszeichnen:

- Hoher Optimierungserfolg

Falls die vereinfachenden Annahmen aus Abschnitt 2.3.2 erfüllt sind, arbeiten die zur Voroptimierung eingesetzten naturanalogen Verfahren zur globalen Optimierung i.a. sehr erfolgreich und liefern als Ergebnis einen Suchpunkt, der mit hoher Wahrscheinlichkeit im Einzugsbereich einer global-optimalen Lösung liegt. Des Weiteren lässt sich durch inverse Voroptimierung wirkungsvoll verhindern, dass irreführende Probleme als solche unerkannt bleiben.

- Hohe Ergebnisqualität

Durch das der Voroptimierung nachgeschaltete deterministische Feinoptimierungsverfahren lassen sich optimale Lösungen mit einer vom Anwender vorgebbaren Genauigkeit lokalisieren.

- Hohe Effizienz

Durch Methoden zur Reduktion des Vor- und Feinoptimierungsaufwands sowie durch geeignete Verfahrensparametrisierung und rechtzeitiges Umschalten von Vor- auf Feinoptimierung werden unnötige Zielfunktionsevaluierungen weitgehend vermieden und damit hohe Effizienz erreicht.

Der Gedanke, hybride Optimierungsverfahren zu entwickeln, ist nicht neu. Kombinationen verschiedener lokaler und globaler Suchstrategien wurden bereits in [Ackl87], [Hard93], [MoHa90], [MüSB91], [ReFl96] und [Wend95] vorgeschlagen und untersucht, wobei in der Regel deutliche Leistungssteigerungen festgestellt werden konnten. Die Entwicklung der in diesem Kapitel vorgestellten Optimierungskomponente endet jedoch nicht bei einer weiteren Variante hybrider Optimierungsmethoden, sondern setzt darüber hinaus deren hohe Leistungsfähigkeit gezielt zur Realisierung mehrstufiger Optimierungsverfahren ein. Bei der mehrstufigen Optimierung geht es nicht mehr wie bisher um das Auffinden genau einer, sondern um die systematische Ermittlung der markantesten Extremstellen einer vorgegebenen Zielfunktion. Den Kernbestandteil eines mehrstufigen Optimierungsverfahrens  $os_{ms}$  stellt aufgrund ihrer vorteilhaften Eigenschaften eine kombinierte 2-Phasen Optimierungsstrategie  $os_{2p}$  dar. Ein weiterer wichtiger Bestandteil von  $os_{ms}$  ist die Methode zur Vermeidung von Reexploration, mit der sichergestellt wird, dass möglichst heterogene Folgen optimaler Lösungen erzeugt werden. Durch die mehrstufige Optimierung, bei der jede weitere Optimierungsstufe einen monotonen Anstieg des Optimierungserfolgs nach sich zieht, lassen sich auch sehr schwierige Problemstellungen bearbeiten, die mit einer kombinierten 2-Phasen Strategie auf Anhieb nicht zu bewältigen wären. Die mehrstufige Optimierung stellt daher eine deutlich verbesserte und systematischere Methodologie zur globalen Optimierung dar, als die bisher gebräuchliche "einstufige" Vorgehensweise.

Durch die in Abschnitt 5.2 vorgestellte Optimierungskomponente werden die in Abschnitt 5.1 beschriebenen direkten Optimierungsverfahren dem Anwender zur Verfügung gestellt. Insgesamt wurde damit eine komfortable Arbeitsumgebung zum Experimentieren mit Modelloptimierungsverfahren geschaffen, welche die zur Modelloptimierung notwendigen Experimentabfolgen vollautomatisch steuert und darüber hinaus Möglichkeiten zur umfassenden empirischen Leistungsbewertung der implementierten Optimierungsverfahren bietet. Die in Abschnitt 5.3 beschriebenen Experimentreihen zu vier unterschiedlichen mathematischen Testproblemen geben einen umfassenden Einblick in das Leistungsvermögen der angebotenen Optimierungsverfahren und darüber hinaus einige Anregungen zum weiteren Ausbau der theoretischen Grundlagen zu direkten Optimierungsverfahren. Auch der Anwender aus der Praxis erhält nützliche Hinweise, um zu einer vorgegebenen Problemstellung

- ein geeignetes Optimierungsverfahren auszuwählen,
- dessen Parametrisierung vorzunehmen sowie
- den zu erwartenden Optimierungserfolg, die Ergebnisqualität und den damit verbundenen Rechenaufwand abzuschätzen.

Aus den Ergebnissen der in Abschnitt 5.3 beschriebenen empirischen Leistungsanalyse lässt sich darüber hinaus schließen, dass die in diesem Kapitel vorgestellte Optimierungskomponente in der Lage ist, modellbasierte Zielfunktionen, die den vereinfachenden Annahmen aus Abschnitt 2.3.2 genügen und innerhalb von zehn Minuten ausgewertet werden können, erfolgreich und in akzeptabler Zeit zu optimieren. Ausführliche Beschreibungen von Modelloptimierungsexperimenten und den dabei erzielten Ergebnissen finden sich in [Syrj97] und [Info99].

## 5.5 Mögliche Weiterentwicklungen

Abschließend sei noch auf folgende Erweiterungsmöglichkeiten für die in diesem Kapitel beschriebene Optimierungskomponente hingewiesen:

- Behandlung von Kombinatorikproblemen

Bei der Modelloptimierung sind neben den bislang betrachteten reellwertigen Parameteroptimierungsproblemen auch Aufgabenstellungen denkbar, die in den Bereich der Kombinatorikprobleme<sup>12</sup> fallen. Zu deren Lösung können Genetische Algorithmen und das Simulated Annealing als universell anwendbare Verfahren grundsätzlich beibehalten werden. Die Mustersuche, die speziell auf die reellwertige Parameteroptimierung zugeschnitten ist, muss jedoch durch einen auf das jeweils vorliegende Kombinatorikproblem angepassten Greedy-Algorithmus ersetzt werden.

- Parallelisierung des Optimierungsprozesses

Gerade die zur Voroptimierung eingesetzten globalen Optimierungsverfahren bieten vielfältige Möglichkeiten zur Parallelisierung. Durch gezieltes Ausnutzen dieser Möglichkeiten lässt sich die zur Durchführung der Optimierung benötigte Rechenzeit in der Regel beträchtlich reduzieren.

- Nachweis von Periodizität

Durch frühzeitiges Erkennen von Periodizität ließe sich bei der mehrstufigen Optimierung von Zielfunktionen mit periodisch verlaufender Ergebnisoberfläche<sup>13</sup> in hohem Maße Optimierungsaufwand einsparen.

- Integration weiterer Feinoptimierungsverfahren

Zur Feinoptimierung stellt die in Abschnitt 5.2 beschriebene Optimierungskomponente bislang ausschließlich die aus den sechziger Jahren stammende Mustersuche von Hooke und Jeeves [HoJe61] zur Verfügung. Dieses weit verbreitete deterministische Hill-Climbing Verfahren wurde deshalb ausgewählt, weil es äußerst effizient arbeitet, relativ hohe Unempfindlichkeit gegenüber stochastisch gestörten Zielfunktionswerten aufweist und in der Lage ist, Extremstellen mit einer vom Anwender einstellbaren Genauigkeit zu lokalisieren. Eine gute Ergänzung zur Mustersuche würde beispielsweise der in den neunziger Jahren entwickelte SPSA (Simultaneous Perturbation Stochastic Approximation) Algorithmus darstellen. Dabei handelt es sich ebenfalls um ein direktes Hill-Climbing Verfahren, das aber im Gegensatz zur Mustersuche auf stochastischen Suchoperatoren basiert. Detaillierte Beschreibungen des sehr effizient arbeitenden SPSA Algorithmus finden sich in [Spal98] und [Spall99].

---

<sup>12</sup> z.B. die Optimierung der Modellstruktur

<sup>13</sup> die Zielfunktionen aus Tabelle 5.3.2.1 und 5.3.2.5 weisen beispielsweise eine periodisch verlaufende Ergebnisoberfläche auf

- Weiterverwertung der Optimierungstrajektorie

Die während des Optimierungsprozesses erzeugte Optimierungstrajektorie enthält neben den ermittelten Optimumpunkten noch eine Vielzahl an weiteren Informationen zum bearbeiteten Optimierungsproblem. Diese aufwändig gesammelten Daten sollten nicht einfach verworfen werden, sondern anderen Spezialkomponenten beispielsweise zur Sensitivitätsanalyse oder Metamodellierung<sup>14</sup> zur Verfügung gestellt werden.

---

<sup>14</sup> Erste Auswertungs- und Analyseergebnisse der beim Optimierungsprozess anfallenden Daten durch maschinelle Neuro-Fuzzy-Lernverfahren wurden bereits in [HuSS93] und [HuBe95] vorgestellt.





# Projektbeschreibungen

---

In diesem Kapitel wird über drei am Institut für Rechnerentwurf und Fehlertoleranz durchgeführte Forschungsprojekte berichtet, in denen die in dieser Arbeit vorgestellten Technologien und Methoden konkret angewendet wurden. Damit sollen beispielhaft einige der vielfältigen Einsatzmöglichkeiten von Web- und Komponenten-Technologien in der Modellierung und Simulation und die sich daraus ergebenden Vorteile aufgezeigt werden. Bei den im Folgenden beschriebenen Forschungsprojekten geht es thematisch um den komponentenorientierten Aufbau eines Web-basierten Werkzeugs zur dynamischen Prozessoptimierung, die Online-Animation direkter Optimierungsverfahren sowie um die verteilte Simulation und Visualisierung künstlichen Lebens.

## 6.1 Komponentenorientierter Aufbau eines Web-basierten Werkzeugs zur dynamischen Prozessoptimierung

Das in diesem Abschnitt beschriebene Forschungsprojekt "dynamische Prozessoptimierung" wurde im Rahmen des Forschungsschwerpunkts "Informationslogistik" der Universität Karlsruhe durchgeführt. An diesem Forschungsschwerpunkt waren neben dem Institut für Rechnerentwurf und Fehlertoleranz vier weitere Institute<sup>1</sup> aus unterschiedlichen Fakultäten der Universität Karlsruhe beteiligt.

Die Zielsetzung des Forschungsprojekts "dynamische Prozessoptimierung", bestand darin, komplexe Prozesse aus der branchenübergreifenden kooperativen Planung durch Einsatz von Methoden aus der Modellierung, Simulation und Optimierung wirkungsvoll zu unterstützen. Dabei wurde angestrebt, eine optimale Allokation von Aufträgen und Bearbeitungsinstanzen unter Berücksichtigung einzuhaltender Randbedingungen zu erreichen. Probleme dieser Art sind i.a. sehr komplex und zeichnen sich durch dynamische Einflussgrößen und Randbedingungen aus, die laufenden Veränderungen unterliegen. Die Instabilität der Problemstellung erfordert häufige Revisionen, wobei existierende Lösungen kontinuierlich überarbeitet und möglichst schnell angepasst werden müssen [GrKR01].

Abb. 6.1.1 gibt einen Überblick über die bei der dynamischen Prozessoptimierung durchzuführenden Arbeitsschritte. Im ersten Arbeitsschritt wird ein deskriptives von den beteiligten Pro-

---

<sup>1</sup> Dabei handelt es sich um das Institut für Rechneranwendung in Planung und Konstruktion, das Institut für Industrielle Bauproduktion, das Institut für Prozessrechentechnik und Robotik und das Institut für Werkzeugmaschinen und Betriebstechnik.

jektpartnern erstelltes Prozessmodell in ein erweitertes Petrinetzmodell überführt, das im Gegensatz zum deskriptiven Ursprungsmodell maschinell ausgewertet werden kann. Die Konvertierung des deskriptiven Modells in ein erweitertes Petrinetzmodell basiert auf einem Satz vordefinierter Petrinetzmodule zur Geschäftsprozessmodellierung [Müll97] und erfolgt automatisch. Bei der hier verwendeten Art von erweiterten Petrinetzen handelt es sich um so genannte generalisierte stochastische Petrinetze (GSPN), die neben zeitlosen auch zeitbehaftete Transitionen zur Verfügung stellen und damit die Modellierung dynamischer Systeme ermöglichen.

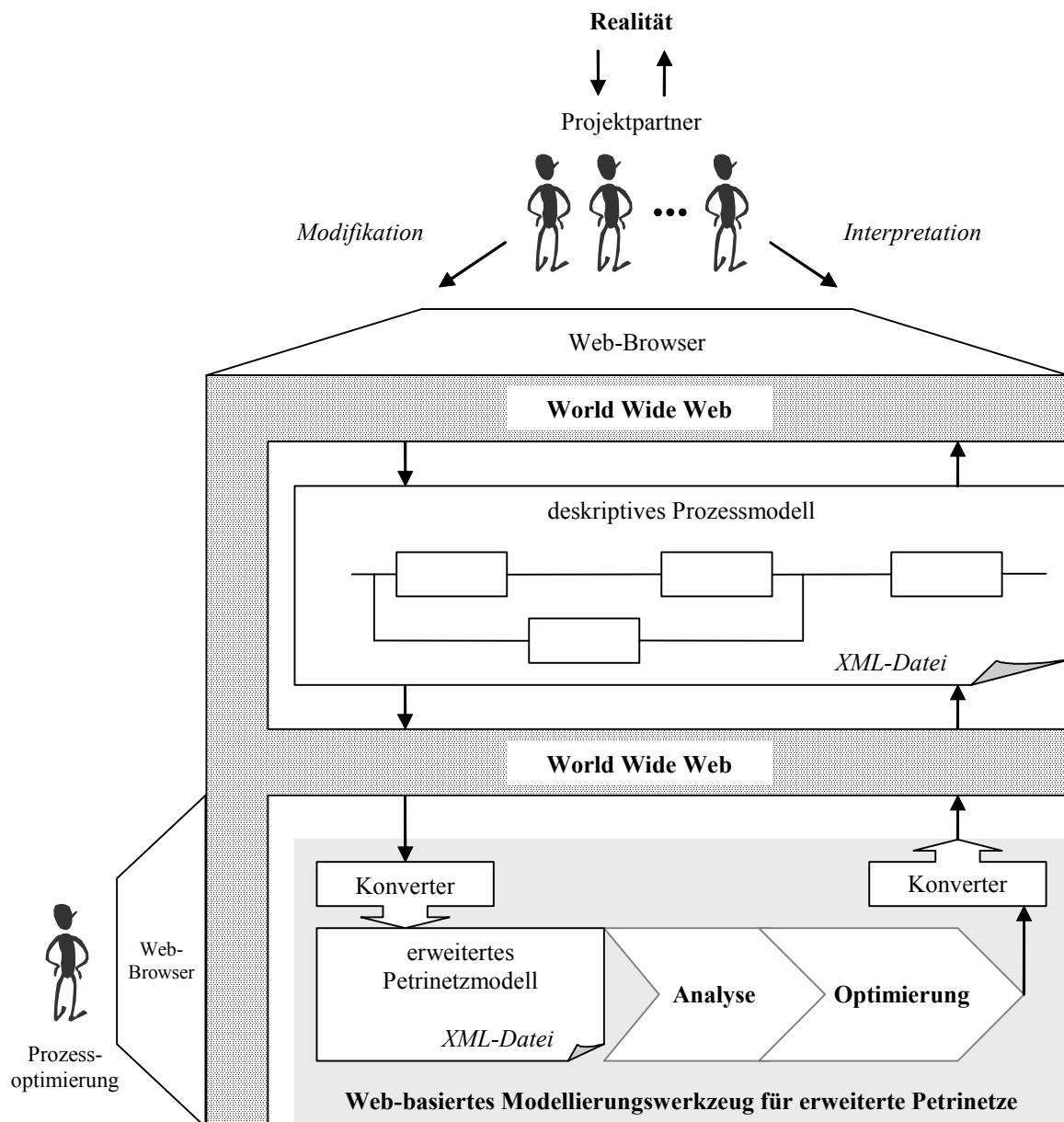


Abb. 6.1.1: Arbeitsschritte bei der dynamischen Prozessoptimierung

Nach seiner Erzeugung wird das GSPN-Modell im zweiten Arbeitsschritt hinsichtlich verschiedener Struktureigenschaften analysiert, um dessen grundsätzliche Ablauffähigkeit und Korrektheit zu überprüfen. Das Ziel des darauf folgenden dritten Arbeitsschrittes ist es, das Petri-

netzmodell so zu parametrisieren, dass es in Bezug auf bestimmte Zielgrößen ein optimales Modellverhalten aufweist. Das optimierte Prozessmodell wird schließlich wieder den beteiligten Projektpartnern zur Verfügung gestellt, wobei es von der Petrinetzdarstellung wieder in die deskriptive von den Projektpartnern leichter zu interpretierende Repräsentation überführt wird.

Wie bereits angedeutet unterliegt das deskriptive Prozessmodell sehr häufigen Änderungen, auf die in der Regel sehr schnell reagiert werden muss. Um dies zu erreichen, mussten die in Abb. 6.1.1 dargestellten Arbeitsschritte automatisiert und nicht zu vermeidende manuelle Eingriffe zwischen den einzelnen Arbeitsschritten so kurz wie möglich gehalten werden. Weitere wichtige Projektforderungen bestanden zum einen in der Web-Fähigkeit der zu realisierenden Werkzeuge, zum anderen in einem einheitlichen Beschreibungsformat für die zwischen den Projektpartnern ausgetauschten Dokumente. Aufgrund seiner in Abschnitt 2.4.3.3 beschriebenen vorteilhaften Eigenschaften wurde dazu XML verwendet.

Für die innerhalb des Forschungsprojekts "dynamische Prozessoptimierung" zu bearbeitende Aufgabenstellung hatten sich existierende Werkzeuge zu GSPN als nicht ausreichend qualifiziert herausgestellt. Die Gründe dafür lagen vor allem darin, dass die beiden zentralen Forderungen nach Web-Fähigkeit und Optimierungsfunktionalität nicht erfüllt wurden. Da eine entsprechende Erweiterung dieser Werkzeuge aufgrund ihres monolithischen Softwareaufbaus<sup>2</sup> wenig Erfolg versprechend war, wurde eine von Grund auf neue komponentenorientierte Werkzeugarchitektur konzipiert, die im Detail in Abschnitt 4.2.2 beschrieben ist. Diese Architektur gestattete eine flexible Unterteilung der zur Lösung der Projektaufgabe erforderlichen Gesamtfunktionalität in überschaubare durch standardisierte Schnittstellen klar voneinander getrennte Einzelkomponenten. Diese Einzelkomponenten und deren Abhängigkeiten untereinander sind in Abb. 6.1.2 dargestellt. Bei den grau unterlegten Rechtecken handelt es sich um die innerhalb des Forschungsprojekts vorgenommenen Neuentwicklungen. Die weißen Rechtecke stellen Komponenten dar, die unter Ausnutzung der in Abschnitt 4.2.5 beschriebenen Integrationsmöglichkeiten aus existierenden Werkzeugen übernommen werden konnten.

Bei den integrierten Komponenten handelt es sich zunächst einmal um Module zur analytischen und simulativen Auswertung von GSPN, die aus den bewährten GSPN-Werkzeugen GreatSPN [PN3], DSPNexpress [PN4] und TimeNet [PN5] entnommen wurden. Wie in Abb. 6.1.2 dargestellt unterstützt jedes dieser Modellierungswerkzeuge ein eigenes proprietäres Modellbeschreibungsformat. Um die dadurch hervorgerufene Inkompatibilität aufzuheben, wurde ein einheitliches XML-basiertes Modellbeschreibungsformat für GSPN entwickelt [Schi02], das sich weitgehend an Vorschlägen orientiert<sup>3</sup>, die bislang im Rahmen der von der ISO vorangetriebenen Standardisierung eines XML-basierten Modellaustauschformats für erweiterte Petrinetze unterbreitet wurden [PN2]. Der Austausch von Modellbeschreibungen zwischen den integrierten GSPN-Werkzeugen wird durch bidirektionale Konverter ermöglicht, welche die proprietären Modellbeschreibungsformate von GreatSPN, DSPNexpress und TimeNet in das XML-basierte Modellbeschreibungsformat umwandeln und umgekehrt<sup>4</sup>. Damit steht dem Modellierer die zurzeit wohl kompletteste Auswahl analytischer und simulativer Auswertungsmethoden für GSPN innerhalb eines Werkzeugs zur Verfügung.

---

<sup>2</sup> siehe dazu auch Abschnitt 4.2.1

<sup>3</sup> dabei handelt es sich im Wesentlichen um die in [JüKW00a] und [JüKW00b] unterbreiteten Vorschläge

<sup>4</sup> siehe dazu auch Abschnitt 4.2.3.5

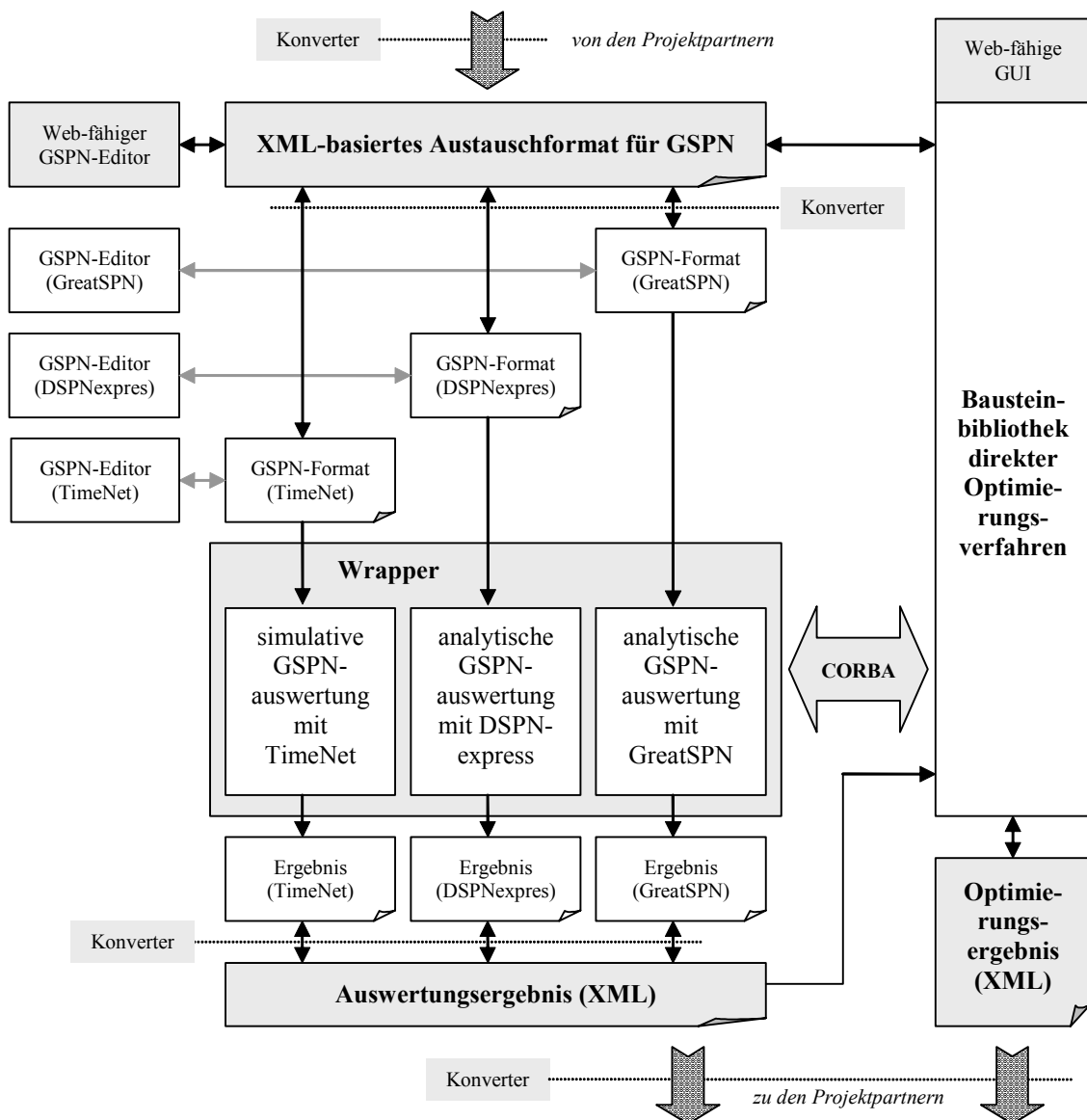


Abb. 6.1.2: Komponentenorientierter Aufbau des Web-basierten Werkzeugs zur dynamischen Prozessoptimierung

Um die für die dynamische Prozessoptimierung erforderliche Optimierungsfunktionalität bereitzustellen, wurde die in Abschnitt 5.2 beschriebene Optimierungskomponente eingebunden. Der interne Aufbau der Optimierungskomponente sowie die darin eingesetzten direkten Optimierungsverfahren wurden bereits ausführlich in Kapitel 5 beschrieben. Die Einbindung der Optimierungskomponente erforderte neben losen Integrationskonzepten auf Basis von Dokumentenaustausch auch den direkten Zugriff auf die Modellauswertungsmodulare der integrierten GSPN-Werkzeuge. Zur Realisierung dieser Zugriffe wurde mit CORBA ein Middlewarestandard eingesetzt, der eine flexible und dynamische Verteilung der beteiligten Komponenten auf heterogene physikalisch möglicherweise weit voneinander entfernt lokalisierte Plattformen erlaubt. Um den CORBA-Zugriff der Optimierungskomponente auf die Modellauswertungsmodulare der integrierten GSPN-Werkzeuge zu ermöglichen, wurden die dazu notwendigen Aufrufchnittstellen durch entsprechende Wrapper nachgebildet.

Abschließend sei angemerkt, dass ohne die konsequente Ausnutzung der vielfältigen Integrationsmöglichkeiten moderner Komponenten-Technologien die gesteckten Projektziele verfehlt worden wären, da eine vollständige Neuimplementierung der benötigten GSPN-Auswertungsmethoden und Optimierungsalgorithmen die zur Verfügung stehende Zeit bei weitem überschritten hätte. Durch die in Abb. 6.1.2 dargestellte Integration dieser Methoden konnte stattdessen ein sehr mächtiges Web-fähiges Werkzeug realisiert werden, das optimal auf die spezifischen Probleme der dynamischen Prozessoptimierung zugeschnitten ist. Weitere Einzelheiten zu diesem Forschungsprojekt finden sich in [SySy00], [SySy01], [SySy02] und [Info02].

## 6.2 Online-Animation direkter Optimierungsverfahren

Die Aufgabenstellung des in diesem Abschnitt beschriebenen Forschungsprojekts bestand in der Konzeption und Realisierung einer Animationsumgebung zur Visualisierung der Arbeitsweise direkter Optimierungsverfahren. Eine solche Animationsumgebung lässt sich in vielen Bereichen gewinnbringend einsetzen. Die wichtigsten Einsatzfelder sowie die dort vorhandenen Einsatzmöglichkeiten sind in Tabelle 6.2.1 zusammengefasst.

Bereich	Einsatzmöglichkeiten
Lehre/Lernen	<ol style="list-style-type: none"> <li>1. Multimediale Bereicherung von Lehrveranstaltungen und/oder virtueller Lernumgebungen</li> <li>2. Interaktives Lernmaterial für das Selbststudium</li> </ol>
Forschung	<ol style="list-style-type: none"> <li>1. Unterstützung der Grundlagenforschung zu direkten Optimierungsverfahren</li> <li>2. Illustrative Demonstration von Forschungsergebnissen</li> </ol>
Praktische Anwendung	Unterstützung des Anwenders bei der <ol style="list-style-type: none"> <li>1. Anpassung direkter Optimierungsverfahren an das vorliegende Optimierungsproblem</li> <li>2. Abschätzung der zur Durchführung der Optimierung benötigten Ressourcen</li> </ol>

*Tabelle 6.2.1: Einsatzmöglichkeiten einer Animationsumgebung zur Visualisierung der Arbeitsweise direkter Optimierungsverfahren*

Um den weltweiten Zugriff über Web-Browser zu ermöglichen, wurde die Animationsumgebung als Java-Applet realisiert. In der zurzeit vorliegenden Ausprägung ist die Animationsumgebung speziell auf die Bedürfnisse von Studenten bzw. auf dem Gebiet der direkten Optimierung unerfahrene Anwender zugeschnitten und wird dieser Zielgruppe im World Wide Web unter [http://goethe.ira.uka.de/~syryjakow/anim\\_env3/start\\_environment.html](http://goethe.ira.uka.de/~syryjakow/anim_env3/start_environment.html) sowie in der in [Zimm99] beschriebenen Lernumgebung zur Verfügung gestellt. Animiert wird die Arbeitsweise weit verbreiteter atomarer globaler und lokaler Optimierungsverfahren, wobei es sich im

Einzelnen um Genetische Algorithmen, das Simulated Annealing und die Mustersuche handelt. Diese Verfahren bilden die Grundbausteine der in Abschnitt 5.2 beschriebenen Optimierungskomponente und konnten aus dieser unverändert übernommen werden. Die Animationsumgebung kann daher wie in Abb. 5.2.1 dargestellt als ein spezieller Graphikaufsatz auf die von der Optimierungskomponente angebotene Bausteinbibliothek direkter Optimierungsverfahren aufgefasst werden.

In Abb. 6.2.1 sind die wesentlichen Komponenten der Animationsumgebung und deren Beziehungen untereinander dargestellt. Die graphische Benutzerschnittstelle erlaubt dem Anwender Optimierungsexperimente mit den angebotenen direkten Optimierungsverfahren durchzuführen. Vor dem Start eines Optimierungsexperiments muss eines der drei implementierten Optimierungsverfahren ausgewählt werden. Darüber hinaus ist ein Testproblem festzulegen, auf welches das Optimierungsverfahren angewendet werden soll. Diese Auswahlmöglichkeiten werden vom "Welcome Part" der Animationsumgebung angeboten, der in Abb. 6.2.2 dargestellt ist. In dem vorliegenden Bildschirmausschnitt hat sich der Anwender bereits für einen Genetischen Algorithmus entschieden, der auf die 2-dimensionale Variante des in Tabelle 5.3.2.2 spezifizierten mathematischen Testproblems ( $L_2^n, F_2^n$ ) angewendet werden soll. Neben ( $L_2^2, F_2^2$ ) werden 12 weitere 2-dimensionale Testprobleme mit sehr unterschiedlichen Oberflächencharakteristiken angeboten.

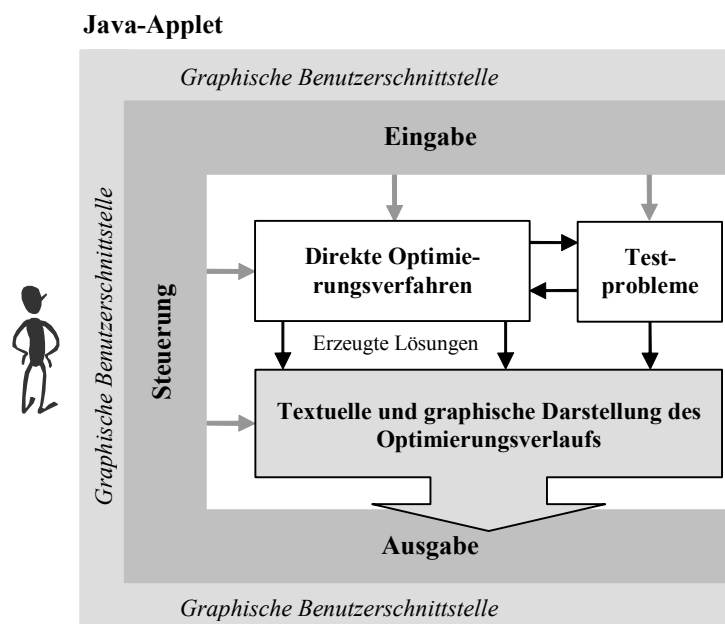


Abb. 6.2.1 Software-Architektur der Animationsumgebung für direkte Optimierungsverfahren

Nachdem ein Optimierungsverfahren und ein Testproblem ausgewählt wurden, kann der Anwender zu dem in Abb. 6.2.3 dargestellten "Animation Part" übergehen. Hier wird über die Schaltfläche "Adjustment" die Möglichkeit geboten, wichtige Kontrollparameter des ausgewählten Optimierungsverfahrens einzustellen. Nachdem der Optimierungsprozess durch Anklicken der Schaltfläche "Start" angestoßen wurde, werden die in den einzelnen Iterationsschritten erzeugten Suchpunkte sowohl textuell als auch graphisch angezeigt. Im Falle des Ge-

netischen Algorithmus werden nacheinander die generierten Suchpunktpopulationen dargestellt, wobei die Animationsgeschwindigkeit den Bedürfnissen des Anwenders entsprechend variiert werden kann.

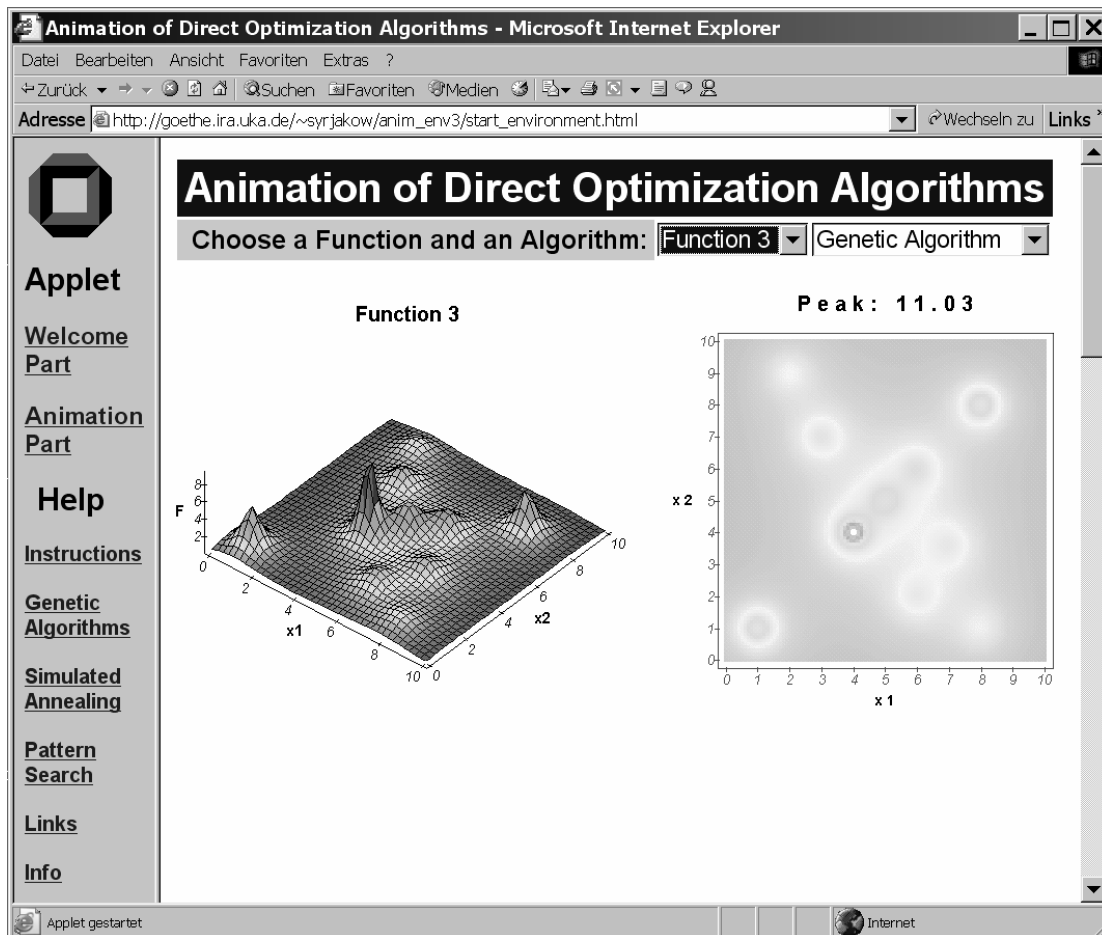


Abb. 6.2.2: "Welcome Part" der Animationsumgebung für direkte Optimierungsverfahren

In dem Textfenster auf der linken Seite werden die Suchpunkte einer Population zum einen in der vom Genetischen Algorithmus verwendeten Binärkodierung angezeigt, zum anderen in der für den menschlichen Betrachter geeigneteren Darstellung als reellwertige Koordinatentupel. Darüber hinaus wird der Zielfunktionswert angegeben sowie Detailinformationen über die Entstehungsgeschichte der Suchpunkte bereitgestellt. Unterhalb der aufgelisteten Lösungsalternativen befinden sich einige statistische Angaben über die Gesamtpopulation.

Rechts neben dem Textfenster ist die Ergebnisoberfläche der zu optimierenden Testfunktion als so genannter "density plot" aufgetragen. Auf diese 2-dimensionale Fläche werden die links im Textfenster dargestellten Suchpunkte eingetragen. Dadurch lassen sich sehr gut die Konvergenz bei der Suche nach einer optimalen Lösung sowie auffällige Suchpunktconstellations und -muster erkennen. In Abb. 6.2.3 sind die Suchpunkte aus der zehnten Generation des zur Optimierung verwendeten Genetischen Algorithmus auf die Ergebnisoberfläche von  $F_2^2$  aufgetragen. Die Häufung dieser Suchpunkte im Einzugsbereich des globalen Optimumpunkts ist

ein starkes Indiz dafür, dass die Konvergenz des Genetischen Algorithmus gegen diesen Punkt bereits weit fortgeschritten ist.

Um Zwischenergebnisse eingehend untersuchen zu können, kann ein gestartetes Optimierungsexperiment durch Anklicken der Schaltflächen "Stop" jederzeit angehalten werden. Durch einen Klick auf "Continue" lässt sich das angehaltene Optimierungsexperiment wieder weiterführen. Die Schaltfläche "Reset" ermöglicht das Abbrechen des Optimierungsexperiments. Über die "Comparison" Schaltfläche lässt sich schließlich auch die Gesamtrajektorie darstellen und mit den Trajektorien anderer Suchverfahren vergleichen.

Neben dem "Welcome Part" und "Animation Part" werden von der Animationsumgebung eine ausführliche Bedienungsanleitung, Detailinformationen über die implementierten direkten Suchverfahren sowie Verweise auf themenverwandte Seiten im World Wide Web bereitgestellt.

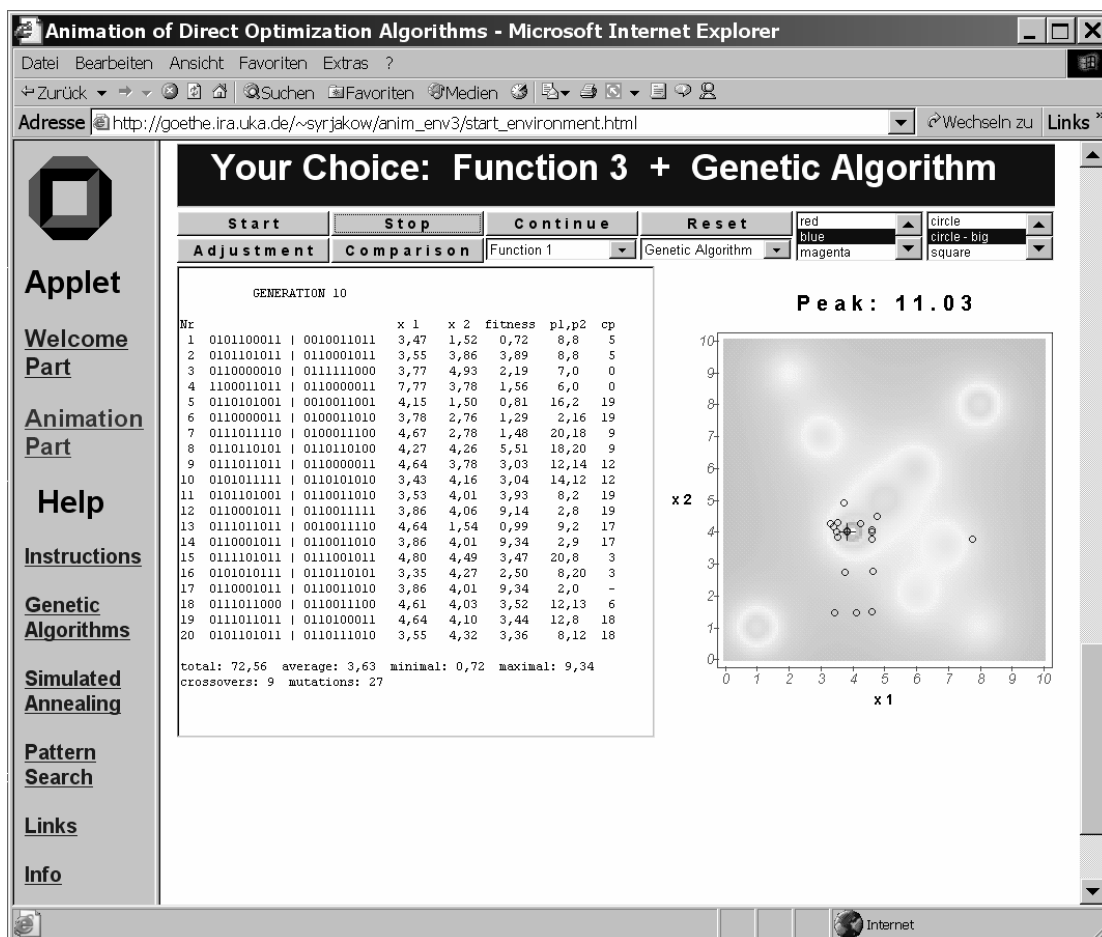


Abb. 6.2.3: "Animation Part" der Animationsumgebung für direkte Optimierungsverfahren

Dadurch, dass die von der Animationsumgebung angebotenen Optimierungsverfahren als vollständige Implementierungen vorliegen, deren Kontrollparameter beliebig eingestellt werden können, eröffnen sich vielfältige Experimentiermöglichkeiten. Um dem unerfahrenen Anwen-



der den Einstieg zu erleichtern, werden von der Animationsumgebung zu jedem der angebotenen Optimierungsverfahren Kontrollparametereinstellungen vorgegeben, die sich in der Praxis bewährt haben. Davon ausgehend lassen sich Optimierungsexperimente mit selbst gewählten Kontrollparametereinstellungen durchführen, wodurch tiefgehende Einblicke in die komplexe Arbeitsweise direkter Optimierungsverfahren gewonnen werden können.

Trotz der vollständigen Implementierung der Optimierungsverfahren hält sich der Ressourcenbedarf der Animationsumgebung in akzeptablen Grenzen, was eine wichtige Voraussetzung für die gewählte Realisierungsform als Java-Applet darstellt. Als solches kann die Animationsumgebung in kurzer Zeit aus dem Netz herunter geladen und auf heutigen PCs problemlos ausgeführt werden.

Wie bereits erwähnt ist die Animationsumgebung zurzeit vor allem auf Anwender zugeschnitten, die den Umgang mit direkten Optimierungsverfahren erlernen wollen. Eine Neuausrichtung auf andere Zielgruppen lässt sich beispielsweise durch das Auswechseln von Testfunktionen sowie das Hinzufügen weiterer direkter Optimierungsverfahren erreichen. Änderungen dieser Art können aufgrund der objektorientierten Realisierung sowie des flexiblen auf alle Arten<sup>5</sup> von direkten Optimierungsverfahren ausgelegten Präsentationsmoduls auf einfache Weise vorgenommen werden. Weitere Einzelheiten zur Animationsumgebung finden sich in [Bent98], [SySz99a] und [SyBS00].

## 6.3 Verteilte Simulation und Visualisierung künstlichen Lebens

Das in diesem Abschnitt beschriebene Forschungsprojekt befasste sich mit dem Themenbereich "Künstliches Leben" (KL), der in den letzten Jahren auf breites Interesse gestoßen ist. Die KL-Forschung ergänzt die traditionellen biologischen Wissenschaften, die sich primär mit der Analyse kohlenstoffbasierter Organismen beschäftigen, durch den Versuch, lebensähnliches Verhalten im Computer oder anderen künstlichen Medien zu synthetisieren. Hinsichtlich der Fragestellungen und zum Teil auch der verwendeten Methoden gibt es vielfältige Überschneidungen mit Arbeiten aus dem Bereich der Robotik und künstlichen Intelligenz. In der KL-Forschung geht es allerdings weder ausschließlich um die Übertragung von Prinzipien aus biologischen Systemen in technische Domänen (Bionik), noch um die Konstruktion möglichst leistungsfähiger menschenähnlicher Roboter, die in einer künstlichen oder realen Umwelt so agieren, dass sie definierte Aufgaben möglichst erfolgreich lösen. Im Fokus steht vielmehr die Konstruktion selbständig überlebensfähiger Artefakte, parallel zur Evolution natürlicher Systeme [Daut95].

Als hochgradig interdisziplinäre Forschungsrichtung, in der sich primär die Forschungsbereiche der Biologie, der Chemie, der Physik und der Informatik vermischen, ist KL enorm facettenreich<sup>6</sup>. Zur grundlegenden KL-Methodik gehörten bislang Lernalgorithmen, zelluläre Automaten sowie evolutionäre Algorithmen. Begünstigt durch die in den letzten Jahren enorm angestiegene Rechenleistung werden heute zunehmend auch Simulationen komplexer künstlicher

---

<sup>5</sup> sowohl Punkt-zu-Punkt vorgehende als auch populationsbasierte Methoden

<sup>6</sup> einen umfassenden Überblick über das Gesamtgebiet geben [Lang96] und [Adam98]

Welten entwickelt, in denen adaptive autonome Agenten verschiedene Aktionen ausführen, um zu überleben und ihr genetisches Material an Nachfolgenerationen weiterzugeben. Im Gegensatz zur äußerst langsamen Evolution organischer Lebensformen ereignet sich die synthetische Evolution in digitalen Systemen so schnell, dass es "über Nacht" zu wesentlichen Veränderungen in der Struktur der künstlichen Organismen kommen kann. Darüber hinaus ist häufig auch so genannte Emergenz von komplexem (globalem) Verhalten zu beobachten, das zutage tritt, ohne dass es explizit programmiert worden wäre. Ein weiterer großer Vorteil digitaler KL-Systeme gegenüber organischen Systemen besteht darin, dass sie sich weitaus besser und ohne jedes Gefahrenpotential beobachten und manipulieren lassen.

Neben künstlicher Evolution, Selbstorganisation und Selbstoptimierung beschäftigt sich KL in zunehmendem Maße auch mit der Interaktion zwischen Gruppen von autonomen Agenten. Untersucht werden in diesem Zusammenhang vor allem Aspekte der Agenten-Kommunikation und -Kooperation. Oft dienen dabei "soziale Insekten" wie beispielsweise Ameisen, Bienen oder Termiten als Vorbild für kollektives Verhalten.

Die Modellierung von Agenten mit ausgeprägtem sozialen Verhalten stellte eine der zentralen Anforderungen an die im Rahmen dieses Forschungsprojekts entwickelte KL-Simulation dar. Zu den weiteren Anforderungen an diese Simulation zählten:

- Bereitstellung einer komfortablen auch von Nicht-Experten leicht zu bedienenden graphischen Benutzeroberfläche, die es erlaubt, die simulierten Abläufe interaktiv zu steuern und anschaulich zu visualisieren.
- Erstellung umfangreicher Statistiken über die simulierten Abläufe, die dem Anwender eine eingehende Analyse der durchgeführten Experimente gestatten.
- Entwicklung eines geeigneten Verteilungskonzeptes unter Verwendung etablierter Middleware-Standards, um auch die Simulation sehr komplexer KL-Szenarien zu ermöglichen.
- Verwendung der Programmiersprache Java, um die KL-Simulation Plattform-unabhängig zu realisieren und auf einfache Weise über das Web zugreifbar zu machen.

Voraussetzung für das komfortable Experimentieren mit einer KL-Simulation ist eine geeignete graphische Benutzeroberfläche, die dem Anwender eine übersichtliche und umfassende Beobachtung der dynamischen Vorgänge in der künstlichen Welt erlaubt. Darüber hinaus müssen vielfältige Möglichkeiten zum interaktiven Eingreifen in die simulierten Abläufe bestehen. Um diesen Anforderungen gerecht zu werden, wurde zur Realisierung der graphischen Benutzeroberfläche auf das am Institut für Rechnerentwurf und Fehlertoleranz entwickelte Java Media Tool<sup>7</sup> (JMT) zurückgegriffen. Das JMT stellt eine flexible Ablaufumgebung für verteilte Multimedia-Anwendungen in Java bereit und ermöglicht eine nahezu vollständige Entkopplung der Simulation von dem zur Darstellung der simulierten Prozesse verwendeten Darstellungsmodell.

---

<sup>7</sup> Einzelheiten zum JMT finden sich in [BeSy00a] und [Berd02]

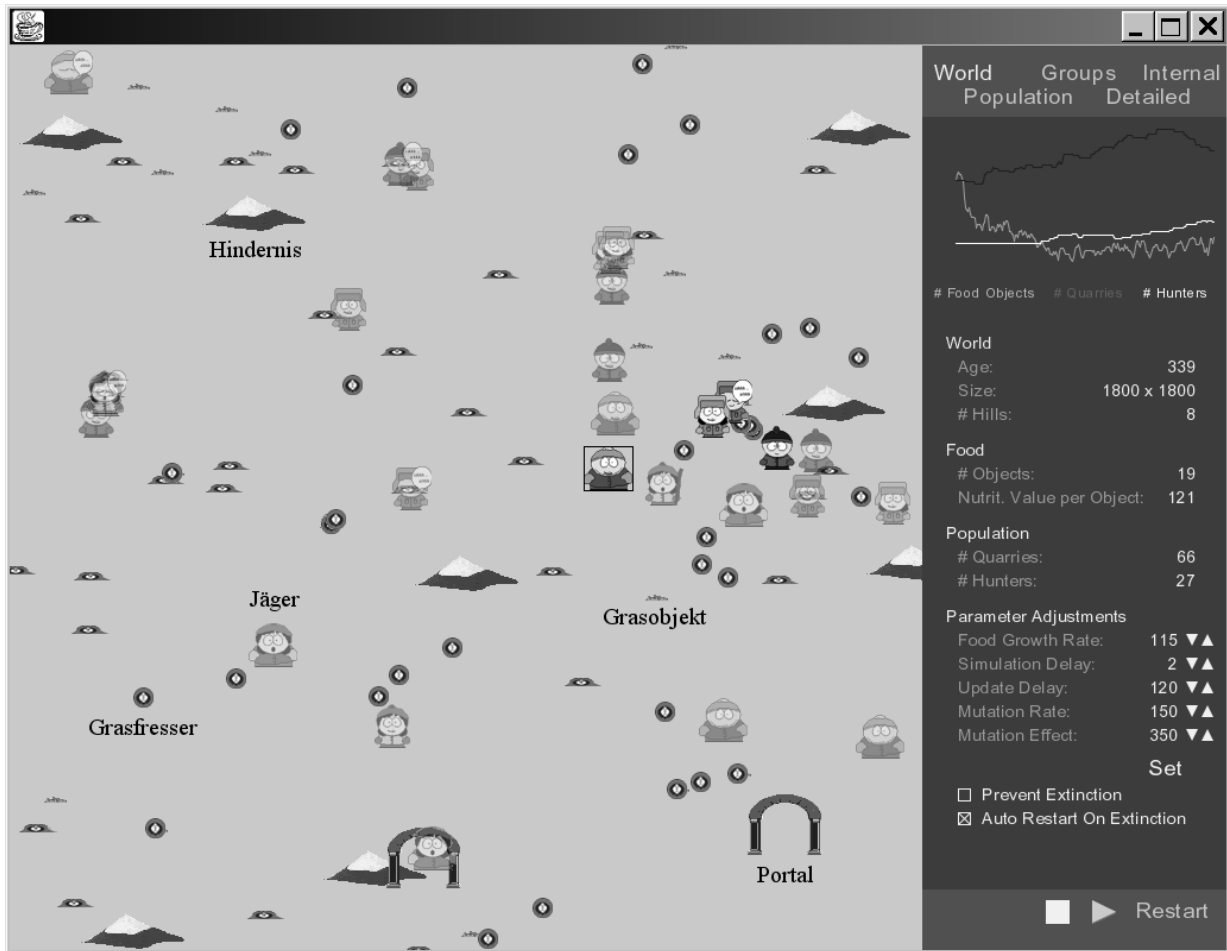


Abb. 6.3.1: Hauptfenster der graphischen Benutzeroberfläche der KL-Simulation

Abb. 6.3.1 zeigt das Hauptfenster der graphischen Benutzeroberfläche der KL-Simulation. Das Rechteck auf der linken Seite stellt das künstliche Ökosystem dar, in dem fünf verschiedene Arten von Objekten vorkommen können:

- Grasobjekte, die mit einer vom Anwender einstellbaren Wachstumsrate erzeugt werden
- eine konstante Anzahl von Hindernissen (Bergen), die während der Initialisierung an zufällig ausgewählten Stellen im Ökosystem platziert werden
- zwei Arten von anpassungsfähigen autonomen Agenten, die im Gegensatz zu den beiden oben genannten statischen Objekten in der Lage sind, innerhalb des Ökosystems ihre Position zu verändern
- Portale, die es den Agenten ermöglichen, in andere Ökosysteme zu gelangen

Das künstliche Ökosystem kann als eine Projektion eines Torus auf ein Rechteck aufgefasst werden, was für die sich darin bewegendenden Objekte bedeutet, dass, wenn sie die rechteckige Welt an einer der vier Seiten verlassen, sie an der gegenüberliegenden Seite wieder in die Welt eintreten.

Bei den zwei Arten autonomer Agenten, die das Ökosystem bevölkern, handelt es sich zum einen um primitive Grasfresser, die sich von den im Ökosystem wachsenden Grasobjekten ernähren, zum anderen um höher entwickelte Jäger, die wiederum von der Jagd auf Grasfresser leben. Damit wird ein Ökosystem mit einer sehr einfachen Nahrungskette etabliert, für das jedoch, wie die damit durchgeführten Experimente gezeigt haben, nur sehr schwer ein stabiler Gleichgewichtszustand gefunden werden kann.

Das Globalziel der in diesem künstlichen Ökosystem lebenden Individuen besteht darin, möglichst alt zu werden und sich während ihrer Lebenszeit möglichst oft fortzupflanzen. Um dies zu erreichen, befriedigen die Agenten zunächst einmal ihr Bedürfnis nach Nahrung und danach ihre Reproduktionsbedürfnisse. Im Einzelnen können die Agenten folgende Basisaktionen ausführen:

- Umschauen, um den Kenntnisstand über ihre unmittelbare Umgebung zu aktualisieren
- Bewegen, um ihre Position im Ökosystem zu verändern
- Essen, um ihren Energievorrat zu erhöhen
- Schlafen, um Energie zu sparen und ihre Vitalität<sup>8</sup> zu erhöhen
- Reproduzieren, um ihre genetischen Informationen an nachfolgende Generationen weiterzugeben

Im Gegensatz zu den primitiven eingeschlechtlichen Grasfressern, verfügen die höher entwickelten Jäger, die in männlicher und weiblicher Form vorkommen, über ein ausgeprägtes soziales Verhalten. Dieses versetzt sie in die Lage, Gruppen zu bilden, um ihre Chancen bei der Jagd auf Grasfresser zu erhöhen. Jede Gruppe wird von einem Gruppenleiter angeführt, der die Aktionen der Gruppe bei der Jagd koordiniert und Gruppenmitglieder rekrutiert bzw. wieder entlässt. Zur Realisierung des dazu erforderlichen Kommunikationsmechanismus wurde auf die im Rahmen der "ARPA<sup>9</sup> Knowledge Sharing" Initiative speziell zum Zweck der Nachrichtenübermittlung zwischen Agenten entwickelte Knowledge Query and Manipulation Language<sup>10</sup> (KQML) zurückgegriffen. KQML wird von den Jägern auch dazu verwendet, um andersgeschlechtliche Partner zur Reproduktion zu finden. Das aus der Reproduktion zwischen zwei Jäger-Agenten hervorgehende Kind weist eine zufällige Mischung des durch einen probabilistischen Mutationsoperator an einigen Stellen geringfügig veränderten elterlichen Erbmaterials auf.

Um die dynamischen Gruppenbildungsprozesse, die unter den Jägern ablaufen, möglichst gut verfolgen zu können, bietet die graphische Benutzeroberfläche dem Anwender folgende Beobachtungsmöglichkeiten:

1. Wird ein Jäger-Agent mit der Maus selektiert, werden alle Jäger-Agenten, die nicht zu seiner Gruppe gehören, transparent dargestellt. Diese Gruppenbeobachtungsmöglichkeit wurde in dem in Abb. 6.3.1 dargestellten Ökosystem verwendet.

---

<sup>8</sup> nur Jäger verfügen über dieses Attribut

<sup>9</sup> Advanced Research Projects Agency

<sup>10</sup> Detailinformationen über KQML werden in [FiLM97] sowie im World Wide Web unter [KQML] zur Verfügung gestellt

2. Um die räumliche Ausdehnung einer Gruppe besser überblicken zu können, kann der Anwender einen zusätzlichen Beobachtungsmechanismus aktivieren, bei dem alle Mitglieder einer Gruppe durch ein transparentes Polygon überdeckt werden. In dem in Abb. 6.3.2 dargestellten Ökosystem wird gerade eine Jäger-Gruppe von einem solchen Polygon überdeckt.
3. Schließlich ist es auch möglich, alle gerade existierenden Gruppen durch transparente Polygone zu überdecken. Diese Möglichkeit wurde in dem in Abb. 6.3.3 dargestellten Ökosystem aktiviert.



Abb. 6.3.2: Visualisierung der räumlichen Ausdehnung einer Jäger-Gruppe durch ein transparentes Polygon

Jäger-Agenten unterscheiden sich nicht nur hinsichtlich ihres Geschlechts, sondern auch hinsichtlich ihrer Spezialfähigkeiten. Neben den bereits bekannten Gruppenleitern gibt es so genannte Lähmer und Angreifer. Die Lähmer haben die spezielle Fähigkeit, Grasfresser aus einiger Entfernung zu lähmen, so dass diese sich für eine bestimmte Zeit nur mit verminderter Geschwindigkeit fortbewegen können. Solche langsamen Grasfresser stellen wiederum eine leichte Beute für die Angreifer dar, welche die Fähigkeit haben, Grasfresser zu erlegen. Nach erfolgreicher Jagd hat der Gruppenleiter die Aufgabe, die Beute unter den Gruppenmitgliedern gerecht zu verteilen.

Um Geschlecht und Typ der Jäger-Agenten unterscheiden zu können, wurden zu ihrer Darstellung verschiedene Figuren verwendet. Dicke Figuren zeigen an, dass es sich um Gruppenleiter handelt, die etwas dünneren Figuren kennzeichnen Lähmer und Angreifer. Bestimmte Zustände der Agenten wie z.B. Essen, Schlafen, Jagen, Kommunizieren und Reproduzieren werden durch Figuren mit der jeweils dazu passenden Gestik dargestellt. Eine Figur mit Stock zeigt beispielsweise an, dass der betreffende Agent gerade auf der Jagd nach Grasfressern ist.



Abb. 6.3.3: Visualisierung der räumlichen Ausdehnung aller vorhandenen Jäger-Gruppen durch transparente Polygone

Wie bereits erwähnt wurde zur Plattform-unabhängigen Realisierung der KL-Simulation die Programmiersprache Java verwendet. Die hier bestehenden hohen Ressourcenanforderungen sowie der Geschwindigkeitsnachteil von Java gegenüber kompilierten Programmiersprachen führen dazu, dass bereits bei mittelgroßen Ökosystemen mit einigen Hundert Agenten die Leistungsgrenze heutiger Standard-PCs (1 GHz, 512 MByte RAM) erreicht wird. Um auch Ökosysteme von sehr großer räumlicher Ausdehnung mit einigen Tausend Agenten simulieren zu können, wurde ein flexibles und einfach zu handhabendes Konzept zur Verteilung der KL-Simulation entwickelt. Dieses Konzept basiert im Wesentlichen auf Portalen, mit denen sich ver-

schiedene Ökosysteme, die auf einem oder mehreren vernetzten Rechnern laufen können, zu einem komplexen Gesamtsystem verknüpfen lassen.

Abb. 6.3.4 zeigt ein solches verteiltes Ökosystem, das aus mehreren über Portale verbundenen Einzelsystemen besteht, die aufgrund ihrer Implementierung in Java auf den unterschiedlichsten Plattformen ablaufen können. Wenn ein Ökosystem für einen Agenten unattraktiv geworden ist, weil er dort beispielsweise nicht mehr genug Futter findet oder es ihm nicht gelingt, sozialen Kontakt zu anderen Agenten herzustellen, kann dieser ein solches Portal benutzen, um zu einem anderen für ihn eventuell besser geeigneten Ökosystem zu gelangen. Um den Kommunikationsaufwand zwischen untereinander verbundenen Ökosystemen gering zu halten, können Agenten keinen Kontakt zu Agenten in anderen Ökosystemen aufnehmen. Darüber hinaus werden Agenten ausschließlich von dem Ökosystem verwaltet, in dem sie sich gerade aufhalten. Wenn ein Agent zu einem anderen Ökosystem wechselt, werden seine Informationen komplett aus dem bisherigen Ökosystem gelöscht.

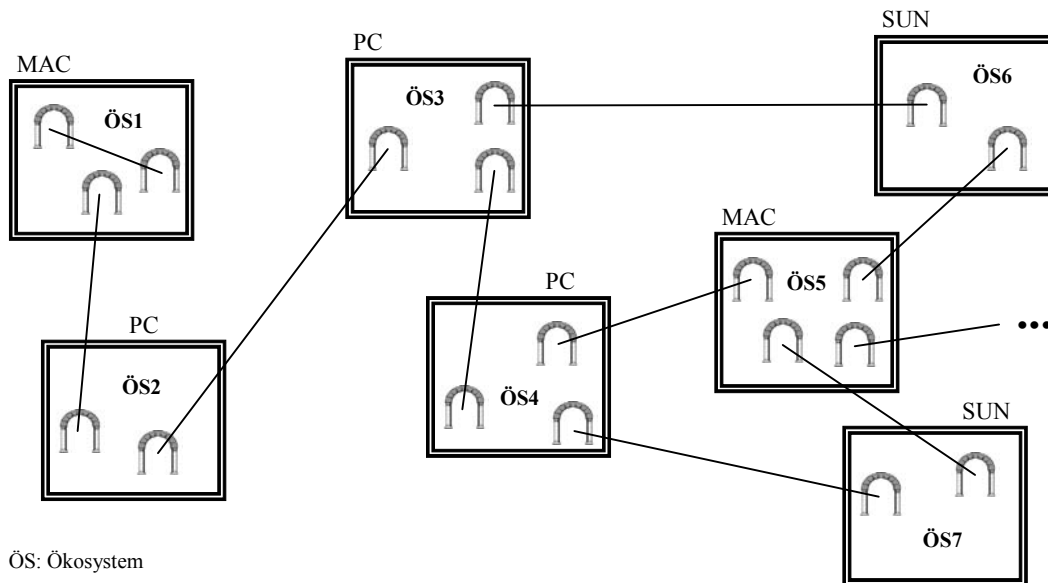


Abb. 6.3.4: Beispiel eines verteilten Ökosystems, das aus mehreren über Portale verbundenen Einzelsystemen besteht

Die simulierten Abläufe in einem Ökosystem beeinflussen also in keiner Weise die Abläufe in den anderen mit diesem System vernetzten Ökosystemen. Auf eine zeitliche Synchronisation der an einer verteilten KL-Simulation beteiligten Einzelsysteme konnte daher verzichtet werden. Dies war schließlich auch einer der Hauptgründe dafür, warum zur technischen Umsetzung des Verteilungskonzepts nicht auf die High Level Architecture, sondern auf Voyager<sup>11</sup> zurückgegriffen wurde, das sich aufgrund seiner Agentenunterstützung zu diesem Zweck als hervorragend geeignet herausstellte. Weitere Details zur Verteilung der KL-Simulation und deren technische Realisierung finden sich in [Pütt00].

<sup>11</sup> siehe dazu auch Abschnitt 2.5.2.2

Die in den Abbildungen 6.3.1, 6.3.2 und 6.3.3 in Ausschnitten dargestellte graphische Benutzeroberfläche ermöglicht dem Anwender ein komfortables Experimentieren mit der KL-Simulation. Das Rechteck auf der linken Seite erlaubt die Beobachtung des gesamten Ökosystems (Macro View) und der sich darin entwickelnden Agentenpopulationen. Zu diesem Zweck werden sämtliche der simulierten statischen (Hindernisse, Grasobjekte, Portale) und dynamischen Objekte (Grasfresser, Jäger) angezeigt. Die verschiedenen Registerreiter auf der rechten Seite stellen dem Anwender statistische Informationen über die sich entwickelnden Populationen sowie über einzelne von ihm selektierte Individuen zur Verfügung (Micro View). Über den Registerreiter "Detailed" erhält man beispielsweise Informationen über sämtliche Attribute (Geschlecht, Fähigkeiten, aktuelle Bedürfnisse, Alter, Sichtfeldgröße, etc.) des gerade selektierten Agenten<sup>12</sup>. Darüber hinaus kann der Anwender über den in Abb. 6.3.1 dargestellten Registerreiter "World" verschiedene Kontrollparameter der Simulation verändern wie beispielsweise die Wachstumsrate für die Grasobjekte oder die den Reproduktionsmechanismus beeinflussende Mutationsrate.

Das Experimentieren mit der KL-Simulation hat sich als sehr interessant und herausfordernd erwiesen. Bei der Art der durchgeführten Experimente lässt sich grundsätzlich unterscheiden zwischen Experimenten

- in deren Verlauf die zu Beginn festgelegten Kontrollparametereinstellungen unverändert beibehalten werden. Eine mögliche Zielsetzung solcher Experimente besteht darin, Parametereinstellungen zu ermitteln, bei denen sich über einen längeren Zeitraum ein stabiles Ökosystem einstellt (keine Spezies stirbt vorzeitig aus).
- in deren Verlauf Änderungen an den Kontrollparametereinstellungen vorgenommen werden, beispielsweise dann, wenn eine der vorhandenen Spezies auszusterben droht, mit der Absicht, das Ökosystem durch gezieltes Eingreifen über einen möglichst langen Zeitraum stabil zu halten.

Interessant zu beobachten ist vor allem, wie sich die Fähigkeiten der Agenten im Laufe der Generationen evolutionär weiterentwickeln. Eine große Herausforderung stellt wie bereits angedeutet die Ermittlung von Kontrollparametereinstellungen dar, die über einen längeren Zeitraum ein stabiles Ökosystem garantieren. Einige interessante Ergebnisse aus Experimenten mit der KL-Simulation werden in [Pütt00] und [SBPS01] vorgestellt und diskutiert.

Die KL-Simulation lässt sich ebenso wie die in Abschnitt 6.2 beschriebene Animationsumgebung zur Visualisierung der Arbeitsweise direkter Optimierungsverfahren sehr flexibel in Forschung und Lehre einsetzen. Für den Einsatz zu Forschungszwecken haben sich vor allem folgende Eigenschaften als sehr vorteilhaft herausgestellt:

- Hohe Skalierbarkeit bezüglich der Ausdehnung der simulierten künstlichen Welt durch das einfache portalbasierte Verteilungskonzept sowie dessen technischer Umsetzung mit der Agentenplattform Voyager.

---

<sup>12</sup> In Abb. 6.3.2 werden auf diese Weise die Attribute eines weiblichen Gruppenleiters angezeigt, in Abb. 6.3.3 können die Attribute eines männlichen Gruppenleiters eingesehen werden.



- Leichte Erweiterbarkeit und Modifikation durch die objektorientierte Realisierung in Java sowie die konsequente Trennung von Simulation und Darstellung durch Verwendung des Java Media Tool.

Aufgrund der einfach zu handhabenden graphischen Benutzeroberfläche sowie der Plattform-unabhängigen Verfügbarkeit ist die KL-Simulation auch hervorragend für den Einsatz in der Lehre geeignet. In ihrer jetzigen Form liegt die KL-Simulation als Java-Applikation vor. Es ist jedoch geplant, auch eine Applet-Version bereitzustellen, auf die flexibel über das World Wide Web zugegriffen werden kann.



## *Kapitel 7*

# Schlussfolgerungen

---

In der vorliegenden Arbeit wurde aufgezeigt, dass sich Web- und Komponenten-Technologien im Bereich der computergestützten Modellierung und Simulation auf vielfältige Weise gewinnbringend einsetzen lassen. In diesem Kapitel wird abschließend auf die durch den Einsatz dieser Schlüsseltechnologien hervorgerufenen Auswirkungen und Veränderungen eingegangen. In Anlehnung an [Fish96], [PBFH98], [Page98], [PaGR98], [Page99], [Nara00], [KuPa00], [PaOp00] und [PBFH00], die sich intensiv mit dieser Thematik auseinandergesetzt haben, wird versucht, sowohl die bereits sichtbaren als auch die aus heutiger Sicht schwer abschätzbaren potentiellen Auswirkungen innerhalb und außerhalb der Modellierung und Simulation zusammenfassend darzustellen.

## **7.1 Auswirkungen innerhalb der Modellierung und Simulation**

Durch den Einsatz von Web- und Komponenten-Technologien ergeben sich für den Bereich der Modellierung und Simulation zahlreiche Chancen aber auch gewisse Risiken. Diese werden im Folgenden aufgezählt und näher erläutert.

### **7.1.1 Chancen**

Innerhalb der Modellierung und Simulation vereinfachen und unterstützen Web- und Komponenten-Technologien vor allem folgende Vorhaben:

- Kollaborative Entwicklung und Ausführung von Simulationsanwendungen

An der Planung, Konzeption, Realisierung, Überprüfung und Ausführung komplexer Simulationsmodelle sind in der Regel mehrere Personen mit sehr unterschiedlichem Kenntnisstand beteiligt. Dazu zählen Simulationsexperten, Fachleute für das zu modellierende System, technisches Personal und letztendlich auch die Entscheidungsträger, die das Simulationsmodell in Auftrag gegeben haben und an den daraus zu gewinnenden Informationen interessiert sind. Web-Technologien ermöglichen, dass diese meist an verschiedenen Orten lokalisierten Personen in unterschiedlichen Formen zusammenarbeiten können. Eine solche Zusammenarbeit kann sich über die gemeinsame Verwaltung von Dokumenten bis hin zum kollaborativen Erstellen, Ausführen und Analysieren von Simulationsmodellen erstrecken.

Voraussetzung dazu sind entsprechende CSCW (Computer Supported Cooperative Work) Systeme bzw. mit dieser Funktionalität ausgestattete Modellierungswerkzeuge.

- Modellaufbau aus vorgefertigten Modellbausteinen

Der Aufbau komplexer Simulationsmodelle ist in der Regel mit hohen Kosten verbunden. Komponenten-Technologien eröffnen die Chance, Modelle aus vorgefertigten Bausteinen zusammensetzen und dadurch Zeit und Entwicklungskosten einzusparen. Diese Vorgehensweise setzt jedoch funktionierende Komponentenmärkte voraus, auf denen sich preiswert und schnell fertige Subsysteme einkaufen lassen. Bisher existieren solche Märkte jedoch nur für einige wenige Domänen wie beispielsweise grafische Oberflächenelemente für GUIs. Weitere Domänen könnten sich durch das zunehmend an Popularität gewinnende Virtual Prototyping<sup>1</sup> ergeben, bei dem eine weitgehend im Rechner integrierte Produktentwicklung angestrebt wird. Grundvoraussetzung zum schnellen Aufbau virtueller Prototypen sind domänenspezifische Bausteinbibliotheken, welche die benötigten Teilmodelle zur Verfügung stellen. Solche Bibliotheken existieren heute bereits ansatzweise in den Bereichen Automobil- und Maschinenbau.

- Verteilte Ausführung von Simulationsanwendungen

Die Ausführung komplexer Simulationsmodelle ist i.a. mit einem hohen Ressourcenbedarf verbunden. Die High Level Architecture stellt mit der Laufzeit-Infrastruktur RTI eine mächtige Kommunikationsinfrastruktur zur Verfügung, die eine flexible Verteilung von Simulationsanwendungen auf verschiedene Rechner erlaubt. Die damit verbundenen Vorteile wurden bereits ausführlich in Abschnitt 4.1 beschrieben.

- Schaffung und Etablierung von Standards

Der Einsatz von Web- und Komponenten-Technologien in einer Domäne stößt dort in der Regel zahlreiche Standardisierungsprozesse an. Dies trifft auch auf den Bereich der computergestützten Modellierung und Simulation zu. Dort wurde mit der High Level Architecture ein speziell auf die Problematik verteilter Simulationen zugeschnittener Komponentenstandard geschaffen. Darüber hinaus werden zurzeit XML-basierte Modellaustauschformate zu gebräuchlichen Modellierungstechniken wie beispielsweise erweiterte Petrinetze und Warteschlangennetze entwickelt und standardisiert.

- Kombination von Simulationen mit bereichsfremden Anwendungen

Standardisierte Schnittstellen ermöglichen nicht nur die Kopplung von Simulationen untereinander, sondern auch deren flexible Kombination mit bereichsfremden Anwendungen. Als sehr fruchtbar haben sich in der Vergangenheit vor allem Kombinationen mit Methoden aus der Künstlichen Intelligenz [SzUt99] erwiesen. Mit der in dieser Arbeit vorgestellten Optimierungskomponente wurde eine seit langem geforderte Verbindung zwischen den beiden Bereichen Modellierung und Optimierung hergestellt, die bislang weitgehend getrennt voneinander existiert haben.

---

<sup>1</sup> Virtuelle Prototypen werden definiert als eine computerbasierte Simulation eines technischen Systems oder Subsystems mit einem Grad an funktionalem Verhalten, der mit entsprechenden physikalischen Prototypen vergleichbar ist.

- Zeit-, orts- und plattformunabhängiger Zugriff auf Simulationsanwendungen

Web-Technologien ermöglichen die Realisierung von Online-Simulationen und -Animationen, auf die zeit-, orts- und plattformunabhängig über das Web zugegriffen werden kann. Damit entfällt der bislang nicht unerhebliche Installationsaufwand und Simulationsanwendungen werden grundsätzlich für jedermann zugänglich. Neben den Primärtechnologien des World Wide Web spielen dabei auch Komponenten-Technologien eine tragende Rolle, da sie die Grundvoraussetzung darstellen, um Anwendungen flexibel über das Web bereitzustellen und miteinander zu integrieren.

- Neue Formen der Distribution von Simulationsanwendungen

Web-Technologien erlauben es Anbietern von Simulationsanwendungen, ihre Produkte direkt über das World Wide Web zu vertreiben. Die sich daraus ergebenden Vorteile für den Endkunden liegen zum einen darin, dass Simulationsanwendungen etwas kostengünstiger angeboten werden können, da der Weg über den Zwischenhandel entfällt. Zum anderen erlaubt das Web dem Kunden einen weltweiten Vergleich von Produkten, was wiederum den Wettbewerb unter den Herstellern fördert.

## 7.1.2 Risiken

Der Einsatz von Web- und Komponenten-Technologien in der Modellierung und Simulation ist auch mit gewissen Risiken verbunden. Diese Risiken ergeben sich vor allem aus

- der schnellen Weiterentwicklung dieser Schlüsseltechnologien

Web- und Komponenten-Technologien haben sich seit ihrem Aufkommen ständig verändert und werden das voraussichtlich auch weiterhin in rasanter Weise tun. So wird heute beispielsweise HTML zunehmend durch XML verdrängt, CGI wird durch die moderneren Java-Servlets ersetzt, Java selbst unterliegt noch häufig Änderungen und Erweiterungen, Web-Browser werden ständig durch neue Funktionalität ergänzt und die Web Services treten zunehmend in Konkurrenz zu den heute etablierten Komponentenstandards. Beim Einsatz von Web- und Komponenten-Technologien sollten die Effekte und Konsequenzen dieses rasanten Wandels auf jeden Fall berücksichtigt werden. Da die Simulation noch nicht richtig am Wettlauf ins Web teilgenommen hat, stehen die Chancen gut, Fehler, wie sie beispielsweise im Bereich des E-Commerce gemacht wurden, zu vermeiden.

- falschem bzw. missbräuchlichem Gebrauch von Simulationsanwendungen

Über das Web können Simulationen heute auf einfache Weise einer breiten Öffentlichkeit zugänglich gemacht werden. Dies ist einerseits sehr begrüßenswert, da nun grundsätzlich jeder, der über einen Internetanschluss verfügt, Simulationen auf vielfältige Weise<sup>2</sup> nutzen kann. Andererseits ergeben sich daraus aber auch gewisse Risiken, beispielsweise dann, wenn der Anwender nicht über das notwendige Expertenwissen zum Umgang mit einer Simulation verfügt. In diesem Fall ist die Gefahr sehr groß, dass Simulationsergebnisse falsch

---

<sup>2</sup> zur Unterhaltung, Weiterbildung, Entscheidungsfindung, etc.

interpretiert werden, was wiederum bei einer Übertragung der Ergebnisse auf die Wirklichkeit zu schwerwiegenden Konsequenzen führen kann. Weitere Gefahrenpotentiale bestehen insbesondere bei Simulationsanwendungen aus dem Unterhaltungsbereich. Hier ist das Risiko sehr groß, dass über Simulationen anstößige, gesetzeswidrige oder anderweitig bedenkliche Inhalte auf eine sehr plastische und damit nachhaltig beeinflussende Weise verbreitet werden.

- zu schnellem und sorglosem "Rapid Prototyping"

Wird eine komplexe Simulationsanwendung nach dem Prinzip des "Rapid Prototypings" aus vorgefertigten Einzelbausteinen zusammengesetzt, ergeben sich in der Regel deutlich kürzere Entwicklungszeiten und damit auch niedrigere Entwicklungskosten. Ein weiterer Vorteil besteht darin, dass sich in kurzer Zeit mehrere Lösungsvarianten durchprobieren lassen. Der zunehmende Druck mit innovativen Produkten früher als die Wettbewerber am Markt zu sein, kann jedoch auch dazu führen, dass Hersteller bei dieser Art der Produktentwicklung zu schnell und ohne die notwendige Sorgfalt vorgehen. Dann ist die Gefahr sehr groß, dass unausgereifte Produkte von minderer Qualität entstehen.

## 7.2 Auswirkungen auf andere Bereiche

Die Möglichkeit, Computersimulationen und -animationen flexibel über das World Wide Web zu verbreiten, hat nicht nur die computergestützte Modellierung und Simulation selbst beeinflusst, sondern auch all diejenigen Bereiche, die Web-basierte Simulationsanwendungen zur Verfügung stellen. Dabei handelt es sich heute vor allem um

- den Bereich der Forschung

Web-basierte Simulationsanwendungen erlauben es, Forschungsergebnisse anschaulich zu präsentieren und die zugrunde liegenden Experimente auch für Nichtexperten nachvollziehbar zu machen. Wissenschaftliche Publikationen sollten daher nach Möglichkeit durch multimediale Präsentationen und Simulationen ergänzt und grundsätzlich im Web verfügbar gemacht werden.

- den Bereich der Lehre

Simulationsanwendungen unterstützen als multimediale und interaktive Techniken die Entwicklung moderner didaktischer Konzepte und ermöglichen neue Formen des Lernens und Verstehens abstrakter Sachverhalte. Das Web ermöglicht deren flexible Nutzung sowohl im traditionellen Schulunterricht als auch in den verschiedenen Bereichen des Studiums, Selbststudiums, der Nachhilfe und der Erwachsenenbildung. Um auf einem globalisierten Bildungsmarkt konkurrenzfähig zu bleiben, sind Bildungsinstitutionen heute in zunehmendem Maße gezwungen, ihre Web-Präsenz durch die Bereitstellung eines breit gefächerten Angebots an multimedialen Lehr- und Lernmaterialien weiter auszubauen. Von großer Bedeutung sind in diesem Zusammenhang Web-basierte Simulationen und Animationen, die es

dem Lernenden erlauben, das Verhalten komplexer Systeme durch interaktives Experimentieren auf anschauliche Weise und in kurzer Zeit zu erschließen<sup>3</sup>.

- die Unterhaltungsbranche

Zu den heute wohl populärsten Simulationsanwendungen zählen Computerspiele, die zurzeit den am schnellsten wachsenden Sektor der Unterhaltungsbranche darstellen. Bei Computerspielen geht es in der Regel um die Simulation einer Phantasiewelt und der darin vorkommenden Charaktere, mit denen der Anwender interagiert. Auch im Spielebereich wird zunehmend Web-Fähigkeit vorausgesetzt, insbesondere dann, wenn die Möglichkeit geboten werden soll, dass mehrere Spieler gleichzeitig über das Internet oder ein lokales Rechnernetz an einem Spiel teilnehmen können (Mehrspieler-Modus). Computerspiele stellen heute nicht nur einen bedeutenden Wachstumsmarkt dar, sondern haben auch Schrittmacherfunktion in vielen angrenzenden Bereichen übernommen. Neben dem Simulationsbereich selbst beeinflussen Computerspiele aufgrund ihrer enormen Ressourcenansprüche vor allem die Hardwareentwicklung. Darüber hinaus wird der Bereich der Künstlichen Intelligenz angeregt, deren Methoden in Computerspielen in zunehmendem Maße zum Einsatz kommen.

Vieles deutet darauf hin, dass das Angebot an Simulationsanwendungen im Web auch weiterhin rasant ansteigt und damit Simulationen bald zum alltäglichen Gebrauchsgegenstand werden. Die meisten der im Web angebotenen Simulationen wurden bislang fast ausschließlich zu Lehr-/Lern- und Unterhaltungszwecken eingesetzt. Heute etablieren sich Simulationsanwendungen jedoch auch in zunehmendem Maße als Hilfsmittel zur Entscheidungsfindung. Beispiele dafür sind Simulationen und Animationen zur Wettervorhersage, Finanzplanung sowie zur virtuellen Haus- und Wohnungsplanung. Damit öffnen sich nun auch Bereiche einer breiten Anwenderschicht, in denen die Simulation bislang ausschließlich einem elitären Kreis von Ingenieuren, Wissenschaftlern und Entscheidungsträgern vorbehalten war.

---

<sup>3</sup> ein gutes Beispiel für eine solche Anwendung stellt die in Abschnitt 6.2 beschriebene Web-basierte Animationsumgebung für direkte Optimierungsverfahren dar





# Literaturverzeichnis

---

- AaKo89 Aarts, Emile; Korst, Jan: Simulated Annealing and Boltzmann Machines; John Wiley & Sons, 1989.
- Abra98 Abrams, Marc (ed.): World Wide Web: Beyond the Basics; Prentice Hall, 1998.
- Ackl87 Ackley, David H.: A Connectionist Machine for Genetic Hillclimbing; Kluwer Academic Publishers, 1987.
- Adam98 Adami, C.: Introduction to Artificial Life; Telos - A Springer-Verlag Imprint, 1998.
- AlEr99 Altherr, M.; Erzberger, M.: Message-oriented Middleware; in Java Spektrum, Ausgabe 6, Nov./Dez. 1999, S. 30-34.
- Bäck96 Bäck, Th.: Evolutionary Algorithms in Theory and Practice; Oxford University Press, New York, 1996.
- BäFM97 Bäck, Th.; Fogel, D.B.; Michalewicz, Z. (editors): Handbook of Evolutionary Computation; Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.
- Balz99 Balzert, H.: Lehrbuch der Objektmodellierung - Analyse und Entwurf; Spektrum Akademischer Verlag, 1999.
- BaKr96 Bause, Falko; Kritzinger, Pieter S.: Stochastic Petri Nets: An Introduction to the Theory; Vieweg, 1996.
- BaNC96 Banks, J.; Nelson, B.L.; Carson, J.S.: Discrete-Event System Simulation; Prentice Hall, 1996.
- BCSS99 Banavar, G.; Chandra, T.; Strom, R.; Sturman, D.: A Case for Message Oriented Middleware; in Proceedings of the 13th International Symposium on Distributed Computing, Jayanti, P. (Ed.), Lecture Notes in Computer Science 1693, Bratislava, Slovak Republic, September 27-29, 1999, pp. 1-18.
- Bent98 Bentz, Christian: Objektorientierte Realisierung einer flexiblen Animationsumgebung für probabilistische Optimierungsverfahren in Java; Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, Karlsruhe, 1998.
- Berd02 Berdux, J.: Integrationskonzept für den Entwurf multimedialer Umgebungen; Dissertation am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe, Elektronisches Volltextarchiv (EVA) der Universitätsbibliothek der Universität Karlsruhe, 2002.

- BeSy00a Berdux, J.; Syrjakow, M.: Java Media Tool - A Multiagent System for Real and Virtual Multimedia Environments; Eighteenth IASTED International Conference on Applied Informatics (AI 2000), Innsbruck, Austria, February 14-17, 2000.
- BeSy00b Berdux, J.; Syrjakow, M.: Subject-Oriented Development of Real and Virtual Multimedia Environments; Eighteenth IASTED International Conference on Applied Informatics (AI 2000), Innsbruck, Austria, February 14-17, 2000.
- BeSy00c Berdux, J.; Syrjakow, M.: The Design of Alice's World; World Multiconference on Systematics, Cybernetics and Informatics (SCI 2000), Vol. 6(2), Orlando, Florida, July 23-26, 2000, pp. 79-84.
- Blac99 Black, Uyles: Internet-Technologien der Zukunft; Addison-Wesley, 1999.
- BlOe75 Blum, E.; Oettli, W.: Mathematische Optimierung - Grundlagen und Verfahren; Springer-Verlag, 1975.
- BoDS69 Box, M.J.; Davies, D.; Swann, W.H.: Nonlinear Optimization Techniques; ICI Monograph 5, Oliver and Boyd, Edinburgh, 1969.
- BoGr93 Bomze, I.M.; Grossmann, W.: Optimierung - Theorie und Algorithmen; BI Wissenschaftsverlag, 1993.
- BoRJ99 Booch, G.; Rumbaugh, J.; Jacobson I.: Das UML-Benutzerhandbuch; Addison-Wesley, 1999.
- Bosc00 Bosch, Jan: Design and use of software architectures: Adopting and evolving a product-line approach; Addison-Wesley, 2000.
- Boss92 Bossel, Hartmut: Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme; Vieweg, 1992.
- Box65 Box, M.J.: A new Method of Constrained Optimization and a Comparison with other Methods; Comp. J. 8, pp. 42-52, 1965.
- Brau99 Braun, Torsten: IPnG: neue Internet-Dienste und virtuelle Netze: Protokolle, Programmierung und Internetworking; dpunkt.verlag, 1999.
- Brow00 Brown, Alan W.: Large-Scale, Component-Based Development; Prentice Hall, 2000.
- BrUP99 1999 International Conference on Web-based Modeling and Simulation (part of the 1999 SCS Western MultiConference on Computer Simulation WMC'99); Bruzzone, A.G.; Uhrmacher, A.; Page, E.H. (editors); San Francisco, USA, January 17-20, 1999, Society for Computer Simulation (Publisher).
- Burk99 Burkhardt, Rainer: UML - Unified Modeling Language - Objektorientierte Modellierung für die Praxis; Addison-Wesley, 1999.

- BuSt96 Buss, A.H.; Stork, K.A.: Discrete Event Simulation on the World Wide Web Using Java; in Proceedings of the 1996 Winter Simulation Conference, Coronado, CA, USA, 8-11 December 1996, pp. 780-785.
- CEKL99 Chen, P.P.; Embley, D.W.; Kouloumdjian, J.; Liddle, S.W.; Roddick, J.F. (eds.): Advances in Conceptual Modeling; Proceedings of the ER '99 Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling; Paris, France, November 15 - 18, Lecture Notes in Computer Science 1727; Springer-Verlag, 1999.
- Cern85 Cerny, V.: Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm; Journal of Optimization Theory and Applications 45, pp. 41-51, 1985.
- ChDa01 Cheesman, J.; Daniels J.: UML Components - A Simple Process for Specifying Component-Based Software; Addison-Wesley, 2001.
- Chow99 Chow, W.S. (ed.): Multimedia Information Systems in Practice; Springer, 1999.
- Claa99 Claassen, Sönke: Erweiterung des Simulationsframeworks DESMO-J um höhere Modellierungskonstrukte und Statistikfunktionen; Studienarbeit, Fachbereich Informatik, Universität Hamburg, 1999.
- CoDG99 Corne, David; Dorigo, Marco; Glover, Fred (eds.): New Ideas in Optimization; McGraw-Hill, 1999.
- CoKr99 Conner-Sax, Kiersten; Krol, Ed: The whole Internet: The next Generation; O'Reilly, 1999.
- Come98 Comer, Douglas E.: Computernetzwerke und Internets; Prentice Hall, 1998.
- Couh99 Couch, J.: Java 2 Networking (Java Masters Series); McGraw-Hill Book Company, 1999.
- DäPa98 Däßler, R.; Palm, H.: Virtuelle Informationsräume mit VRML. Informationen recherchieren und präsentieren in 3D; dpunkt-Verlag, Heidelberg, 1998.
- Daut95 Dautenhahn, K.: Artificial Life = Künstliches Leben?; Lexikon-Beitrag in der Zeitschrift KI, Ausgabe 2, 1995.
- DePe00 Denninger, S.; Peters, I.: Enterprise JavaBeans; Addison-Wesley, 2000.
- DoGL96 Dorwarth, H.; Graebner, H.; Lorenz, P.: Simulation und Animation im World Wide Web; in Proceedings der ASIM Fachtagung Simulation und Animation für Planung, Bildung und Präsentation 1996, S. 173 - 184.
- Dotz87 Dotzauer, Ernst: Grundlagen der digitalen Simulation; Hanser, 1987.
- DrJW99 Droste, S.; Jansen, T.; Wegener, I.: Perhaps Not a Free Lunch But At Least a Free Appetizer, GECCO'99, Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, July 13-17, 1999, pp. 833-839.

- Dühr99 Dührkoop, Tim: Die Entstehung und Durchsetzung des Internet: Medienwandel aus betriebswirtschaftlicher Sicht; Dissertation Nr. 2276 an der Universität St. Gallen; Vorländer Verlag, 1999.
- FaSJ99 Fayad, M.E.; Schmidt, D.C.; Johnson, R.E.: Implementing Application Frameworks: Object-Oriented Frameworks at Work; Wiley, 1999.
- FiHS98 1998 International Conference on Web-Based Modeling and Simulation (part of the 1998 SCS Western MultiConference on Computer Simulation WMC'98); Fishwick, P.A.; Hill, D.R.C.; Smith, R. (editors); San Diego, California, USA, January 11-14, 1998, Society for Computer Simulation (Publisher).
- FiLM97 Finin, T.; Labrou, Y.; Mayfield, J.: KQML as an Agent Communication Language; in Software Agents, J. Bradshaw (edt.), The MIT Press, 1997.
- Fish78 Fishman, George S.: Principles of Discrete Event Simulation; Wiley, 1978.
- Fish95 Fishwick, P.A.: Simulation Model Design and Execution: Building Digital Worlds; Prentice Hall, 1995.
- Fish96 Fishwick, P.A.: Web-Based Simulation: Some Personal Observations; in Proceedings of the 1996 Winter Simulation Conference, Coronado, CA, USA, 8-11 December 1996, pp. 772-779.
- Fish01 Fishman, George S.: Discrete Event Simulation - Modeling, Programming, and Analysis, Springer Verlag, 2001.
- FINM99 Floreano, D.; Nicoud, J.-D.; Mondada, F. (eds.): Advances in Artificial Life: Proceedings of the 5th European Conference, ECAL'99, Lausanne, Switzerland, September 13-17, Lecture Notes in Computer Science 1674; Springer Verlag, 1999.
- FoFI97 Fonseca, C.M.; Fleming, P.J.: Multiobjective Optimization; in [BäFM97], Chapter C4.5.
- FrLa99 Friedrich, H.; Lange, C.: Stochastische Prozesse in Natur und Technik - Modellierung, Simulation, Zuverlässigkeit; Verlag Harri Deutsch, 1999.
- FrLo79 Frank, M.; Lorenz, P.: Simulation diskreter Prozesse; VEB Fachbuchverlag Leipzig, 1979.
- FrSa47 Friedmann, M.; Savage, L.J.: Planning Experiments Seeking Maxima; in Selected Techniques of Statistical Analysis for Scientific and Industrial Research and Production and Management Engineering, McGraw-Hill, New York, 1947.
- Fuji00 Fujimoto, R.M.: Parallel and Distributed Simulation Systems; Wiley (Wiley Series on Parallel and Distributed Computing), 2000.
- Gold89 Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning; Addison-Wesley, 1989.

- Gram92 Grams, Timm: Simulation: Strukturiert und objektorientiert programmiert; BI Wissenschaftsverlag, 1992.
- GrFi85 Grefenstette, J.J.; Fitzpatrick, J.M.: Genetic Search with Approximate Function Evaluations; Proceedings of the First International Conference on Genetic Algorithms and their Applications, Carnegie-Mellon University Pittsburgh, PA, USA, July 24-26, 1985, pp. 112-118.
- Grif98 Griffel, Frank: Componentware: Konzepte und Techniken eines Softwareparadigmas; dpunkt-Verlag, 1998.
- GrKR01 Grötschel, M.; Krumke, S.O.; Rambau J. (editors): Online Optimization of Large Scale Systems; Springer, 2001.
- GrTh00 Gruhn, V.; Thiel, A.: Komponentenmodelle - DCOM, JavaBeans, Enterprise JavaBeans, CORBA; Addison-Wesley, 2000.
- HaBa94 Hammer, P.L.; Balci, O. (eds.): Annals of Operations Research – Simulation and Modeling (Volume 53); J.C. Baltzer AG, 1994.
- Hard93 Hardhienata, Soewarto: Numerische Optimierungsstrategie für Simulationsmodelle mit Anwendungen in Informatik und Verfahrenstechnik, Dissertation der Technischen Fakultät der Universität Erlangen-Nürnberg, 1993.
- HaZo95 Haas, M; Zorn, W.: Methodische Leistungsanalyse von Rechensystemen; Oldenbourg Verlag; 1995.
- HeAN99 Hegering, H.-G.; Abeck, S.; Neumair, B.: Integriertes Management vernetzter Systeme - Konzepte, Architekturen und deren betrieblicher Einsatz; dpunkt.verlag, 1999.
- HeKi97 Healy, K.J.; Kilgore, R.A.: Silk: A Java-Based Process Simulation Language; in Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, 7-10 December 1997, pp. 475-482.
- HeKi98 Healy, K.J.; Kilgore, R.A.: Introduction to Silk and Java-Based Simulation; in Proceedings of the 1998 Winter Simulation Conference, Washington, D.C., 13-16 December 1998, pp. 327-334.
- HeKK98 Healy, K.; Kilgore, R.; Kleindorfer, G.B.: Silk: Usable and Reusable, Java-Based, Object Oriented Simulation; in Proceedings of the 12th European Simulation Multiconference, Society for Computer Simulation International, Ghent, Belgium, 1998.
- Henn97 Hennig, Alexander: Die andere Wirklichkeit - Virtual Reality - Konzepte, Standards, Lösungen; Addison Wesley, 1997.
- HeSi00 Herzum, P; Sims, O.: Business Component Factory - A Comprehensive Overview of Component-Based Development for the Enterprise; Wiley & Sons, 2000.

- HoHo71 Hoffmann, Ulrich; Hofmann, Hans: Einführung in die Optimierung; Verlag Chemie GmbH, 1971.
- HoJe61 Hooke, R.A.; Jeeves, T.A.: Direct Search Solution for Numerical and Statistical Problems; Journal ACM 8, pp. 212-221, 1961.
- Holl75 Holland, J.: Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.
- Holz98 Holzner Steven: XML Complete; McGraw-Hill, 1998.
- HoMc98 Howell, F.W.; McNab, R.: Simjava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling; in Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation, San Diego, CA, 11-14 January 1998.
- HoSh98 Hoque, Reaz; Sharma, Tarun: Programming Web Components; McGraw-Hill, 1998.
- HuBe95 Huber, K.-P.; Berthold, M.R.: Building Precise Classifiers with Automatic Rule Extraction; Proceedings of the IEEE International Conference on Neural Networks ICNN'95, Perth, Western Australia, November 27 - December 1, 1995, Vol. 3, pp. 1263-1268.
- HuRi99 Huntbach, Matthew M.; Ringwood, Gream A.: Agent-Oriented Programming: From Prolog to Guarded Definite Clauses; Lecture Notes in Artificial Intelligence 1630, Springer-Verlag, 1999.
- HuSS93 Huber, K.-P.; Sytjakow M.; Szczerbicka H.: Extracting Knowledge Supports Model Optimization; Proceedings of the International Simulation Technology Multiconference (SIMTEC'93), San Francisco, California, USA, November 7-10, 1993, pp. 237-242.
- IaDA98 Iazeolla, G; D'Ambrogio, A.: A Web-Based Environment for the Reuse of Simulation Models; in Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation (part of the 1998 SCS Western MultiConference on Computer Simulation WMC'98); San Diego, California, USA, January 11-14, 1998.
- Info99 Informationslogistik für unternehmens- und branchenübergreifende Kooperation; Forschungsschwerpunkt an der Universität Karlsruhe, Abschlussbericht, 1999.
- Info01 Informationslogistik für die internetbasierte Prozessinteraktion bei der branchenübergreifenden Kooperation, Forschungsschwerpunkt an der Universität Karlsruhe, Zwischenbericht, 31.12.2001.
- Info02 Informationslogistik für die internetbasierte Prozessinteraktion bei der branchenübergreifenden Kooperation, Forschungsschwerpunkt an der Universität Karlsruhe, Abschlussbericht, Oktober 2002.

- JaBR99 Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process; Addison Wesley, 1999.
- Jain91 Jain, Raj: The Art of Computer Systems Performance Analysis; Wiley, 1991.
- JLKS00 Jesse, R.; Lange, S.; Klein, U.; Schulze, T.: Animation in einer HLA-Federation; in Proceedings of Simulation und Visualisierung 2000, Schulze, T.; Lorenz, P.; Hinz, V. (eds.), Magdeburg, Germany, March 23-24, 2000, pp. 55-67.
- JüKW00a Jüngel, M.; Kindler, E.; Weber, M.: Towards a Generic Interchange Format for Petri Nets; in Proceedings of the Meeting on XML/SGML based Interchange Formats for Petri Nets, a satellite event at 21st International Petri Net Conference, Aarhus, Denmark, June 27, 2000, pp. 1-6.
- JüKW00b Jüngel, M.; Kindler, E.; Weber, M.: The Petri Net Markup Language; in Proceedings of the 7th Workshop on "Algorithmen und Werkzeuge für Petrinetze", Fachberichte Informatik, No. 7-2000, Koblenz, Germany, October 2-3, 2000, pp. 47-52.
- KaSm94 Kaufmann, W.J.; Smarr, L.L.: Simulierte Welten; Spektrum Akademischer Verlag, 1994.
- KaZY97 Kaylan, A.R.; Ziya, S.; Yilmaz, O.: Web-based Architecture for Simulation Modeling and Analysis; in Proceedings of the 1997 European Simulation Multi-conference, Istanbul, Turkey, June 1-4, 1997.
- KiGV82 Kirkpatrick, S.; Gelatt Jr., C.D.; Vecchi, M.P.: Optimization by Simulated Annealing; IBM Research Report RC 9355, 1982.
- KiGV83 Kirkpatrick, S.; Gelatt Jr., C.D.; Vecchi, M.P.: Optimization by Simulated Annealing; Science 220, pp. 671-680, 1983.
- Kirk84 Kirkpatrick, S.: Optimization by Simulated Annealing: Quantitative Studies; Journal of Statistical Physics 34, pp. 975-986, 1984.
- KlSW99 Klusch, Matthias; Shehory, Onn M.; Weiss, Gerhard (Eds.): Proceedings of the Third International Workshop on Cooperative Information Agents (CIA'99), Springer Verlag, Lecture Notes in Computer Science 1652, Uppsala, Sweden, July 31 – August 2, 1999.
- Klüg01 Klügl, Franziska: Multiagentensimulation - Konzepte, Werkzeuge, Anwendung; Addison-Wesley; 2001.
- KrHM97 Kreutzer, W.; Hopkins, J.; Mierlo, M.v.: SimJAVA - A Framework for Modeling Queueing Networks in Java; in Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, 7-10 December 1997, pp. 483-488.
- KuPa00 Kuljis, J; Paul, R.J.: A Review of Web Based Simulation: Whither We Wander?; in Proceedings of the 2000 Winter Simulation Conference (WSC 2000), Orlando, Florida, USA, December 10-13, 2000.

- KuWD99 Kuhl, F.; Weatherly, R.; Dahmann, J.: Creating Computer Simulation Systems - An Introduction to the High Level Architecture; Prentice Hall, 1999.
- Lang96 Langton, Christopher G. (ed.): Artificial Life: An Overview; The MIT Press, 1996.
- LaSU99 Lantzsch, G.; Straßburger, S.; Urban, C.: HLA-basierte Kopplung der Simulationssysteme Simplex III und SLX; in Proceedings of Simulation und Visualisierung, Magdeburg, Germany, March 4-5, 1999.
- LDRS97 Lorenz, P.; Dorwarth, H.; Ritter, K.-C.; Schriber T.J.: Towards a Web Based Simulation Environment; in Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, 7-10 December 1997.
- Lech99 Lechler, Tim: Entwurf und Implementierung eines Frameworks für diskrete Simulatoren in Java; Diplomarbeit, Fachbereich Informatik, Universität Hamburg, 1999.
- Lehm77 Lehman, Richard S.: Computer Simulation and Modeling: An Introduction; Lawrence Erlbaum Associates Publishers, 1977.
- Lieb92 Liebl, Franz: Simulation: Problemorientierte Einführung; Oldenbourg, 1992.
- Lief99 Liefänder, Matthias: Animation anpassungsfähiger autonomer Agenten in einer dynamischen Umwelt; Studienarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, Karlsruhe, 1999.
- Lisc99 Lischka, Rainer: Techniken zur Erweiterung von WWW-Servern für interaktive Anwendungen; Dissertation an der Johannes-Kepler-Universität Linz, Universitätsverlag Rudolf Trauner, 1999.
- Litt99 Little, M.C.: JavaSim - User's Guide; Department of Computing Science, Computing Laboratory, University of Newcastle upon Tyne, UK, 1999.
- Löwy01 Löwy, Juval: COM and .NET Component Services; O'Reilly, 2001.
- Lore99 Lorenz, P.: Web-basierte Simulation und HLA; in "Modellierung, Simulation und Künstliche Intelligenz"; H. Szczerbicka, T. Uthmann (Hrsg.); SCS European Publishing House, 1999, S. 417-436.
- Mars99 Marschall, Georg: Zum modellbasierten Entwurf dezentraler Automatisierungssysteme: Modellbildung, Identifikation und Validierung dezentraler Automatisierungssysteme auf der Basis von Petri-Netzen am Beispiel von Profibus-Netzwerken; Fortschritt-Berichte VDI, Reihe 8, Nr. 769, VDI-Verlag, 1999.
- McHo96 McNab, R.; Howell, F.W.: Using Java for Discrete Event Simulation; in Proceedings of the Twelfth UK Computer and Telecommunications Performance Engineering Workshop (UKPEW), Univ. of Edinburgh, September 1996, pp. 219-228.
- Mehl94 Mehl, Horst: Methoden verteilter Simulationen; Vieweg Verlag, 1994.



- Mich92 Michalewicz, Zbigniew: Genetic Algorithms + Data Structures = Evolution Programs; Springer, 1992.
- MiSX00 Miller, J.A.; Seila, A.F.; Xiang, X.: The JSIM Web-Based Simulation Environment; in Future Generation Computer Systems (FGCS), Special Issue on Web-Based Modeling and Simulation, Vol. 17, No. 2, pp. 119-133, Elsevier North-Holland, October 2000.
- MoHa90 Molnár, I.; Hardhienata, S.: Comparison of Optimization Strategies for Combined Simulation; in Proceedings of the European Simulation Multiconference ESM'90, Nürnberg, Germany, June 10-13, 1990, pp. 198-203.
- Morg84 Morgan, Byron J.T.: Elements of Simulation; Chapman and Hall, 1984.
- Müll97 Müller, B.: Modellierung von Geschäftsprozessen; Studienarbeit am Institut für Rechnerentwurf und Fehlertoleranz, Fakultät für Informatik, Universität Karlsruhe, 1997.
- MüSB91 Mühlenbein, H.; Schomisch, M.; Born, J.: The Parallel Genetic Algorithm as Function Optimizer; Proceedings of the Fourth International Conference on Genetic Algorithms, University of California, San Diego, USA, July 13-16, 1991, pp. 271-278.
- Myer02 Myerson, J.M.: The Complete Book of Middleware; Auerbach Pub, 2002.
- Nair97 Nair, R.S.: JSIM: A Java-Based Query Driven Simulation and Animation Environment; Masters Thesis (M.S. in CS Degree), University of Georgia, March 1997, employed by Sun Microsystems, Inc., Mountain View, CA.
- Nara00 Narayanan, S.: Web-Based Modeling and Simulation; in Proceedings of the 2000 Winter Simulation Conference (WSC 2000), Orlando, Florida, USA, December 10-13, 2000.
- Nath02 Nathan, Adam: .NET and COM: The Complete Interoperability Guide; Sams, 2002.
- Oest99 Oestereich, B. (Hrsg.): Erfolgreich mit Objektorientierung; Oldenbourg, 1999.
- Oest01 Oestereich, B.: Die UML-Kurzreferenz für die Praxis; Oldenbourg, 2001.
- Olst99 Olderog, Ernst-Rüdiger; Steffen, Bernhard (eds.): Correct System Design: Recent Insights and Advances; Springer, 1999.
- ONei98 O'Neil, J.: JavaBeans Programming from the Ground Up; Osborne Publishing, 1998.
- Page91 Page, Bernd: Diskrete Simulation: Eine Einführung mit Modula-2; Springer, 1991.

- Page98 Page, E.H.: The Rise of Web-Based Simulation: Implications for the High Level Architecture; in Proceedings of the 1998 Winter Simulation Conference, Washington, D.C., 13-16 December 1998, pp. 1663-1668.
- Page99 Page, E.H.: Beyond Speedup: PADS, the HLA and Web-Based Simulation; in Proceedings of the 13th Workshop on Parallel and Distributed Simulation, R.M. Fujimoto and S. Turner, Eds., Atlanta, GA, 1-4 May 1999, pp. 2-9.
- PaGR98 Page, E.H.; Griffin, S.P.; Rother, S.L.: Providing Conceptual Framework Support for Distributed Web-Based Simulation within the High Level Architecture; in Proceedings of SPIE Vol. 3369: Enabling Technology for Simulation Science II, Orlando, FL, 13-17 April 1998, pp. 287-292.
- PaLC00 Page B.; Lechler, T.; Claassen, S.: Objektorientierte Simulation in Java mit dem Framework DESMO-J; BoD GmbH, Norderstedt, 2000.
- PaMG97 Page, E.H.; Moose, R.L.; Griffin, S.P.: Web-Based Simulation in Simjava using Remote Method Invocation; in Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, 7-10 December 1997, pp. 468-474.
- PaOp00 Page, E.H.; Opper, J.M.: Investigating the Application of Web-Based Simulation Principles within the Architecture for a Next-Generation Computer Generated Forces Model; in Future Generation Computer Systems, Elsevier Science Publishing, Vol. 17, 2000, pp. 159-169.
- PBFH98 Page, E.H.; Buss, A.; Fishwick, P.A.; Healy, K.J.; Nance, R.E.; Paul, R.J.: The Modeling Methodological Impacts of Web-Based Simulation; in Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation, San Diego, CA, 11-14 January 1998, pp. 123-128.
- PBFH00 Page, E.H.; Buss, A.; Fishwick, P.A.; Healy, K.J.; Nance, R.E.; Paul, R.J.: Web-Based Simulation: Revolution or Evolution?; in ACM Transactions on Modeling and Computer Simulation, Vol. 10, No. 1, January 2000, pages 3-17.
- Prüf82 Prüfer, Hans-Peter: Parameteroptimierung - Ein Werkzeug des rechnerunterstützten Konstruierens; Dissertation am Institut für Konstruktionstechnik der Ruhr-Universität Bochum, Schriftenreihe 82.2, 1982.
- Pütt00 Püttmann, Dietmar: Künstliches Leben - Simulation und Visualisierung durch ein verteiltes Multiagenten-System; Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, Karlsruhe, 2000.
- PuRö01 Puder, A.; Römer, K.: Middleware für verteilte Systeme - Konzepte und Implementierung anhand der Corba-Plattform MICO; dpunkt.verlag, 2001.
- RBPE94 Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorenzen, W.: Objektorientiertes Modellieren und Entwerfen; Eine Coedition der Verlage Carl Hanser und Prentice-Hall, 1994.
- Rech73 Rechenberg, Ingo: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution; Friedrich Frommann Verlag, 1973.

- ReF196     Renders, J.-M.; Flasse, S.P.: Hybrid Methods using Genetic Algorithms for Global Optimization; in IEEE Transactions on Systems, Man, and Cybernetics (Part B), 26(2):243-258, 1996.
- RePo02     Rechenberg, P.; Pomberger, G. (Hrsg.): Informatik-Handbuch (3. Auflage); Carl Hanser Verlag, 2002.
- RKSD98     Ritter, K.C.; Klein, U.; Strassburger, S.; Diessner, M.: Web-basierte Animation verteilter Simulationen auf Basis der High Level Architecture; Tagung Simulation und Visualisierung 1998, Lorenz, P.; Preim, B. (eds.), Magdeburg, Germany, 1998, S. 32-40.
- RoAJ01     Roman, E.; Ambler, S.W.; Jewell, T.: Mastering Enterprise JavaBeans (2nd Edition); John Wiley & Sons, 2001.
- Rose60     Rosenbrock, H.H.: An Automatic Method for Finding the Greatest or Least Value of a Function; Comp. J. 3, pp. 175-184, 1960.
- RuHK00     Ruh, William; Herron, Thomas; Klinker, Paul: IIOP Complete: Understanding CORBA and Middleware Interoperability; Addison-Wesley, 2000.
- Rukz01     Rukzio, Enrico: Formate, Technologien und Architekturkonzepte für 3D-Web-Applikationen; Belegarbeit an der Technischen Universität Dresden, Fakultät für Informatik, Institut für Software- und Multimediatechnik, August 2001.
- Same97     Sameting, J.: Software Engineering with Reusable Components; Springer Verlag, 1997.
- SaZe98     Sarjoughian, H.S.; Zeigler B.P.: DEVSJava: Basis for a DEVS-based Collaborative M&S Environment; in Proceedings of the 1998 SCS International Conference on Web-Based Modeling and Simulation, San Diego, CA, 11-14 January 1998.
- SBPS01     Syrjakow, M.; Berdux, J.; Püttmann, D.; Szczerbicka, H.: Simulation and Visualization of Distributed Artificial Life Scenarios; Proceedings of the 3rd International Conference on Intelligent Processing and Manufacturing of Materials (IPMM2001), Vancouver, British Columbia, Canada, July 29 - August 3, 2001.
- SBSZ99     Syrjakow, M.; Berdux, J.; Szczerbicka, H.; Zimmermann, B.; Otto, A.: A Flexible Java-Based Authoring System for Building Multimedia-Enriched CBT Applications; 13th European Simulation Multiconference (ESM'99), Warsaw, Poland, June 1-4, 1999, Volume I, pp. 324-328.
- SCED89     Schaffer, J.D.; Caruana, R.A.; Eshelman, L.J.; Das R.: A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization; Proceedings of the Third International Conference on Genetic Algorithms, George Mason University, June 4-7, 1989, pp. 51-60.
- Schi02     Schillinger, Sabine: Entwicklung eines XML-basierten Austauschformates für Stochastische Petri-Netze; Studienarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, Karlsruhe, 2002.

- Schm85 Schmidt, B.: Systemanalyse und Modellaufbau; Springer Verlag, 1985.
- Schw77 Schwefel, H.-P.: Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie; Birkhäuser-Verlag, 1977.
- ScSc98 Schader, Martin; Schmidt-Thieme, Lars: Java: Einführung in die objektorientierte Programmierung; Springer-Verlag, 1998.
- ScSK00 Schulze, T.; Straßburger, S.; Klein, U.: HLA-Federate Reproduction Procedures in Public Transportation Federations; Proceedings of the 2000 Summer Computer Simulation Conference, Vancouver, Canada, July 16-20, 2000.
- ScWe01 Schneider, U.; Werner, D.: Taschenbuch der Informatik (4. Auflage); Fachbuchverlag Leipzig, 2001.
- SeCr02 Serain, D.; Craig, I. (Translator): Middleware and Enterprise Application Integration; Springer Verlag, 2002.
- SeGu00 Seemann, J.; Gudenberg, J.W.: Software-Entwurf mit UML; Springer, 2000.
- Seib97 Seibt, Frank: Konzept und Komponenten einer Web-basierten Simulationsentwicklungsumgebung, Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, Institut für Simulation und Graphik, Magdeburg, 1997.
- SeSB98 Seibt, F.; Schumann, M.; Beikirch, J.: Concept and Components for a Web-based Simulation Environment (WBSE); in Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation (part of the 1998 SCS Western MultiConference on Computer Simulation WMC'98); San Diego, California, USA, January 11-14, 1998.
- Sesh99 Seshadri, Govind: Enterprise Java Computing: Applications and Architecture; Cambridge University Press, 1999.
- SiBl00 2000 International Conference on Web-Based Modeling and Simulation (part of the 2000 SCS Western MultiConference on Computer Simulation WMC'00); Signorile, R.; Blais, C. (editors); San Diego, California, USA, January 23-27, 2000, Society for Computer Simulation (Publisher).
- Sieg99 Siegmund, Gerd (Hrsg.): Intelligente Netze: Technik, Dienste, Vermarktung; Hüthig Verlag, 1999.
- SiWi02 2002 International Conference on Web-Based Modeling and Simulation (part of the 2002 SCS Western MultiConference on Computer Simulation WMC'02); Signorile, R.; Wilsey, P.A. (editors); San Antonio, Texas, USA, January 27-31, 2002, Society for Computer Simulation (Publisher).
- SiWM01 2001 International Conference on Web-Based Modeling and Simulation (part of the 2001 SCS Western MultiConference on Computer Simulation WMC'01); Signorile, R.; Wilsey, P.A.; Miller, J.A. (editors); Phoenix, Arizona, USA, January 7-11, 2001, Society for Computer Simulation (Publisher).

- SoSo99 Sodhi, Jag; Sodhi, Prince: Software Reuse: Domain Analysis and Design Processes; McGraw-Hill, 1999.
- Sout40 Southwell, R.V.: Relaxation Methods in Engineering Science: A Treatise on Approximate Computation; Oxford University Press, Oxford, 1940.
- Sout46 Southwell, R.V.: Relaxation Methods in Theoretical Physics; Clarendon Press, Oxford, 1946.
- Spal98 Spall, J.C.: An Overview of the Simultaneous Perturbation Method for Efficient Optimization; Johns Hopkins APL Technical Digest, Vol. 19, pp. 482–492.
- Spal99 Spall, J. C.: Stochastic Optimization: Stochastic Approximation and Simulated Annealing; in Encyclopedia of Electrical and Electronics Engineering, J.G. Webster (editor), Wiley, New York, Vol. 20, pp. 529–542.
- SpHH62 Spendley, W.; Hext, G.R.; Himsworth, F.R.: Sequential Application of Simplex Designs in Optimization and Evolutionary Operation; Technometrics 4, pp. 441-461, 1962.
- SSBH96 Syrjakow, M.; Szczerbicka, H.; Berthold, M.R.; Huber, K.-P.: Acceleration of Direct Model Optimization Methods by Function Approximation; Proceedings of the 8th European Simulation Symposium (ESS'96), Genoa, Italy, October 24-26, 1996, Volume II, pp. 181-186.
- StCe94 Standridge, C.R.; Centeno, M.A.: Concepts for Modular Simulation Environments; in Proceedings of the 1994 Winter Simulation Conference, Lake Buena Vista, FL, USA, December 11-14, 1994, pp. 657 - 663.
- StKI98 Straßburger, S.; Klein, U.: Integration des Simulators SLX in die High Level Architecture; Proceedings of the Conference on Simulation und Visualisation, Preim, B.; Lorenz, P. (Eds.), Magdeburg, Germany, March 5-6, 1998, pp. 32-40.
- Stra99 Straßburger, S.: On the HLA-based Coupling of Simulation Tools; Proceedings of the 1999 European Simulation Multiconference, Warsaw, Poland, June 1-4, 1999.
- Stra01a Straßburger, S.: HLA- und Web-basierte Simulation: Ziele, Unterschiede und Synergien; Proceedings of the Conference on Simulation und Visualisation, Schulze, T.; Hinz, V.; Schlechtweg, S. (Eds.), Magdeburg, Germany, March 22-23, 2001.
- Stra01b Straßburger, S.: Distributed Simulation Based on the High Level Architecture in Civilian Application Domains; Dissertationsschrift, Fakultät für Informatik, Otto-von-Guericke Universität Magdeburg, herausgegeben in SCS-Series "Advances in Simulation", SCS-Europe BVBA, 2001, ISBN 1-56555-218-0.
- Stra01c Straßburger, S.; Schulze, T.: Verteilte- und Web-basierte Simulation: Gemeinsamkeiten und Unterschiede; Proceedings of the 15<sup>th</sup> Simulation Symposium ASIM 2001, Paderborn, Germany, Sept. 11-14, 2001.

- Swan64 Swann, W.H.: Report on the Development of a new Direct Searching Method of Optimization; ICI, Centr. Instr. Lab., Research Note 64-3, Middlesborough, Yorks., June 1964.
- SyBS00 Syrjakow, M.; Berdux, J.; Szczerbicka, H.: Interactive Web-based Animations for Teaching and Learning; Winter Simulation Conference (WSC 2000), Orlando, Florida, USA, December 10-13, 2000, pp. 1651-1659.
- Syrj97 Syrjakow, M.: Verfahren zur effizienten Parameteroptimierung von Simulationsmodellen; Dissertation am Institut für Rechnerentwurf und Fehlertoleranz der Universität Karlsruhe, Berichte aus der Informatik, Shaker Verlag, 1997.
- Syrj99 Syrjakow, M.: Effiziente Parameteroptimierung von Simulationsmodellen; in "Modellierung, Simulation und Künstliche Intelligenz"; H. Szczerbicka, T. Uthmann (Hrsg.); SCS European Publishing House, 1999, S. 333-363.
- SySy00 Syrjakow, E.; Syrjakow, M.: Parameter Optimization of Complex Simulation Models; in Proceedings of the Workshop on System Design Automation, Rathen, Germany, March 13-14, 2000, pp. 114-121.
- SySy01 Syrjakow, E.; Syrjakow, M.: Parameter Optimization of Complex Simulation Models; in System Design Automation - Fundamentals, Principles, Methods, Examples; R. Merker, W. Schwarz (eds.); Kluwer Academic Publishers, March 2001, pp. 233-246.
- SySy02 Syrjakow, E.; Syrjakow, M.: Web-based Business Process Modeling and Optimisation; in Proceedings of the International Conference on Internet and Multimedia Systems and Applications (IMSA'02); Kauai, Hawaii, USA, August 12-14, 2002, pp. 124-129.
- SySz92 Syrjakow, M.; Szczerbicka, H.: An Efficient Method for Performability Evaluation of a Multiprocessor Machine under Consideration of Bus Failures; Proceedings of the European Simulation Multiconference (ESM'92), York, UK, June 1-3, 1992, pp. 261-266.
- SySz93a Syrjakow, M.; Szczerbicka, H.: REMO - Ein Werkzeug zur Modelloptimierung; in den ASIM Mitteilungen aus den Arbeitskreisen zum 6. Workshop "Simulation und Künstliche Intelligenz" an der Universität Karlsruhe, Heft Nr. 35, April 1993.
- SySz93b Syrjakow, M.; Szczerbicka, H.: REMO - Ein Werkzeug zur Modelloptimierung; in Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen (MMB), Kurzberichte und Werkzeughvorstellungen, Hrsg. Walke/Spaniol, 7. ITG/GI-Fachtagung, Aachen, 21.-23. September, 1993.
- SySz93c Syrjakow, M.; Szczerbicka, H.: REMO - A Tool for the Automatic Optimization of Performance Models; Proceedings of the European Simulation Symposium (ESS'93), Delft, NL, October 25-28, 1993, pp. 597-603.

- SySz93d Syrjakow, M.; Szczerbicka, H.: Efficient Model Optimization with REMO; Proceedings of the Fifth Workshop on Neural Networks (WNN93/FNN93), San Francisco, USA, November 7-10, 1993, pp. 101-106.
- SySz94a Syrjakow, M.; Szczerbicka, H.: Optimization of Simulation Models with REMO; Proceedings of the European Simulation Multiconference (ESM'94), Barcelona, Spain, June 1-3, 1994, pp. 274-281.
- SySz94b Syrjakow, M.; Szczerbicka, H.: Ein hybrider Optimierungsalgorithmus zur Optimierung von Simulationsmodellen; 18. Deutsche Jahrestagung für Künstliche Intelligenz, KI-94 Workshops, Saarbrücken, 18.-23. September, 1994, S. 228-229.
- SySz95a Syrjakow, M.; Szczerbicka, H.: Simulation and Optimization of Complex Technical Systems; Proceedings of the 1995 Summer Computer Simulation Conference (SCSC'95), Ottawa, Ontario, Canada, July 24-26, 1995, pp. 86-95.
- SySz95b Syrjakow, M.; Szczerbicka, H.: Combination of Direct Global and Local Optimization Methods; Proceedings of the International Conference on Evolutionary Computing (ICEC'95), Perth, Western Australia, November 29 - December 1, 1995.
- SySz97a Syrjakow, M.; Szczerbicka, H.: REMO - REsearch Model Optimization Package; Tool Descriptions from the 9<sup>th</sup> International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (Performance Tools 1997) and the 7<sup>th</sup> International Workshop on Petri Nets and Performance Models (PNPM'97), Saint-Malo, France, June 3-6, 1997, pp. 20-22.
- SySz97b Syrjakow, M.; Szczerbicka, H.: Efficient Methods for Parameter Optimization of Simulation Models; Proceedings of the 1st World Congress on Systems Simulation (WCSS'97), Singapore, Republic of Singapore, September 1-3, 1997, pp. 54-59.
- SySz99a Syrjakow, M.; Szczerbicka, H.: Java-Based Animation of Probabilistic Search Algorithms; International Conference on Web-based Modeling and Simulation (part of the Western MultiConference WMC'99), San Francisco, USA, January 17-20, 1999, pp. 182-187.
- SySz99b Syrjakow, M.; Szczerbicka, H.: Efficient Parameter Optimization based on Combination of Direct Global and Local Search Methods; in "Evolutionary Algorithms" (Ima Volumes in Mathematics and Its Applications, Vol. 111); L.D. Davis, K. De Jong, M.D. Vose, L.D. Whitley (eds.); Springer Verlag New York, 1999, pp. 227-249.
- SzSB98 Szczerbicka, H.; Syrjakow, M.; Becker, M.: Genetic Algorithms: A Tool for Modelling, Simulation, and Optimization of Complex Systems; in: Cybernetics and Systems: An International Journal, Vol. 29, No. 7, pp. 639-660, 1998.
- SzUt99 Szczerbicka, H.; Uthmann, T. (Hrsg.): "Modellierung, Simulation und Künstliche Intelligenz"; SCS European Publishing House, 1999.

- Szyp99 Szyperski, Clemens: Component Software - Beyond Object-Oriented Programming; Addison-Wesley, 1999.
- Thro94 Thro, Ellen: Künstliches Leben - eine spielerische Entdeckungsreise: Einführung in Theorie und Praxis einer neuen Wissenschaft; Addison-Wesley, 1994.
- TöZi89 Törn, Aimo; Zilinskas, Antanas: Global Optimization; Springer Verlag, 1989.
- Vacca97 Vacca, John R.: VRML; Sybex-Verlag, 1997.
- Vett95 Vetter, Max: Objektmodellierung: Eine Einführung in die objektorientierte Analyse und das objektorientierte Design; Teubner, 1995.
- Vose99 Vose, Michael D.: The Simple Genetic Algorithm: Foundations and Theory; MIT Press, 1999.
- WaBi99 Warren, Nigel; Bishop, Philip: Java in Practice: Design styles and idioms for effective Java; Addison-Wesley, 1999.
- Wals99 Walser, J.P.: Integer Optimization by Local Search; Springer-Verlag 1999.
- Wend95 Wendt, Oliver: Tourenplanung durch Einsatz naturanaloger Verfahren; Deutscher Universitätsverlag, 1995.
- Wild99 Wilde, Erik: Wilde's WWW: Technical Foundations of the World Wide Web; Springer-Verlag, 1999.
- WoMa97 Wolpert, D.H; Macready, W.G.: No Free Lunch Theorems for Optimization; in IEEE Transactions on Evolutionary Computation, Vol.1, No.1, 67-82, April 1997.
- Wrig95 Wright, Margaret H.: Direct search methods: Once scorned, now respectable; in Numerical Analysis (D. F. Griffiths and G. A. Watson, eds.), Pitman Research Notes in Mathematics, pp. 191-208, Addison Wesley Longman Limited, 1995.
- WRKZ01 Wolf, H.; Roock, S.; Kornstädt, A.; Züllighoven, H.; Gryczan, G.: Das JWAM-Framework und Komponenten - Eine konzeptionelle Bestandsaufnahme; im Tagungsbild des 3. Workshops über "Komponentenorientierte betriebliche Anwendungssysteme" (WKBA 3), veranstaltet vom GI Arbeitskreis 5.10.3, Turowski, K. (Hrsg.), Frankfurt am Main, 4. April 2001.
- Zaha99 Zahavi, Ron: Enterprise Application Integration with CORBA - Component and Web-Based Solutions; Wiley, 1999.
- ZeHS97a Zeigler, B.P.; Hall, S.B.; Sarjoughian, H.S.: DEVSJAVA - User's Guide; The University of Arizona, 1997.
- ZeHS97b Zeigler, B.P.; Hall, S.B.; Sarjoughian, H.S.: DEVSJAVA - Reference Guide; The University of Arizona, 1997.



- Zhan97 Zhang, Z.: A Java-Based Simulation and Animation Environment: JSIM's Foundation Library; Masters Thesis (M.S. in CS Degree), University of Georgia, March 1997, employed by ICL, Greenville, SC.
- ZhSo00 Zhang, Chengqi; Soo, Von-Wun (Eds.): Design and Applications of Intelligent Agents; Proceedings of the Third Pacific Rim International Workshop on Multi-Agents, PRIMA 2000, Melbourne, Australia, August 28-29, 2000; Lecture Notes in Artificial Intelligence 1881, Springer-Verlag, 2000.
- Zimm87 Zimmermann, Hans-Jürgen: Operations Research Methoden und Modelle; Vieweg, 1987.
- Zimm99 Zimmermann, Bernd: Konzeption einer Internet-basierten Lernumgebung zur Vorlesungsbegleitung; Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, Karlsruhe, 1999.
- ZuMR94 Zurada, J.M.; Marks, R.J.; Robinson, C.J. (eds.): Computational Intelligence Imitating Life; IEEE Press, 1994.

## Konferenzbände

- BrSw96 Proceedings of the 1996 Winter Simulation Conference; Brunner, D.; Swain, J. (editors); Coronado, CA, USA, December 8-11, 1996.
- BrUP99 Proceedings of the 1999 International Conference on Web-based Modeling and Simulation (part of the 1999 SCS Western MultiConference on Computer Simulation WMC'99); Bruzzone, A.G.; Uhrmacher, A.; Page, E.H. (editors); San Francisco, CA, USA, January 17-20, 1999, Society for Computer Simulation (Publisher).
- FiHS98 Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation (part of the 1998 SCS Western MultiConference on Computer Simulation WMC'98); Fishwick, P.A.; Hill, D.R.C.; Smith, R. (editors); San Diego, California, USA, January 11-14, 1998, Society for Computer Simulation (Publisher).
- SiBl00 Proceedings of the 2000 International Conference on Web-Based Modeling and Simulation (part of the 2000 SCS Western MultiConference on Computer Simulation WMC'00); Signorile, R.; Blais, C. (editors); San Diego, California, USA, January 23-27, 2000, Society for Computer Simulation (Publisher).
- SiWi02 Proceedings of the 2002 International Conference on Web-Based Modeling and Simulation (part of the 2002 SCS Western MultiConference on Computer Simulation WMC'02); Signorile, R.; Wilsey, P.A. (editors); San Antonio, Texas, USA, January 27-31, 2002, Society for Computer Simulation (Publisher).
- SiWM01 Proceedings of the 2001 International Conference on Web-Based Modeling and Simulation (part of the 2001 SCS Western MultiConference on Computer Simulation WMC'01); Signorile, R.; Wilsey, P.A.; Miller, J.A. (editors); Phoenix, Arizona, USA, January 7-11, 2001, Society for Computer Simulation (Publisher).

## Verweise auf Seiten im World Wide Web

### Zu verschiedenen Themengebieten

#### Common Object Request Broker Architecture

COR1      [http://www.omg.org/technology/documents/corba\\_spec\\_catalog.htm](http://www.omg.org/technology/documents/corba_spec_catalog.htm)

#### Java

JAVA1      <http://java.sun.com/>

JAVA2      <http://java.sun.com/products/>

JAVA3      <http://java.sun.com/products/archive/index.html>

#### Knowledge Query and Manipulation Language

KQML      <http://www.cs.umbc.edu/kqml/>

#### High Level Architecture

HLA1      <http://www.kompetenzzentrum-hla.de/>

HLA2      <https://www.dmso.mil/public/transition/hla/>

HLA3      <https://www.dmso.mil/public/transition/hla/rti/>

#### Petrinetze

PN1      <http://www.daimi.au.dk/PetriNets/tools/>

PN2      <http://www.daimi.au.dk/PetriNets/standardisation/>

PN3      <http://www.di.unito.it/~greatspn/>

PN4      <http://ls4-www.cs.uni-dortmund.de/home/lindemann/software/DSPNexpress/main.html>

PN5      <http://pdv.cs.tu-berlin.de/~timenet/>

### **Softwarearchitektur**

SWA1 <http://www.sei.cmu.edu/architecture/definitions.html>

### **Web Services**

WS1 <http://www.w3.org/2002/ws/>

### **X3D**

X3D1 <http://www.web3d.org/x3d.html>

## **Zu verschiedenen Institutionen**

### **Object Management Group**

OMG1 <http://www.omg.org/>

### **World Wide Web Consortium**

W3C1 <http://www.w3.org/>