Fachhochschule für
Wirtschaft Berlin
Berlin School of Economics
IMB Institute of Management Berlin

Working Papers

Optimierung des Zusammenwirkens maschineller und intellektueller Spezialisten

Author: Prof. Helmut Jarosch

Section: Business & Management
Paper No. 44, 01/2009

Optimierung des Zusammenwirkens maschineller und intellektueller Spezialisten

Prof. Dr. Helmut Jarosch

Paper No. 44, 01/2009

Working Papers of
the Institute of Management Berlin at the
Berlin School of Economics (FHW Berlin)
Badensche Str. 50-51, D-10825 Berlin

Editors:
Prof. Dr. Gert Bruche
Prof. Dr. Hansjörg Herr
Prof. Dr. Friedrich Nagel
Prof. Dr. Sven Ripsas

ISSN 1436 3151

Biographic note:

Prof. Dr. sc. techn. Dr.-Ing. Helmut Jarosch ist Professor für betriebswirtschaftliche Informations- und Kommunikationssysteme an der Fachhochschule für Wirtschaft Berlin (Berlin School of Economics). Nach einem Physik-Studium an der TU Dresden widmete er sich zunächst an der Akademie der Wissenschaften der Forschung und Entwicklung auf den Gebieten des Information Retrieval und der logischen Programmierung. Nach Promotion und Habilitation an der TU Ilmenau war er als wissenschaftlicher Berater bei der Thyssen Edelstahlwerke AG in Krefeld und als Leiter der Abteilung Software-Technik im Hahn-Meitner-Institut Berlin tätig. Seit 1992 führt er an der FHW Berlin Forschungsprojekte zu den Themen Datenbanken, Information Retrieval und Künstliche Intelligenz durch. Ein besonderer Schwerpunkte seiner Arbeit liegt in der interdisziplinären Verknüpfung traditioneller Verfahren der Datenbank-Technologie und des Information Retrieval mit modernen Methoden der Künstlichen Intelligenz.

Kontakt: Prof. Dr. Helmut Jarosch, Fachhochschule für Wirtschaft Berlin, Badensche Str. 50-51, 10825 Berlin.
Email: HelmutJarosch@fhw-berlin.de, Phone: +49(0)30 85789 126.

Abstract:

This paper presents a method aiming at the optimal cooperation of machine-made and human specialists in the course of a task performing process. The overall task is split up into subtasks. Each subtask is performed by a specialist that either may be an end user or may be realized by a software module. Such a specialist is installed and entrusted with the performance of a task delegated by another specialist, if it is best fitted to solve this task. The installed specialists are arranged in a communication system of agents interacting by solely exchanging messages. In order to ensure the minimal load of the computational resources, technological knowledge is used to select optimal specialists. This knowledge is represented in form of

- descriptions of the domains of competence of the specialists,
- instructions to simulate the task performing process,
- and metaalgorithms to generate “particularly tailored” algorithms.

Zusammenfassung:

Der vorliegende Beitrag beschreibt eine Methode, nach der im Verlauf eines Problemlösungs-Prozesses eine optimale Kooperation zwischen maschinellen und intellektuellen Spezialisten erreicht werden kann. Die Gesamtaufgabe wird in Teilaufgaben zerlegt. Jede dieser Teilaufgaben wird von einem Spezialisten gelöst, bei dem es sich entweder um einen Endbenutzer handeln kann oder der durch ein Software-Modul realisiert wird. Ein solcher Spezialist wird dann generiert und mit der Lösung einer von einem anderen Spezialisten delegierten Aufgabe betraut, wenn er am besten für die Lösung dieser Aufgabe geeignet ist. Die generierten Spezialisten werden in ein Kommunikationssystem von Agenten aufgenommen, die ausschließlich durch den Austausch von Nachrichten miteinander in Beziehung treten. Um eine minimale Ressourcen-Belastung zu erreichen, wird technologisches Wissen ausgewertet, um den optimalen Spezialisten auszuwählen. Dieses Wissen liegt vor in Form von

- Beschreibung der Kompetenzbereiche der Spezialisten,
- Anweisungen zur Simulation des Problemlösungs-Prozesses und
- Metaalgorithmen zur Erzeugung eines „maßgeschneiderten“ Algorithmus’.

Inhaltsverzeichnis

Symbolverzeichnis

$\mathbf{A}^{\lambda}$	Menge von Attributen für einen Spezialistentyp λ. Sie umfasst System-Attribute, Aufgaben-Attribute und Spezialistentyp-Attribute
$\underline{\mathrm{act}}(s)$	Empfängerangabe zum Aktivieren des Spezialisten s
$\underline{\mathrm{aret}}$	Empfängerangabe im Aktivator-Rückkehrpunkt
$\underline{\mathrm{aret}^{+}}$	Empfängerangabe im Endpunkt zur Weitergabe der Steuerung an den Aktivator
C^{λ}	Gesamtkosten, die ein Repräsentant des Spezialistentyps λ bei der Ausführung einer zu delegierenden Aufgabe voraussichtlich verursacht
d	Datenobjekt eines Spezialisten
d_{kom}	Datenobjekt mit Daten, die für die Benutzer-Kommunikation erforderlich sind
e	Empfängerangabe in einem Kommunikationspunkt
$\mathbf{G}_{\lambda}$	Grammatik zur Beschreibung des Spezialistentyps λ
$\underline{\mathrm{gen}}(\lambda^{*}, s)$	Empfängerangabe zum Generieren eines Spezialisten s des optimalen Spezialistentyps λ^{*}
$\underline{\mathrm{gret}^{+}}$	Empfängerangabe im Endpunkt zur Weitergabe der Steuerung an den Aktivator
K^{λ}	Kompetenzbereich des Spezialistentyps λ
M^{λ}	Metaalgorithmus, der den Algorithmus erzeugt, nach dem ein Repräsentant des Spezialistentyps λ arbeiten soll
n	Nachricht, die in einem Kommunikationspunkt gesendet wird
n_a	Nachricht, die in einem Aktivierungspunkt übergeben wird
n_g	Generierungsnachricht, die eine Formulierung der zu delegierenden Aufgabe enthält
$\mathbf{O}^{\lambda}$	Menge der Operationen, die ein Repräsentant des Spezialistentyps λ ausführen kann
o^{j}	Operation innerhalb einer Operationen-Sequenz
p_a	Aktivierungspunkt in einem Algorithmus
p_{ar}	Aktivator-Rückkehrpunkt in einem Algorithmus
p_g	Generierungspunkt in einem Algorithmus
p_{φ}	Kommunikationspunkt innerhalb des Algorithmus' φ
p_{φ}^{+}	Endpunkt des Algorithmus' φ
R^{λ}	Ressourcen-Belastung, die ein Spezialist des Spezialistentyps λ bei der Ausführung einer Operation $o \in \mathbf{O}^{\lambda}$ verursacht
r	Vektor, dessen Elemente die Ressourcen-Belastung bei der Ausführung einer Operation angeben
r_i	Element des Vektors r, das die spezielle Belastung der i-ten Ressource bei der Ausführung einer Operation angibt

$\mathbf{S}^t$	Menge der Spezialisten zum Zeitpunkt t
s	Spezialist
s_0	Spezialist als Manager, der die Startaufgabe des Informationsverarbeitungs-Prozesses stellt
s_a	Spezialist, der Aktivator eines anderen Spezialisten ist
s_g	Spezialist, der Generator eines anderen Spezialisten ist
t	Zeitpunkt
t_0	Anfangszeitpunkt
w	Wert eines Attributs
w_{seq}	Operationen-Sequenz
$\mathbf{Z}$	Menge der Zustandsvariablen zur Beschreibung des Bearbeitungsstandes eines Spezialisten
$\mathbf{Z}^0$	Anfangswerte der Zustandsvariablen
α^t	Aktivierungsrelation zum Zeitpunkt t
γ^t	Generierungsrelation zum Zeitpunkt t
$\kappa(r)$	Funktion zur Berechnung der Kosten, die die Ressourcen-Belastung r verursacht
Λ^t	Menge der zum Zeitpunkt t deklarierten Spezialistentypen
Λ_{komp}	Menge der Spezialistentypen, die für die Lösung einer zu delegierenden Aufgabe kompetent sind
λ	Spezialistentyp
λ^*	Spezialistentyp, der eine zu delegierende Aufgabe mit minimalen Kosten ausführen kann
ν	Name eines Attributs
ν_{seq}	Name des Attributs zur Beschreibung einer Operationen-Sequenz
ρ_i	Funktion zur Berechnung der Belastung der i-ten Ressource bei Ausführung einer Operation
φ	Algorithmus, nach dem ein Spezialist arbeitet
φ_{kom}	Algorithmus, nach dem die Mensch-Maschine-Kommunikation abläuft
Ω^t	System der Spezialisten zum Zeitpunkt t

1 Einführung

Wir betrachten den Informationsverarbeitungs-Prozess als einen Problemlösungs-Prozess, bei dem individuelle Kommunikations-Partner in einem arbeitsteiligen Zusammenspiel Teilaufgaben lösen. Jeder einzelne Kommunikations-Partner ist für die Lösung einer abgrenzbaren Aufgabenklasse fachlich kompetent und verfügt über die Fähigkeit, sich durch Kommunikation mit den anderen Partnern in den Problemlösungs-Prozess einzugliedern. Aus der Sicht des Gesamtprozesses ist ein einzelner Kommunikations-Partner ausschließlich durch sein Kommunikations-Verhalten bestimmt, wobei es auf dieser allgemeinen Betrachtungsebene nicht von Bedeutung ist, ob er nun als Software-Modul realisiert ist oder ob ein Benutzer am Bildschirm als intellektueller Spezialist fungiert. Eine wesentliche Aufgabe bei der Gestaltung der Technologie des Problemlösungs-Prozesses besteht in der optimalen Auswahl fachlich kompetenter Kommunikations-Partner und in der Organisation ihrer Kommunikation.

Wir stellen eine Methode vor, nach der der Problemlösungs-Prozess durch ein System von Spezialisten realisiert wird. Jeder Spezialist formuliert zunächst die Aufgabenklasse, für deren Lösung er kompetent ist, und gibt an, mit welchem Aufwand er jeweils eine Aufgabe bearbeiten kann. Im Problemlösungs-Prozess nutzen die Spezialisten des Systems ihre Fähigkeit zur Kommunikation, um Teilaufgaben zu formulieren, deren Bearbeitung sie einem anderen Spezialisten übertragen möchten. Das System betraut dann unter den verfügbaren Spezialisten denjenigen mit der Lösung der delegierten Aufgabe, der dazu fachlich kompetent ist und die Aufgabe mit minimalem Aufwand lösen kann.

Bei der vorgeschlagenen Art der Zusammenarbeit muss der auftraggebende Spezialist den beauftragten Spezialisten nicht mehr explizit angeben. Zur Zeit der Programmierung werden lediglich die allgemeinen Fähigkeiten zur Aufgabenlösung formuliert, während sich das konkrete System der Spezialisten mit seinen speziellen Auftraggeber-Auftragnehmer-Beziehungen erst im Verlauf der eigentlichen Problembearbeitung herausbildet. Damit lassen sich Informations-Technologien mit einem hohen Maß an Flexibilität gestalten.

2 Methode

Die vorgeschlagene Methode beruht auf der Vorstellung, dass ein Problem durch das organisierte Zusammenwirken von ***Spezialisten*** gelöst wird.[1] Jeder dieser Spezialisten ist Repräsentant eines ***Spezialistentyps***, der im wesentlichen durch folgende Merkmale charakterisiert wird:

- ***Kompetenzbereich***: Darunter verstehen wir eine Klasse von Aufgaben, für deren Lösung ein Spezialist dieses Typs geeignet ist.
- ***Ressourcen-Belastung***: Durch die Angaben zur Ressourcen-Belastung wird beschrieben, in welchem Maße ein Spezialist dieses Typs bei der Lösung einer Aufgabe aus dem Kompetenzbereich die rechentechnischen Ressourcen in Anspruch nimmt.
- ***Metaalgorithmus***: Der Metaalgorithmus hat die Aufgabe, einen speziell ausgestalteten Algorithmus für die Lösung einer konkreten Aufgabe aus dem Kompetenzbereich zu erzeugen. Nach diesem Algorithmus arbeitet dann der Spezialist, dem die Aufgabe übertragen wird. Handelt es sich bei dem Spezialisten um einen Benutzer am Bildschirm, so beschreibt der erzeugte Algorithmus lediglich die aufgabenspezifische Benutzer-Kommunikation.

Im Prozess der Problembearbeitung können mehrere Spezialisten als Repräsentanten ein und desselben Spezialistentyps eingerichtet werden. Diese Repräsentanten lösen im allgemeinen unterschiedliche Aufgaben aus demselben Kompetenzbereich.

Ein Spezialist wird als selbständige informationsverarbeitende Einheit eingerichtet, wenn eine Aufgabe delegiert werden soll, die in seinem Kompetenzbereich liegt, und wenn er sie mit den geringsten Kosten bearbeiten kann. Während seiner Lebenszeit tritt er mit anderen Spezialisten ausschließlich durch den Austausch von Nachrichten in Beziehung. Zur Speicherung von erworbenem Wissen verfügt der Spezialist über ein „Gedächtnis“, dessen Größe sich während seiner Lebenszeit ändern kann. Der Spezialist hört erst dann auf zu existieren, wenn er die Aufgabe vollständig gelöst hat. Ein solcher Spezialist, der nach einem nur ihm bekannten Algorithmus arbeitet, der ein individuelles Gedächtnis besitzt und der mit anderen Spezialisten kommuniziert, kann entweder ein Software-Modul (ein maschineller Spezialist) oder ein Benutzer am Bildschirm (ein intellektueller Spezialist) sein.

Die Tätigkeit eines Spezialisten besteht in der eigenverantwortlichen Lösung der ihm übertragenen Aufgabe. Diese Aufgabe muss er allerdings nicht unbedingt in einem Schritt lösen. Er kann jeweils ein Zwischenergebnis bereitstellen, seine Arbeit unterbrechen und die Aufgabenbearbeitung erst dann wieder aufnehmen, wenn er erneut aktiv wird. Dies ist die

[1] Vgl. beispielsweise Bentley/Appelt (1997) und Eberbach/Burgin (2007).

typische Verhaltensweise bei der intellektuellen Lösung von Aufgaben, der im Kontext automatisierter Prozesse das Konzept der Koroutinen entspricht.[2]

Ein Spezialist muss die ihm übertragene Aufgabe nicht unbedingt allein lösen: Er kann Teilaufgaben an andere Spezialisten delegieren. Zu diesem Zweck beschreibt er die zu delegierende Aufgabe und übergibt diese Beschreibung einer Systemkomponente, die wir als ***Dispatcher*** bezeichnen. Der Dispatcher richtet daraufhin einen Spezialisten ein, der in der Lage ist, die delegierte Aufgabe mit minimalen Kosten zu lösen.[3] Für die Auswahl des Spezialisten setzt er das ihm zur Verfügung stehende ***technologische Wissen*** ein. Die Repräsentation des technologischen Wissens setzt sich aus zwei Bestandteilen zusammen:

- aus den Beschreibungen aller ***Spezialistentypen*** und
- aus einer Menge von benannten Variablen, denen im Zuge der Problemlösung Werte zugewiesen werden und die wir als ***Attribute*** bezeichnen.

Die Attribute erhalten ihre Werte aus drei Quellen:

- Der Manager des Informationsverarbeitungs-Prozesses legt sogenannte ***System-Attribute*** fest, die für die Berechnung der Ressourcen-Belastung benötigt werden (beispielsweise sind das die spezifischen Kosten für die verbrauchte Prozessorzeit, die belegten Speicherbereiche, die Plattenzugriffe usw.).
- Aus der vom Spezialisten formulierten Beschreibung der zu delegierenden Aufgabe werden Fakten extrahiert, die als ***Aufgaben-Attribute*** gespeichert werden.
- Die konkreten Erfahrungen, die ein Repräsentant eines Spezialistentyps bei der Aufgabenbearbeitung sammelt, finden ihren Niederschlag in den sogenannten ***Spezialistentyp-Attributen*** (dazu gehören beispielsweise Messwerte über verbrauchte Ressourcen und wesentliche Dateneigenschaften).

Wird der Dispatchers beauftragt, einen Spezialisten für die Lösung einer zu delegierenden Aufgabe einzurichten, so laufen die folgenden Aktivitäten ab:

a) Durch Prüfen der Kompetenzbereiche, die in den Beschreibungen der einzelnen Spezialistentypen angegeben wurden, ermittelt der Dispatcher jene Spezialistentypen, die in der Lage sind, die zu delegierende Aufgabe zu lösen.

b) Sind für die Lösung der Aufgabe mehrere Spezialistentypen fachlich kompetent, muss derjenige Spezialistentyp ermittelt werden, der die Aufgabe mit minimalen Kosten lösen kann. Unter Kosten verstehen wir dabei ein Maß für die Abweichung der Ressourcen

[2] Vgl. beispielsweise de Moura/Rodriguez/Ierusalimschy (2004).
[3] Vgl. beispielhaft Westfechtel/Conradi (2001).

Belastungen vom Optimum.[4] Der Dispatcher errechnet für jeden in Frage kommenden Spezialistentyp die prognostizierten Kosten und findet so den optimalen Spezialistentyp.

c) Durch den Metaalgorithmus des optimalen Spezialistentyps werden Algorithmus und „Gedächtnis“ des einzurichtenden Spezialisten generiert.

d) Der Spezialist wird als ein neuer Repräsentant des optimalen Spezialistentyps in den Informationsverarbeitungs-Prozess eingeordnet und aktiviert.

Die beschriebene Methode bietet die Möglichkeit, Informationsverarbeitungs-Technologien mit einem hohen Maß an Flexibilität zu gestalten. Diese Flexibilität basiert hauptsächlich auf drei Mechanismen:

a) Da vom methodischen Ansatz her nicht unterschieden wird, ob ein Spezialist als ein maschineller oder als ein intellektueller Spezialist realisiert wird, lässt sich der Problemlösungs-Prozess schrittweise automatisieren. Ein Teil der Spezialisten kann zunächst für eine intellektuelle Aufgabenbearbeitung eingerichtet und erst später vollständig oder teilweise automatisiert werden. Dann muss dem System lediglich mitgeteilt werden, dass für die Aufgabe, die bisher vom Benutzer am Bildschirm gelöst wurde, nunmehr auch ein Software-Modul kompetent ist.

b) Das Arsenal der Spezialistentypen lässt sich jederzeit erweitern, wenn man neue Lösungsmethoden in das System aufnehmen möchte. Hat beispielsweise für die Lösung einer breiten Aufgabenklasse bisher nur ein einziger allgemeiner Spezialistentyp zur Verfügung gestanden, so können nun Spezialistentypen hinzugenommen werden, die in der Lage sind, spezielle Aufgaben der Klasse in effektiverer Weise zu lösen.

c) Die Spezialisten können während ihrer Lebenszeit das gesammelte technologische Wissen erweitern, indem sie Dateneigenschaften und Messwerte über die Belastung der Ressourcen in den Spezialistentyp-Attributen ablegen. Diese Attribute werden vom Dispatcher berücksichtigt, wenn wieder für eine zu delegierende Aufgabe ein Spezialist einzurichten ist.

Die erläuterte Methode soll an einem einfachen Beispiel dargestellt werden, das in Abbildung 1 veranschaulicht wird.

Der `Spezialist 1 vom Spezialistentyp A` möge in seinem Algorithmus einen Kommunikationspunkt p_1 erreicht haben, in dem er eine Teilaufgabe delegieren möchte. Er wendet sich mit der Aufgabenbeschreibung an den Dispatcher. Dieser gliedert zunächst die Attribute der Aufgabenbeschreibung in die Attributmenge des technologischen Wissens ein. Der Dispatcher wählt dann durch Auswertung des technologischen Wissens denjenigen Spe-

[4] Vgl. beispielsweise Conitzer/Sandholm (2002).

zialistentyp aus, der die zu delegierende Aufgabe mit minimalen Kosten lösen kann. Durch Auswertung der Kompetenzbereiche der Spezialistentypen möge er zunächst ermittelt haben, dass nicht nur der `Spezialistentyp B`, sondern auch der `Spezialistentyp C` für die Lösung der Aufgabe kompetent ist.

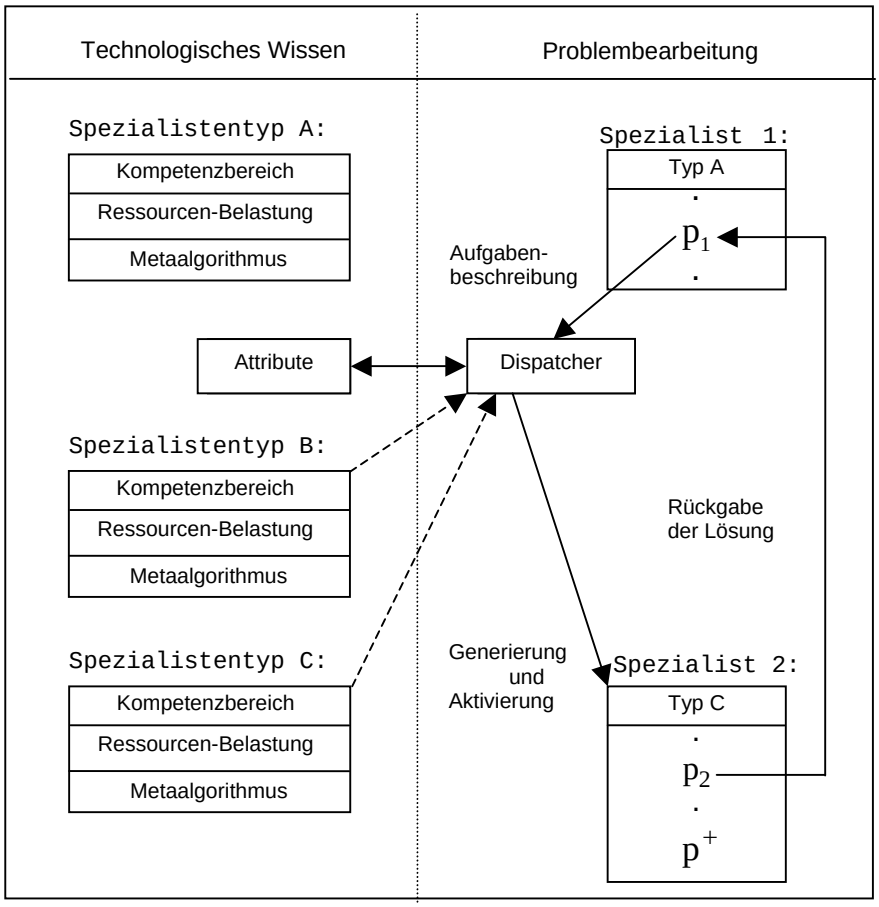

Abbildung 1: Auswahl eines Spezialisten für die Lösung einer zu delegierenden Aufgabe

Durch Auswertung der Angaben zur Ressourcen-Belastung, die für die beiden kompetenten Spezialistentypen hinterlegt sind, möge nun der Dispatcher feststellen, dass der `Spezialistentyp C` die zu delegierende Aufgabe mit geringeren Kosten lösen kann. Aus diesem Grunde wird vom Dispatcher der `Spezialist 2` vom `Spezialistentyp C` generiert. Der Dispatcher ordnet ihm einen Algorithmus zu, den der Metaalgorithmus dieses Typs erzeugt. Daraufhin wird der `Spezialist 2` aktiviert. Er erreicht schließlich in seinem Algorithmus den Kommunikationspunkt p_2. In diesem Kommunikationspunkt gibt er eine Teillösung der delegierten Aufgabe an `Spezialist 1` zurück. Der `Spezialist 1` wird erneut aktiviert und fährt in seinem Algorithmus fort. Der `Spezialist 2` wird passiv, sein „Gedächtnis“ bleibt aber erhalten. Er wartet solange, bis er von einem beliebigen Spezialisten wieder aktiviert wird und fährt dann in seinem Algorithmus fort. Irgendwann

erreicht der Algorithmus, nach dem der `Spezialist 2` arbeitet, seinen Endpunkt p^+. Nun wird der `Spezialist 2` einschließlich seines „Gedächtnisses“ gelöscht. Die Steuerung wird an `Spezialist 1` zurückgegeben, weil dieser die Generierung von `Spezialist 2` veranlasst hatte.

Diese bisher nur grob skizzierte Grundidee, die gesamte Problembearbeitung in die Tätigkeit kompetenter Spezialisten zu zerlegen und die Steuerung ihres Zusammenwirkens einem Dispatcher zu übertragen, wird nun im folgenden Abschnitt exakter dargestellt.

3 Modell

3.1 Spezialistentyp

Ein Spezialistentyp λ wird als ein Quintupel beschrieben:

$$\lambda = \left(\mathbf{O}^{\lambda}, K^{\lambda}, R^{\lambda}, M^{\lambda}, \mathbf{A}^{\lambda}\right) \tag{1}$$

In Anlehnung an das Konzept der abstrakten Datentypen[5] wird für einen Spezialistentyp λ die ***Menge der Operationen*** $\mathbf{O}^{\lambda}$ angegeben, die ein Repräsentant dieses Typs ausführen kann. Enthält die Menge $\mathbf{O}^{\lambda}$ nur eine einzige Operation, so beschreibt λ eine Prozedur im herkömmlichen Sinne.

Durch K^{λ} wird der ***Kompetenzbereich*** des Spezialistentyps beschrieben. K^{λ} ist eine logische Funktion, die gemäß

$$K^{\lambda}: \quad \mathbf{A}^{\lambda} \rightarrow \{\text{'kompetent'}, \text{'nicht kompetent'}\} \tag{2}$$

eine Abbildung aus der Menge der Attribute $\mathbf{A}^{\lambda}$ des Spezialistentyps auf die Werte 'kompetent' bzw. 'nicht kompetent' realisiert.

Die Komponente R^{λ} ist eine Funktion, die die ***Ressourcen-Belastung*** zum Ausdruck bringt, die mit der Ausführung einer Operation $o \in \mathbf{O}^{\lambda}$ verbunden ist. $R^{\lambda}(o)$ liefert einen Belastungs-Vektor $r = (r_1, r_2, \ldots, r_n)$, wobei die Komponente r_i die Belastung der i-ten Ressource bei Ausführung der Operation o darstellt. Ihr Wert hängt von der konkreten Aufgabe und vom aktuellen Arbeitszustand des Spezialisten ab.

Die Komponente M^{λ} ist ein ***Metaalgorithmus***. Dieser wertet die Attribute $\mathbf{A}^{\lambda}$ aus und liefert einen Algorithmus, nach dem ein Repräsentant des Spezialistentyps arbeiten soll.
Die Komponente $\mathbf{A}^{\lambda}$ ist eine Menge von ***Attributen***, die System-Attribute, Aufgaben-Attri-

[5] Vgl. beispielsweise Lipeck (1989).

bute und Spezialistentyp-Attribute umfasst. Diese Attribute werden zu folgenden Zwecken verwendet:

- Die Funktion K^{λ} leitet aus ihnen die Entscheidung ab, ob der Spezialistentyp λ für die Lösung einer zu delegierenden Aufgabe kompetent ist.
- Die Komponente R^{λ} nutzt sie dazu, die Ressourcen-Belastung bei der Ausführung der Operationen aus $\mathbf{O}^{\lambda}$ abzuschätzen.
- Der Metaalgorithmus M^{λ} generiert in Abhängigkeit von diesen Attributen einen „maßgeschneiderten“ Algorithmus für die Repräsentanten des Spezialistentyps.
- Sie dienen dazu, jene Erfahrungen zu fixieren, die die Repräsentanten des Spezialistentyps bei ihrer Aufgabenbearbeitung sammeln.

3.2 Technologisches Wissen

Das technologische Wissen ist eine zeitabhängige Datensammlung. Sie besteht zum Zeitpunkt t aus der Menge Λ^{t} der zu diesem Zeitpunkt deklarierten Spezialistentypen und aus der Menge $\mathbf{A}^{t}$ der zu diesem Zeitpunkt im System verwalteten Attribute.

Zum Zeitpunkt t_0, in dem der Problemlösungs-Prozess startet, besteht Λ^{t_0} aus der Menge der anfänglich deklarierten Spezialistentypen. $\mathbf{A}^{t_0}$ enthält alle Attribute der Spezialistentypen aus Λ^{t_0}:

$$\mathbf{A}^{t_0} = \bigcup_{\lambda \in \Lambda^{t_0}} \mathbf{A}^{\lambda} \tag{3}$$

Die in $\mathbf{A}^{t_0}$ enthaltenen System-Attribute werden vom Manager des Informationsverarbeitungs-Prozesses belegt. Die Aufgaben-Attribute haben keine Werte, da zu diesem Zeitpunkt noch keine Aufgaben delegiert wurden. Die Spezialistentyp-Attribute können mit Schätzwerten belegt sein.

Während des Informationsverarbeitungs-Prozesses tritt zu einem bestimmten Zeitpunkt t eine Veränderung des technologischen Wissens ein, die durch eines der folgenden Ereignisse ausgelöst wird (wir bezeichnen den Zeitpunkt vor dieser Änderung symbolisch mit „t-1“):

- Wird ein neuer Spezialistentyp λ deklariert, so tritt eine Erweiterung des technologischen Wissens ein:

$$\Lambda^t = \Lambda^{t-1} \cup \{\lambda\}; \quad \mathbf{A}^t = \mathbf{A}^{t-1} \cup \mathbf{A}^\lambda \tag{4}$$

- Wird ein Spezialistentyp λ aus dem System entfernt, so führt das zu einer Reduzierung des technologischen Wissens: aus $\mathbf{A}^{t-1}$ werden alle Attribute gestrichen, die ausschließlich in λ verwendet wurden:

$$\Lambda^t = \Lambda^{t-1} - \{\lambda\}; \quad \mathbf{A}^t = \bigcup_{\lambda \in \Lambda^t} \mathbf{A}^\lambda \tag{5}$$

- Durch ein privilegiertes Kommando des Managers können System-Attribute hinzugefügt oder geändert werden.
- Wenn ein Spezialist eine Aufgabe delegieren möchte, dann beschreibt er sie durch Aufgaben-Attribute. Diese werden in die Menge der Attribute aufgenommen.
- Ein Spezialist kann Spezialistentyp-Attribute durch neu ermittelte Messwerte und Dateneigenschaften belegen bzw. verändern.

3.3 System der Spezialisten

Im Verlauf des Problemlösungs-Prozesses werden Spezialisten als Repräsentanten von Spezialistentypen eingerichtet. Sie treten in Kommunikation mit anderen Spezialisten und hören schließlich auf zu existieren. Dieses Verhalten der Spezialisten beschreiben wir durch ein zeitabhängiges System.[6] Das System Ω^t besteht zum Zeitpunkt t aus einer Menge von Spezialisten $\mathbf{S}^t$, über der zwei Relationen γ^t und α^t definiert sind:

$$\Omega^t = \left(\mathbf{S}^t, \gamma^t, \alpha^t\right); \quad \gamma^t \subset \mathbf{S}^t \times \mathbf{S}^t; \quad \alpha^t \subset \mathbf{S}^t \times \mathbf{S}^t \tag{6}$$

Ein Spezialist $s \in \mathbf{S}^t$ ist gegeben durch einen Algorithmus φ, nach dem er arbeitet, und durch ein Datenobjekt d (sein individuelles „Gedächtnis“), das durch den Algorithmus verändert wird: $s = (\varphi, d)$. Wir abstrahieren in unserem Modell von der konkreten Ausprägung des Algorithmus φ. Als wesentlich betrachten wir lediglich, dass er eine Menge von Punkten enthält, in denen eine Nachricht an einen Empfänger abgeschickt und diesem gleichzeitig die Steuerung übergeben wird. Einen derartigen Punkt bezeichnen wir als einen ***Kommunikationspunkt*** p_φ:

$$\varphi = \left\{ p_{\varphi_1}, p_{\varphi_2}, ..., p_\varphi^+ \right\} \tag{7}$$

[6] Vgl. beispielsweise Jarosch (2007).

Jeder Algorithmus besitzt wenigstens einen Kommunikationspunkt p_φ^+, den wir als ***Endpunkt*** bezeichnen. In diesem Kommunikationspunkt endet die Aufgabenbearbeitung durch den Algorithmus φ. Die Steuerung muss dann an einen anderen Spezialisten übergeben werden. Das erfolgt gemeinsam mit dem Senden einer - eventuell leeren - Nachricht.

In jedem Kommunikationspunkt p_φ eines Algorithmus φ wird eine Nachricht n an einen Empfänger abgeschickt, der durch die Empfängerangabe e gekennzeichnet ist:

$$p_\varphi = (n, e) \in \varphi \tag{8}$$

Die Nachricht n informiert den Empfänger über den Stand der Abarbeitung des Algorithmus φ, der im Kommunikationspunkt p_φ erreicht wurde.

Die Spezialisten des Systems Ω^t treten in den Kommunikationspunkten ihrer Algorithmen miteinander in Beziehung und ordnen sich dabei in die Relationen γ^t und α^t ein.

Ein Element $(s_g, s) \in \gamma^t$ bringt zum Ausdruck, dass der Spezialist s_g den Spezialisten s eingerichtet hat. Wir bezeichnen s_g als den ***Generator*** von s. Bei der Beschreibung des Zeitverhaltens des Systems Ω^t spielt diese Relation dann eine Rolle, wenn der Spezialist s aufhört zu existieren, wenn dessen Algorithmus also seinen Endpunkt erreicht hat.

Ein Element $(s_a, s) \in \alpha^t$ bringt zum Ausdruck, dass der Spezialist s_a den Spezialisten s mit dem Zusenden einer Nachricht zur Fortsetzung seiner Arbeit aufgefordert, ihn also wieder aktiviert hat. Wir bezeichnen s_a als ***Aktivator*** von s. Bei der Beschreibung des Zeitverhaltens des Systems Ω^t spielt diese Relation dann eine Rolle, wenn s in einem Kommunikationspunkt eine Nachricht an seinen Aktivator s_a zurückschicken möchte.

Zum Zeitpunkt t_0, an dem der Problemlösungs-Prozess startet, hat das System der Spezialisten die Form

$$\Omega^{t_0} = (\{s_0\}, \varnothing, \varnothing) \tag{9}$$

Der Spezialist s_0 ist der Manager, der die Startaufgabe des Informationsverarbeitungs-Prozesses stellt. Da er zu diesem Zeitpunkt der einzige Spezialist ist, sind die Relationen γ^{t_0} und α^{t_0} leer.

3.4 Kommunikation zwischen den Spezialisten

Das Zeitverhalten des Systems der Spezialisten wird durch die Aktivitäten bestimmt, die in den Kommunikationspunkten der Algorithmen ablaufen. Bei der Darlegung dieser Aktivitäten werden wir folgende Arten von Kommunikationspunkten unterscheiden:

- Kommunikationspunkte, in denen eine Aufgabe delegiert und ein Spezialist zu ihrer Bearbeitung ausgewählt, eingerichtet und aktiviert wird (***Generierungspunkte***),
- Kommunikationspunkte, in denen ein bereits existierender Spezialist zur Fortsetzung seiner Arbeit aufgefordert wird (***Aktivierungspunkte***),
- Kommunikationspunkte, in denen ein Spezialist aus dem System ausscheidet (***Endpunkte***).

3.4.1 Generierungspunkt

Das Erweitern des Systems Ω^{t-1} zum Zeitpunkt t erfolgt dadurch, dass ein neuer Spezialist seine Tätigkeit im Rahmen der Problembearbeitung aufnimmt. Diesen Vorgang bezeichnen wir als ***Generierung***. Wird von einem Spezialisten s_g, der nach dem Algorithmus φ_g arbeitet, eine Aufgabe an einen neu einzurichtenden Spezialisten s vom Spezialistentyp λ^* delegiert, so muss der Algorithmus φ_g einen Kommunikationspunkt enthalten, den wir als ***Generierungspunkt***

$$p_g = \left(n_g, \underline{gen}(\lambda^*, s)\right) \in \varphi_g \tag{10}$$

bezeichnen. Die Nachricht n_g enthält die Formulierung der zu delegierenden Aufgabe in der Syntax

$$n_g ::= \nu = w \{, \nu = w\} ... \tag{11}$$

Dabei ist ν jeweils der Name eines Attributs und w der dazugehörige Wert. Das Absenden dieser Nachricht löst die folgenden Aktivitäten aus:

- Aufnahme der Aufgaben-Attribute in die Attributmenge $\mathbf{A}^t$;
- Ermittlung derjenigen Spezialistentypen, die für die Lösung der zu delegierenden Aufgabe kompetent sind;
- Ermittlung des optimalen Spezialistentyps λ^*;
- Generieren des Spezialisten s;

- Aufnahme des Spezialisten s in das System der Spezialisten Λ^t;
- Aktivieren des Spezialisten s.

Die genannten Aktivitäten werden in den folgenden Abschnitten der Reihe nach beschrieben.

3.4.1.1 Übernahme der Aufgaben-Attribute

Die in der Generierungsnachricht n_g enthaltenen Attribute bilden die Menge $\mathbf{A}_g = \{(\nu_1 = w_1), (\nu_2 = w_2), \ldots, (\nu_k = w_k)\}$, die in die Attributmenge $\mathbf{A}^{t-1}$ „eingemischt" wird: Enthält $\mathbf{A}_g$ einen neuen Wert w für ein Attribut ν, so wird in $\mathbf{A}^{t-1}$ der alte Wert w' durch den neuen Wert ersetzt:

$$\mathbf{A}^t = \mathbf{A}^{t-1} - \left\{ (\nu = w') \;\middle|\; \begin{matrix} (\nu = w') \in \mathbf{A}^{t-1} \wedge \\ (\nu = w) \in \mathbf{A}_g \end{matrix} \right\} \cup \mathbf{A}_g \qquad (12)$$

3.4.1.2 Ermittlung der kompetenten Spezialistentypen

Für jeden Spezialistentyp $\lambda \in \Lambda^t$ wird die Funktion K^λ unter Berücksichtigung der aktuellen Attributwerte berechnet. Die Menge Λ_{komp} der kompetenten Spezialistentypen ergibt sich gemäß

$$\Lambda_{komp} = \left\{ \lambda \;\middle|\; \lambda \in \Lambda^t,\ K^\lambda\left(\mathbf{A}^t\right) = \text{'kompetent'} \right\} \qquad (13)$$

Dabei sind drei Fälle zu unterscheiden:

$|\Lambda_{komp}| = 0$: Für die zu delegierende Aufgabe gibt es keinen kompetenten Spezialistentyp. Der Problemlösungs-Prozess verharrt solange, bis dieser Mangel behoben wurde.

$|\Lambda_{komp}| = 1$: Da es nur einen einzigen kompetenten Spezialistentyp gibt, ist er der gesuchte Spezialistentyp.

$|\Lambda_{komp}| > 1$: Da es mehrere kompetente Spezialistentypen gibt, muss der optimale Spezialistentyp ermittelt werden.

3.4.1.3 Ermittlung des optimalen Spezialistentyps

Aus der Menge Λ_{komp} der kompetenten Spezialistentypen muss nun derjenige Spezialistentyp ermittelt werden, der in der Lage ist, die zu delegierende Aufgabe mit minimalen Kosten zu lösen. Den Ausgangspunkt für die Kostenberechnung bildet der Vektor

$r = (r_1, r_2, ..., r_n)$, dessen Element r_i jeweils die Belastung der i-ten Ressource bei der Ausführung einer Operation $o \in \mathbf{O}^{\lambda}$ des Spezialistentyps $\lambda \in \Lambda_{komp}$ angibt. Ehe ein Spezialist vom Typ λ die Operation o ausführt, hat er im allgemeinen bereits andere Operationen aus der Menge $\mathbf{O}^{\lambda}$ ausgeführt, die ihn in einen konkreten Bearbeitungszustand versetzt haben. Dieser Zustand wird durch Zustandsvariable beschrieben, die in ihrer Gesamtheit die Menge $\mathbf{Z}$ bilden. (Eine solche Zustandsvariable ist im nachfolgend behandelten Beispiel des Wörterbuchs die Anzahl der zum jeweiligen Zeitpunkt gespeicherten Wörter).

Die Belastung der i-ten Ressource bei Ausführung der Operation $o \in \mathbf{O}^{\lambda}$ hängt von den aktuellen Werten der Zustandsvariablen $\mathbf{Z}$ und von Werten aus der Attributmenge $\mathbf{A}^{\lambda}$ (insbesondere von Werten der Aufgaben-Attribute und der Spezialistentyp-Attribute) ab: $r = r\left(o, \mathbf{Z}, \mathbf{A}^{\lambda}\right)$. (Im Beispiel des Wörterbuchs fungiert als ein Aufgaben-Attribut die maximale Anzahl der zu speichernden Wörter und als ein Spezialistentyp-Attribut die für den Vergleich zweier Wörter benötigte Prozessorzeit).

Um also die Ressourcen-Belastung berechnen zu können, die durch die Ausführung einer Operation verursacht wird, muss dem Dispatcher bekannt sein, welche Operationen des Spezialistentyps in welcher Reihenfolge ausgeführt werden sollen. Die Beschreibung der auszuführenden ***Operationen-Sequenz*** ist deshalb ein wesentlicher Bestandteil der Aufgabenbeschreibung. Diese Angabe kann natürlich nur den Charakter einer „Absichtserklärung“ haben, ist also nur eine Schätzung. In der Aufgabenbeschreibung wird die Operationen-Sequenz durch ein Attribut (ν_{seq}, w_{seq}) angegeben, wobei w_{seq} die Form

$$w_{seq} = \left(o^1, o^2, \ldots, o^p\right); \quad o^j \in \mathbf{O}^{\lambda}; \quad j = 1, 2, \ldots, p \tag{14}$$

hat. Durch Auswertung der Operationensequenz w_{seq} lässt sich das Verhalten eines Repräsentanten vom Spezialistentyp $\lambda \in \Lambda_{komp}$ hinsichtlich der von ihm verursachten Ressourcen-Belastung simulieren. Diese Simulation führt der Dispatcher unter Verwendung der Funktion R^{λ} durch. Die Simulation läuft in folgender Schrittfolge ab:

1. Die Zustandsvariablen der Menge $\mathbf{Z}$ werden entsprechend den Angaben in R^{λ} auf ihre Anfangswerte gesetzt: Es entsteht der Initialzustand $\mathbf{Z}^0$.

2. Für jede Operation $o^j; \quad j = 1, 2, \ldots, p$ aus w_{seq} wird der Belastungsvektor $r = r\left(o^j, \mathbf{Z}^{j-1}, \mathbf{A}^{\lambda}\right)$ berechnet. Er hängt sowohl vom Zustand $\mathbf{Z}^{j-1}$ ab, der im vorangegangenen Simulationsschritt erreicht wurde, als auch von Werten aus der

Attributmenge $\mathbf{A}^{\lambda}$. Diese Berechnung erfolgt durch Auswertung von Funktionen ρ_i, die angeben, wie stark die i-te Ressource bei Ausführung der Operation o^j belastet wird. Der Dispatcher wendet eine Kostenfunktion κ auf den Belastungsvektor r an und gewinnt die Kosten, die durch die Operation o^j verursacht werden. Dem Dispatcher stehen mehrere Kostenfunktionen zur Verfügung. Eine Kostenfunktion kann zum Beispiel darauf abzielen, eine Minimierung des Speicherbedarfs herbeizuführen. Eine andere Kostenfunktion kann darauf abzielen, die Aufgabe in minimaler Zeit zu beenden. Der Spezialist, der eine Aufgabe delegiert, kann durch ein Aufgaben-Attribut den Dispatcher anweisen, eine bestimmte Kostenfunktion zu verwenden. Standardmäßig wird eine Kostenfunktion verwendet, die eine ausgewogene Lastverteilung über sämtliche Ressourcen herbeiführt. Vor dem Beginn des nächsten Simulationsschrittes werden die Zustandsvariablen verändert, so dass der neue Zustand $\mathbf{Z}^j$ eintritt. Dazu enthält R^{λ} für jede Operation $o \in \mathbf{O}^{\lambda}$ die entsprechenden Anweisungen.

3. Durch Aufsummierungen der Kostenanteile $\kappa(r)$ über alle Operationen der Operationen-Sequenz w_{seq} werden die Gesamtkosten C^{λ} ermittelt, die ein Repräsentant des Spezialistentyps λ voraussichtlich verursacht:

$$C^{\lambda} = \sum_{j=1}^{p} \kappa\left(r\left(o^j, \mathbf{Z}^{j-1}, \mathbf{A}^{\lambda}\right)\right) \tag{15}$$

Nachdem für jeden kompetenten Spezialistentyp $\lambda \in \Lambda_{komp}$ eine solche Simulation durchgeführt wurde, liegen die Schätzwerte dafür vor, mit welchen Kosten der jeweilige Spezialistentyp die zu delegierende Aufgabe lösen würde. Nun kann der optimale Spezialistentyp λ^* ausgewählt werden, der voraussichtlich die minimalen Kosten verursacht:

$$\lambda^* = \arg \min_{\lambda \in \Lambda_{komp}} \left(C^{\lambda}\right) \tag{16}$$

3.4.1.4 Generieren des Spezialisten

Nachdem nun feststeht, dass der Spezialistentyp λ^* am besten dafür geeignet ist, die durch die Generierungsnachricht n_g formulierte Aufgabe zu lösen, wird ein Repräsentant s dieses Typs generiert. Durch Auswertung des Metaalgorithmus M^{λ^*} werden der Algorithmus φ und das Datenobjekt d_{φ} des Repräsentanten erzeugt.

Handelt es sich bei den Repräsentanten von λ^* um maschinelle Spezialisten, so ist φ das Programm, das unter Berücksichtigung der aktuellen Werte der Attribute $\mathbf{A}^{\lambda^*}$ durch den Metaalgorithmus M^{λ^*} erzeugt wird. Das Datenobjekt d_φ ist dabei ebenfalls durch M^{λ^*} bestimmt.

Sind die Repräsentanten von λ^* dagegen intellektuelle Spezialisten, so ist der Algorithmus φ_{kom} lediglich der Algorithmus, nach dem die Mensch-Maschine-Kommunikation abläuft. Das Datenobjekt d_{kom} beinhaltet dann die für die Abwicklung der Kommunikation erforderlichen Daten:

$$s = \begin{cases} (\varphi, d_\varphi),\ \varphi = M^{\lambda^*}\left(\mathbf{A}^{\lambda^*}\right) & \text{Der Repräsentant von } \lambda^* \text{ ist ein maschineller Spezialist} \\ (\varphi_{kom}, d_{kom}) & \text{Der Repräsentant von } \lambda^* \text{ ist ein intellektueller Spezialist} \end{cases} \quad (17)$$

3.4.1.5 Aufnahme des Spezialisten in das System der Spezialisten

Der neu hinzugekommene Spezialist s wird in das System der Spezialisten Ω^{t-1} aufgenommen. Dabei werden die beiden Relationen γ^{t-1} und α^{t-1} um das Element (s_g, s) erweitert:

$$\Omega^t = \left(S^{t-1} \cup \{s\}, \gamma^{t-1} \cup \{(s_g, s)\}, \alpha^{t-1} \cup \{(s_g, s)\}\right) \quad (18)$$

Dadurch wird festgehalten, dass der Spezialist s_g, der die Aufgabe delegieren möchte, sowohl der Generator als auch der Aktivator des neu eingerichteten Spezialisten s sein wird. Der Dispatcher teilt s_g gemäß (10) mit, dass er die zu delegierende Aufgabe dem Spezialisten s vom Spezialistentyp λ^* übertragen wird.

3.4.1.6 Aktivieren des Spezialisten

Der Generator s_g wird passiv und verharrt im Generierungspunkt p_g. Dem Spezialisten s wird die Steuerung übergeben. Er bleibt solange aktiv, bis er seinen ersten Kommunikationspunkt erreicht. In diesem Punkt übergibt er die Steuerung (gemeinsam mit einer Nachricht) an einen anderen Spezialisten und wird wieder passiv. Der Generator s_g verlässt den Generierungspunkt erst dann, wenn er wieder aktiviert wird. Welcher Spezialist ihn zur Fortsetzung der Arbeit auffordert, ist in unserem Modell nicht vorherbestimmt, sondern hängt vollkommen vom Gang der Problembearbeitung ab.

3.4.2 Aktivierungspunkt

Zu jedem beliebigen Zeitpunkt t ist jeweils ein Spezialist des Systems Ω^t aktiv, während alle anderen passiv sind und in ihrem jeweiligen Bearbeitungszustand verharren. Der aktive Spezialist s_a, der nach dem Algorithmus φ arbeitet, kann in einem Kommunikationspunkt seine Arbeit unterbrechen und die Steuerung gemeinsam mit einer Nachricht s_a an einen beliebigen anderen Spezialisten s des Systems Ω^t übergeben. Eine solche Steuerungsübergabe bezeichnen wir als ***Aktivierung***. Der Sendepunkt, in dem das geschieht, wird ***Aktivierungspunkt*** genannt:

$$p_a = (n_a, \underline{act}(s)) \in \varphi \tag{19}$$

Wenn s_a einen Spezialisten s wieder aktivieren möchte, muss dieser ihm bekannt sein. Ist s der Generator von s_a, so ist er ihm gemäß (10) im Generierungspunkt mitgeteilt worden. Andernfalls muss der „Adressat“ dem Aktivator im Zuge der Kommunikation mit anderen Spezialisten bekanntgegeben worden sein.

Sendet der Spezialist s_a im Aktivierungspunkt p_a eine Nachricht n_a an den Spezialisten s, so werden folgende Aktivitäten des Dispatchers ausgelöst:

a) Die Relation α^{t-1} wird aktualisiert. Enthält sie bisher ein Element $(x, s) \in \alpha^{t-1}$, so wird dieses aus der Relation entfernt: Damit wird die „Erinnerung“ an eine frühere Aktivierung von s gelöscht. Das Element (s_a, s) wird zur Relation α^{t-1} hinzugefügt:

$$\alpha^t = \alpha^{t-1} - \left\{(x, s) \,\middle|\, (x, s) \in \alpha^{t-1}\right\} \cup \{(s_a, s)\} \tag{20}$$

   Durch dieses Vorgehensweise wird gesichert, dass die Relation α^t zu jedem Zeitpunkt genau einen Aktivator von s ausweist. Der Aktivator s_a wird passiv und verharrt im Aktivierungspunkt p_a.

b) Gemeinsam mit der Nachricht n_a wird dem Spezialisten s die Steuerung übergeben. Der Spezialist s setzt seine Arbeit an der Stelle fort, an der er sie zuletzt unterbrochen hat. Er bleibt solange aktiv, bis er seinen nächsten Kommunikationspunkt erreicht.

Ein Spezialfall der Aktivierung eines Spezialisten s_a durch einen Spezialisten s legt dann vor, wenn s_a derjenige Spezialist ist, der s das letztemal aktiviert hat. Das entspricht der Situation, dass s im Rahmen der delegierten Aufgabe eine Teilaufgabe zur Bearbeitung erhalten hat und nun deren Lösung zurückmelden möchte. Häufig sind solche Teilaufgaben so beschaffen, dass zu ihrer Lösung die Kenntnis des Auftraggebers nicht erforderlich ist.

Um auch dann noch ein Zurückmelden der Lösung zu gewährleisten, wurde die Relation α^t in das System Ω^t aufgenommen.

Einen derartigen Kommunikationspunkt, in dem der nach dem Algorithmus φ arbeitende Spezialist s die Lösung der ihm gestellten Teilaufgabe in Form der Nachricht n an seinen Aktivator zurücksenden möchte, bezeichnen wir als ***Aktivator-Rückkehrpunkt***:

$$p_{ar} = (n, \underline{aret}) \in \varphi \tag{21}$$

In einem solchen Punkt werden vom Dispatcher folgende Aktivitäten ausgeführt:

a) Der Aktivator s_a wird auf Grund des Elements $(s_a, s) \in \alpha^{t-1}$ ermittelt und dieses Element aus der Relation α^{t-1} entfernt:

$$\alpha^t = \alpha^{t-1} - \{(s_a, s)\} \tag{22}$$

b) Gemeinsam mit der Nachricht n wird die Steuerung dem Spezialisten s_a übergeben.

3.4.3 Endpunkt

Erreicht ein Spezialist s, der nach dem Algorithmus φ arbeitet, seinen Endpunkt p^+, so hat er die delegierte Aufgabe vollständig gelöst und kann aus dem System Ω^t entfernt werden. Da er im Besitz der Steuerung ist, muss er diese gemeinsam mit einer - eventuell leeren - Nachricht n an einen anderen Spezialisten s' übergeben. Dazu kann s den Spezialisten s' explizit angeben. Der Endpunkt hat dann die Form:

$$p^+ = (n, \underline{act}(s')) \in \varphi \tag{23}$$

Der Spezialist s kann die Steuerung aber auch an einen Spezialisten übergeben, zu dem er in einer besonderen Relation steht - nämlich an seinen Generator oder an seinen Aktivator. Wir schreiben das in der Form

$$p^+ = \begin{cases} \left(n, \underline{gret^+}\right) & \text{Übergabe an den Generator} \\ \left(n, \underline{aret^+}\right) & \text{Übergabe an den Aktivator} \end{cases} \tag{24}$$

Im Endpunkt p^+ führt der Dispatcher folgende Aktivitäten aus:

a) Der Spezialist s wird aus der Menge der Spezialisten des Systems Ω^{t-1} entfernt:

$$\mathbf{S}^t = \mathbf{S}^{t-1} - \{s\} \tag{25}$$

b) Soll die Steuerung an den Generator übergeben werden, so wird der Generator $s' = s_g$

auf Grund des Elements $(s_g, s) \in \gamma^{t-1}$ ermittelt. Soll dagegen der Aktivator $s' = s_a$ die Steuerung erhalten, so wird dieser auf Grund des Elements $(s_a, s) \in \alpha^{t-1}$ bestimmt. Ansonsten hat s den Spezialisten s', dem die Steuerung zu übergeben ist, explizit festgelegt.

c) Aus den Relationen γ^{t-1} und α^{t-1} werden alle Hinweise auf die Beziehungen gelöscht, die der Spezialist s zu anderen Spezialisten einging:

$$\gamma^t = \gamma^{t-1} - \{(s, x) \mid (s, x) \in \gamma^{t-1}\} - \{(s_g, s)\}$$

$$\alpha^t = \alpha^{t-1} - \{(s, x) \mid (s, x) \in \alpha^{t-1}\} - \{(s_a, s)\} \quad (26)$$

d) Die Steuerung wird gemeinsam mit der Nachricht n dem Spezialisten s' übergeben.

3.5 Kommunikation zwischen den Spezialisten

Die im Abschnitt 3.4 beschriebenen Kommunikationspunkte dienen dazu, die Kommunikation der Spezialisten untereinander abzuwickeln. Der Dispatcher hat dabei die Aufgabe, diese Kommunikation zu lenken und zu unterstützen. Dazu nutzt er einerseits das technologische Wissen und andererseits seine Kenntnis vom jeweiligen Zustand des Systems der Spezialisten. Auf diese Wissenskomponenten können die Spezialisten jedoch auch außerhalb der Kommunikationspunkte - durch Vermittlung des Dispatchers - zugreifen.

Ein Spezialist kann das technologische Wissen nur hinsichtlich der Spezialistentyp-Attribute beeinflussen. Diese Attribute haben die besondere Funktion, Messwerte über verbrauchte Ressourcen und über wesentliche Dateneigenschaften festzuhalten. Die Werte, die diesen Attributen in den Spezialistentyp-Beschreibungen zugeordnet wurden, sind lediglich Schätzwerte, die erst während der Problembearbeitung durch die Repräsentanten des Spezialistentyps an die tatsächlichen Verhältnissen angepasst werden müssen. Die Spezialistentyp-Attribute stellen damit ein Wissen dar, das über die Lebenszeit der einzelnen Repräsentanten hinaus als Wissen des Spezialistentyps kumuliert wird. Der Dispatcher stellt daher einfache Lernalgorithmen[7] zur Adaption der Spezialistentyp-Attribute bereit.

Beispielsweise kann der Dispatcher die Bildung eines arithmetischen Mittelwerts vornehmen. Hat ein Repräsentant des Spezialistentyps λ einen Wert w^* zum Attribut $(\nu, w_n) \in \mathbf{A}^{\lambda}$ gemessen, so kann er in Kommunikation mit dem Dispatcher treten und diesem (ν, w^*) als neuen Messwert mitteilen. Der Repräsentant kann dabei nicht wissen, wie viele Adaptions-

[7] Vgl. beispielsweise Stone/Veloso (2000).

schritte n bereits zur Bildung des bisherigen Mittelwertes w_n stattgefunden haben. Dieses Wissen verwaltet der Dispatcher und korrigiert den Mittelwert gemäß

$$w_{n+1} = \frac{n * w_n + w^*}{n+1} \tag{27}$$

Entsprechend dem jeweiligen Charakter der einzelnen Spezialistentyp-Attribute werden weitere einfache Adaptions-Algorithmen angewendet.

Die laufende Korrektur der Spezialistentyp-Attribute bewirkt, dass der Dispatcher bei der Auswahl des optimalen Spezialistentyps zur Bearbeitung einer zu delegierenden Aufgabe stets „praxisnah“ handeln kann.

In seiner Kommunikation mit dem Dispatcher kann ein Spezialist auch Auskünfte über das System der Spezialisten Ω^t einholen. Er kann beispielsweise die Frage stellen, ob ein ihm bekannter Spezialist s noch existiert und somit noch aktivierbar ist. Der Dispatcher prüft dann, ob $s \in \mathbf{S}^t$ gilt, und gibt das Ergebnis als Antwort zurück.

Nach dem bisher beschriebenen Zeitverhalten des Systems Ω^t wurde ein Spezialist $s = (\varphi, d)$ erst dann aus dem System entfernt, wenn der Algorithmus $\varphi = (p_1, p_2, \ldots, p^+)$ seinen Endpunkt p^+ erreicht hat. Nun sind aber Spezialisten, die zur Ausführung von Dienstleistungen für viele Auftraggeber eingerichtet wurden, häufig so gestaltet, dass sie ständig dienstbereit sind und von sich aus nie ihren Endpunkt erreichen. Arbeiten andererseits mehrere Spezialisten als Koroutinen zusammen, so endet die kollektive Problembearbeitung typischerweise dann, wenn eine der Koroutinen ihren Endpunkt erreicht hat. In all diesen Situationen besteht die Notwendigkeit, dass ein aktiver Spezialist einen passiven Spezialisten s zwangsweise löscht. Mit diesem Auftrag kann er sich an den Dispatcher wenden. Dieser führt dann folgende Handlungen aus:

a) Der gelöschte Spezialist s wird gemäß (25) aus der Menge der Spezialisten des Systems Ω^{t-1} entfernt.

b) Aus den Relationen γ^{t-1} und α^{t-1} werden gemäß (26) alle Hinweise auf die Beziehungen gelöscht, die s zu anderen Spezialisten einging.

c) Der Spezialist, der den Löschauftrag gestellt hat, erhält wieder die Steuerung zurück.

4 Realisierung des Modells

Im Abschnitt 2 wurde eine Methode vorgestellt, nach der für die Lösung einer zu delegierenden Aufgabe der optimale Spezialistentyp ausgewählt wird. Im Abschnitt 3 wurde das zugehörige mathematische Modell entwickelt. Im weiteren soll nun an Hand eines einfachen Beispiels gezeigt werden, wie dieses Modell realisiert werden kann.

4.1 Syntax für die Deklaration eines Spezialistentyps

Zunächst muss eine Syntax festgelegt werden, nach der ein Spezialistentyp λ deklariert wird. Wir schlagen eine Syntax vor, die durch die folgende Grammatik G_λ beschrieben wird:

Grammatik G_λ:

$$
\begin{aligned}
&< \text{Spezialistentyp} > ::= < \text{Typart} > < \text{Typname} > ; \\
&\qquad < \text{Kompetenzbereich} > \\
&\qquad [< \text{Spezialistentyp} - \text{Attribute} >] \\
&\qquad < \text{Ressourcen} - \text{Belastung} > \\
&\qquad \text{end} < \text{Typname} > ;
\end{aligned}
$$

Die <Typart> gibt an, ob die Repräsentanten des Spezialistentyps als maschinelle oder als intellektuelle Spezialisten einzurichten sind:

$$< \text{Typart} > ::= \text{machine} \mid \text{human}$$

Als <Typname> wird ein Bezeichner angegeben. Im Falle eines maschinellen Spezialisten benennt er zugleich den Metaalgorithmus M^λ, der den Algorithmus des einzurichtenden Repräsentanten erzeugt.

Der <Kompetenzbereich> gibt die Kompetenzbedingungen K^λ in Form eines logischen Ausdrucks an:

$$< \text{Kompetenzbereich} > ::= \text{authority} < \text{logischer Ausdruck} > ;$$

Der logische Ausdruck enthält Namen von logischen Attributen und/oder Vergleichsausdrücke, die gemäß den Regeln der Boole'schen Algebra miteinander verknüpft sind, z.B.:

$$\text{authority task} = \text{"sort"} \text{ and } ((\text{number} > 5000) \text{ or strings}) ;$$

Die Vergleichsausdrücke können Konstanten und/oder Namen von Attributen enthalten. Wenn der logische Ausdruck durch den Dispatcher ausgewertet wird, werden die Namen der Attribute durch ihre aktuellen Werte substituiert.

Die <Spezialistentyp-Attribute> sind fakultativ. Sie können wie folgt angegeben werden:

< Spezialistentyp – Attribute > ::=
attributes < Attributname > = < Attributwert > ;
{ < Attributname > = < Attributwert > ; }

Als <Ressourcen-Belastung> werden sämtliche Informationen angegeben, die der Dispatcher zur Simulation des Informationsverarbeitungs-Prozesses benötigt:

< Ressourcen – Belastung > ::=
simulation [< Anfangszustand >] < Operationseffekte >

Der Anfangszustand $\mathbf{Z}^0$ wird durch die Aufzählung von Zustandsvariablen mit ihren Anfangswerten beschrieben:

< Anfangszustand > ::=
vars < Variablenname > = < Ausdruck > ;
{ < Variablenname > = < Ausdruck > ; }

In den Ausdrücken können neben Konstanten auch Namen von Attributen aus $\mathbf{A}^\lambda$ auftreten.

Als <Operationseffekte> wird für jede Operation $o \in \mathbf{O}^\lambda$ und jede Ressource i angegeben, wie die Komponente r_i des Belastungsvektors $r\left(o, \mathbf{Z}, \mathbf{A}^\lambda\right)$ zu berechnen ist. Aus Gründen der Zweckmäßigkeit erfolgt eine solche Angabe nur für Komponenten, deren Belastung ungleich Null ist. Darüber hinaus können den Zustandsvariablen veränderte Werte zugewiesen werden:

< Operationseffekte > ::=
operations < Operationsname > : < Anweisungsfolge >
{ < Operationsname > : < Anweisungsfolge > }

< Anweisungsfolge > ::= < Belastungsangaben >
< Zustandsänderung >

< Belastungsangaben > ::=
< Ressourcen – Name > = < Ausdruck > ;
{ < Ressourcen – Name > = < Ausdruck > ; }

< Zustandsänderung > ::=
< Variablen – Name > = < Ausdruck > ;
{ < Variablen – Name > = < Ausdruck > ; }

Sowohl bei den Belastungsangaben als auch bei der Zustandsänderung können in den Ausdrücken neben Konstanten auch Namen von Attributen aus $\mathbf{A}^{\lambda}$ sowie die Namen der Zustandsvariablen auftreten.

4.2 Beispiel für die Auswahl eines Spezialistentyps

Im folgenden Beispiel betrachten wir zwei unterschiedliche Spezialistentypen „sorted_area_dictionary“ und „binary_tree_dictionary“, die beide die Arbeit mit einem Wörterbuch unterstützen. Sie beschreiben beide einen abstrakten Datentyp „dictionary“, der durch die vier Operationen „add“ (Wort hinzufügen), „delete“ (Wort entfernen), „identify“ (Wort identifizieren) und „browse“ (alle Wörter in sortierter Folge bereitstellen) charakterisiert sein möge.

Der Spezialistentyp „sorted_area_dictionary“ arbeitet mit einem Feld, das von Anfang an so groß ausgelegt wird, dass es für die Speicherung aller zu verwaltenden Wörter ausreicht. Die Wörter sind zu jedem Zeitpunkt alphabetisch sortiert. Das Feld enthält (3 Byte lange) Verweise auf die Zeichenketten, die auf einem Heap abgespeichert werden. Das Hinzufügen und Streichen eines Wortes erfordert das Verschieben der nachfolgenden Verweise. Das Aufsuchen eines Wortes erfolgt durch binäre Suche.

Dieser Spezialistentyp kann in der vorgeschlagenen Syntax wie folgt beschrieben werden:

```
1  machine sorted_area_dictionary;
2
3   authority
4     task='dictionary';
5
6   attributes
7     string_compare_time =3.2;
8     string_shift_time =1.7;
9     pointer_shift_time  =1.7;
10
11  simulation
12    vars
13      area_size=word_max_number*3;
14      word_number=1;
```

```
15
16    operations
17     add:
18      space=area_size +
19           word_number*average_word_length; +
20      time=ld(word_number)*string_compare_time
21           word_number/2*pointer_shift_time +
22           string_shift_time +
23           pointer_shift_time;
24      word_number=word_number+1;
25
26     delete:
27      space=area_size +
28           word_number*average_word_length; +
29      time=ld(word_number)*string_compare_time
30           word_number/2*pointer_shift_time;
31      word_number=word_number-1;
32
33     identify:
34      space=area_size +
35           word_number*average_word_length;
36      time=ld(word_number)*string_compare_time;
37
38     browse:
39      space=area_size +
40           word_number*average_word_length;
41      time=word_number*string_shift_time;
42
43  end sorted_area_dictionary;
```

Da es sich hierbei um einen abstrakten Datentyp mit fest umrissenem Aufgabenspektrum handelt, kann der Kompetenzbereich einfach durch den Namen des abstrakten Datentyps angegeben werden.

Die Prozessorzeiten (in ms) für das Vergleichen bzw. Verschieben der Daten werden durch die drei Spezialistentyp-Attribute string_compare_time, string_shift_time und pointer_shift_time angegeben. Die Zustandsvariable area_size berechnet sich aus dem Aufgaben-Attribut word_max_number. Bei der Ressourcen-Belastung werden lediglich der

Speicherplatzbedarf und die Prozessorzeit ermittelt. Bei der Berechnung des Speicherplatzbedarfs findet das Aufgaben-Attribut average_word_length Berücksichtigung. Da die Prozessorzeit für fast alle Operationen durch Logarithmen-Funktionen berechnet wird, beginnen wir die Simulation im Zustand, dass im Wörterbuch bereits ein Wort vorliegt (word_number=1).

Der Spezialistentyp „binary_tree_dictionary“ arbeitet mit einem binären Baum, der im Zuge des Hinzufügens von Wörtern erweitert wird. Jeder Knoten enthält außer dem Verweis auf ein Wort auch noch zwei (3 Byte lange) Verweise auf die beiden Unterbäume. Beim Erweitern des Baums findet kein Ausbalancieren des Baums statt. Die Abweichung der tatsächlichen mittleren Baumtiefe von der optimalen Baumtiefe findet durch das Spezialistentyp-Attribut balance_factor Berücksichtigung. Das Löschen eines Wortes erfolgt nicht durch eine Umstrukturierung des Baums, sondern lediglich durch das Markieren des Knotens mit einem Löschflag. Dieser Spezialistentyp lässt sich wie folgt beschreiben:

```
1  machine binary_tree_dictionary;
2
3   authority
4     task=’dictionary’;
5
6   attributes
7     string_compare_time =3.2;
8     string_shift_time =1.7;
9     pointer_shift_time  =1.7;
10    flag_set_time     =1.7;
11    balance_factor    =1.5;
12
13  simulation
14    vars
15      word_number=1;
16      word_space=average_word_length+9;
17
18    operations
19     add:
20      space=word_number*word_space;
21      time=  balance_factor*ld(word_number)*
22                          string_compare_time +
23           string_shift_time +
```

```
24            pointer_shift_time;
25       word_number=word_number+1;
26
27      delete:
28       space=word_number*word_space;
29       time=  balance_factor*ld(word_number)*
30                           string_compare_time +
31            flag_set_time;
32
33      identify:
34       space=word_number*word_space;
35       time=balance_factor*ld(word_number)*
36                           string_compare_time;
37
38      browse:
39       space=word_number*word_space;
40       time=word_number*string_shift_time;
41
42   end binary_tree_dictionary;
```

Wir nehmen nun an, dass ein Spezialist für seine Arbeit ein Wörterbuch benötigt. Die vom Wörterbuch zu lösende Aufgabe beschreibt er in einem Generierungspunkt durch die folgenden Aufgaben-Attribute:

`task:`	Angabe der zu lösenden Aufgabenklasse
`word_max_number:`	maximale Anzahl der zu speichernden Wörter
`average_word_length:`	mittlere Wortlänge in Bytes
`operation_sequence:`	Reihenfolge, in der die Operationen des abstrakten Datentyps ausgeführt werden sollen

Für das Aufgabenattribut operation_sequence wird eine Schreibweise mit Wiederholungsfaktoren verwendet.

Die zu delegierende Aufgabenstellung sei nun zunächst wie folgt beschrieben:

```
task               = 'dictionary'
word_max_number    = 5000
average_word_length= 12
operation_sequence = '10(500(add),
                         100(delete),
                         400(identify),
                         browse)'
```

Der Dispatcher ermittelt in einem ersten Schritt die beiden Spezialistentypen „sorted_area_dictionary“ und „binary_tree_dictionary“ als kompetent. Um nun denjenigen Spezialistentyp auszuwählen, der die zu delegierende Aufgabe optimal lösen kann, simuliert der Dispatcher den Problemlösungs-Prozess für beide Spezialistentypen. Die Kosten der jeweiligen Operation soll er dabei aus dem Belastungsvektor $(\text{space}, \text{time})$ gemäß folgender Funktion berechnen:

$$\kappa(\text{space}, \text{time}) = 1.0 * \frac{\text{space}}{10240} * \text{time} + 3.0 * \text{time}$$

Der Dispatcher stellt dabei fest, dass ein Repräsentant des Spezialistentyps „sorted_area_dictionary“ die 31-fachen Kosten gegenüber einem Repräsentanten des Spezialistentyps „binary_tree_dictionary“ verursacht. Der Dispatcher überträgt also einem Repräsentanten des Spezialistentyps „binary_tree_dictionary“ die zu delegierende Aufgabe.

Ändert sich dagegen die Operationen-Sequenz und wird die zu delegierende Aufgabe wie folgt beschrieben:

```
task               = 'dictionary'
word_max_number    = 500
average_word_length= 12
operation_sequence = '500(add),
                      10(delete),
                      10000(identify),
                      browse'
```

so stellt der Dispatcher fest, dass die Kosten, die ein Repräsentant des Spezialistentyps „binary_tree_dictionary“ verursacht, das 1,5-fache der Kosten des Spezialistentyps „sorted_area_dictionary“ betragen. In diesem Falle generiert er zur Lösung der Aufgabe einen Repräsentanten des Spezialistentyps „sorted_area_dictionary“.

5 Zusammenfassung

In den vorangegangenen Abschnitten wurde eine Methode entwickelt, nach der das Zusammenspiel von maschinellen und intellektuellen Spezialisten während der Problembearbeitung optimiert wird. Das Modell beschreibt ein System, in dem die problemlösenden Komponenten ausschließlich durch den Austausch von Nachrichten miteinander kommunizieren. Jede Komponente ist ein selbständig arbeitender Spezialist, der Repräsentant eines Spezialistentyps ist. Die Beschreibungen aller Spezialistentypen bilden das technologische Wissen des Systems. Durch Ausnutzung dieses Wissens ist es möglich, für die Lösung einer zu delegierenden Aufgabe einen solchen Spezialisten einzurichten, der hinsichtlich eines gewünschten Ziels optimal arbeitet. Dadurch kann die Struktur des Systems der Spezialisten flexibel an die jeweiligen Besonderheiten der Problembearbeitung angepasst werden.

Literaturverzeichnis

Bentley/Appelt (1997): Bentley, R.; Appelt, W. et al: *Basic Support for Cooperative Work on the World Wide Web*. International Journal of Human-Computer Studies, 1997.

Bishop (2006): Bishop, C. M.: *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006.

Brügge/Dutoit (2004): Brügge, B.; Dutoit, A. H.: *Objektorientierte Softwaretechnik*. München: Pearson Education, 2004.

Conitzer/Sandholm (2002): Conitzer, V.; Sandholm, T.: *Complexity of mechanism design*. Proceedings of the "Uncertainty in Artificial Intelligence Conference" (UAI), Edmonton, Canada, 2002.

de Moura/Rodriguez/Ierusalimschy (2004): de Moura, A. L.; Rodriguez, N.; Ierusalimschy, R.: *Coroutines in Lua*. Journal of Universal Computer Science **10**(2004)7, S. 910-925.

Eberbach/Burgin (2007): Eberbach, E.; Burgin, M.: *Theoretical Framework for Cooperation and Competition in Evolutionary Computation*. Proc. 2nd Intern. Conf. on Software and Data Technologies, Barcelona, Spain, 2007, S. 229-234.

Jarosch (2007): Jarosch, H.: *Information Retrieval und Künstliche Intelligenz*. Wiesbaden: Deutscher Universitätsverlag, 2007.

Klusch/Bergamaschi/Edwards/Petta (2003): Klusch, M.; Bergamaschi, S.; Edwards, P.; Petta, P. (Hrsg.): *Intelligent Information Agents: The Agentlink Perspective*. Berlin, ...: Springer-Verlag, 2003.

Lipeck (1989): Lipeck, U. W.: *Algebraische Spezifikation abstrakter Datentypen*. Stuttgart: B. G. Teubner, 1989

Maurer (2006): Maurer, A.: *Bounds for Linear Multi-Task Learning*. Journal of Machine Learning Research **7**(2006), S. 117-139.

Russell/Norvig (2004): Russell, S.; Norvig, P.: *Künstliche Intelligenz. Ein moderner Ansatz*. 2. Auflage. München: Pearson Education, 2004.

Schnauder/Jarosch/Thieme (2001). Schnauder, V.; Jarosch, H.; Thieme, I.: *Praxis der Software-Entwicklung*. Renningen-Malmsheim: Expert Verlag, 2001.

Sommerville (2001): Sommerville, I.: *Software Engineering*. 6. Auflage. München: Pearson Studium, 2001.

Stone/Veloso (2000): Stone, P.; Veloso, M. M.: *Multiagent Systems: A Survey from a Machine Learning Perspective*. Autonomous Robots **8**(2000)3, S. 345-383.

Tong/Koller (2001) Tong, S.; Koller, D.: *Active learning: theory and applications*. Stanford University, 2001.

Vapnik (1999): Vapnik, V. N.: *The Nature of Statictical Learning Theory – Statistics for Engineering and Information Science*. 2. Auflage. Berlin, ...: Springer-Verlag, 1999.

Westfechtel/Conradi (2001): Westfechtel, B.; Conradi, R.: *Software Configuration Management and Engineering Data Management: Differences and Similarities*. Lecture Notes in Computer Science Nr. 1439. Berlin, Heidelberg, New York: Springer-Verlag, 2001, S. 96-106.

Working Papers des Institute of Management Berlin an der Fachhochschule für Wirtschaft Berlin

1 Bruche, Gert / Pfeiffer, Bernd: Herlitz (A) – Vom Großhändler zum PBS-Konzern – Fallstudie, October 1998
2 Löser, Jens: Das globale Geschäftsfeld „Elektrische Haushaltsgroßgeräte“ Ende der 90er Jahre – Fallstudie, October 1998
3 Lehmann, Lutz Lars: Deregulation and Human Resource Management in Britain and Germany – Illustrated with Coca-Cola Bottling Companies in Both Countries, March 1999
4 Bruche, Gert: Herlitz (B) - Strategische Neuorientierung in der Krise - Fallstudie, April 1999
5 Herr, Hansjörg / Tober, Silke: Pathways to Capitalism - Explaining the Difference in the Economic Development of the Visegrad States, the States of the Former Soviet Union and China, October 1999
6 Bruche, Gert: Strategic Thinking and Strategy Analysis in Business - A Survey on the Major Lines of Thought and on the State of the Art, October 1999, 28 pages
7 Sommer, Albrecht: Die internationale Rolle des Euro, December 1999, 31 Seiten
8 Haller, Sabine: Entwicklung von Dienstleistungen - Service Engineering und Service Design, January 2000
9 Stock, Detlev: Eignet sich das Kurs-Gewinn-Verhältnis als Indikator für zukünftige Aktienkursveränderungen?, March 2000
10 Lau, Raymond W.K.: China’s Privatization, June 2000
11 Breslin, Shaun: Growth at the Expense of Development? Chinese Trade and Export-Led Growth Reconsidered, July 2000, 30 pages

12 Michel, Andreas Dirk: Market Conditions for Electronic Commerce in the People’s Republic of China and Implications for Foreign Investment, July 2000, 39 pages
13 Bruche, Gert: Corporate Strategy, Relatedness and Diversification, September 2000, 34 pages
14 Cao Tingui: The People's Bank of China and its Monetary Policy, October 2001, 21 pages
15 Herr, Hansjörg: Wages, Employment and Prices. An Analysis of the Relationship Between Wage Level, Wage Structure, Minimum Wages and Employment and Prices, June 2002, 60 pages
16 Herr, Hansjörg / Priewe, Jan (eds.): Current Issues of China’s Economic Policies and Related International Experiences – The Wuhan Conference 2002 - , February 2003, 180 pages
17 Herr, Hansjörg / Priewe, Jan: The Macroeconomic Framework of Poverty Reduction An Assessment of the IMF/World Bank Strategy, February 2003, 69 pages
18 Wenhao, Li: Currency Competition between EURO and US-Dollar, June 2004, 18 pages
19 Kramarek, Maciej: Spezifische Funktionen des Leasings in der Transformationsperiode, June 2004, 32 Seiten
20 Godefroid, Peter: Analyse von Multimedia-Lern/Lehrumgebungen im Fach Marketing im englischsprachigen Bereich – inhaltlicher Vergleich und Prüfung der Einsatzfähigkeit an deutschen Hoschulen, September 2004, 48 Seiten
21 Kramarek, Maciej: Die Attraktivität des Leasings am Beispiel polnischer Regelungen der Transformationsperiode, April 2005, 33 Seiten
22 Pan, Liu / Tao, Xie: The Monetary Policy Transmission in China – „Credit Channel” and its Limitations
23 Hongjiang, Zhao / Wenxu, Wu / Xuehua, Chen: What Factors Affect Small and Medium-sized Enterprise’s Ability to Borrow from Bank: Evidence from Chengdu City, Capital of South-western China’s Sichuan Province, May 2005, 23 pages
24 Fritsche, Ulrich: Ergebnisse der ökonometrischen Untersuchung zum Forschungsprojekt Wirtschaftspolitische Regime westlicher Industrienationen, March 2006, 210 Seiten
25 Körner, Marita: Constitutional and Legal Framework of Gender Justice in Germany, November 2006, 14 pages
26 Tomfort, André: The Role of the European Union for the Financial Integration of Eastern Europe, December 2006, 20 pages
27 Gash, Vanessa / Mertens, Antje / Gordo, Laura Romeu: Are Fixed-Term Job Bad for Your Health? A Comparison between Western Germany and Spain, March 2007, 29 pages
28 Kamp, Vanessa / Niemeier, Hans-Martin / Müller, Jürgen: Can we Learn From Benchmarking Studies of Airports and Where do we Want to go From Here? April 2007, 43 pages
29 Brand, Frank: Ökonomische Fragestellungen mit vielen Einflussgrößen als Netzwerke. April 2007, 28 pages
30 Venohr, Bernd / Klaus E. Meyer: The German Miracle Keeps Running: How Germany’s Hidden Champions Stay Ahead in the Global Economy. May 2007, 31 pages
31 Tomenendal, Matthias: The Consultant-Client Interface – A Theoretical Introduction to the Hot Spot of Management Consulting. August 2007, 17 pages
32 Zenglein, Max J.: US Wage Determination System. September 2007, 30 pages
33 Figeac, Alexis: Socially Responsible Investment und umweltorientiertes Venture Capital. December 2007, 45 pages
34 Gleißner, Harald A.: Post-Merger Integration in der Logistik – Vom Erfolg und Misserfolg bei der Zusammenführung von Logistikeinheiten in der Praxis. March 2008, 27 pages

35 Bürkner, Fatiah: Effektivitätssteigerung im gemeinnützigen Sektor am Beispiel einer regionalen ‚Allianz für Tanz in Schulen'. April 2008, 29 pages
36 Körner, Marita: Grenzüberschreitende Arbeitsverhätnisse – Grundlinien des deutschen Internationalen Privatrechts für Arbeitsverträge. April 2008, 22 pages
37 Pan, Liu / Junbo, Zhu: The Management of China's Huge Foreign Reserve and ist Currency Composition. April 2008, 22 pages
38 Rogall, Holger: Essentiales für eine nachhaltige Energie- und Klimaschutzpolitik. May 2008, 46 pages
39 Maeser, Paul P.: Mikrofinanzierungen – Chancen für die Entwicklungspolitik und Rahmenbedingungen für einen effizienten Einsatz. May 2008, 33 pages
40 Pohland, Sven / Hüther, Frank / Badde, Joachim: Flexibilisierung von Geschäftsprozessen in der Praxis: Case Study „Westfleisch eG – Einführung einer Service-orientierten Architektur (SOA). June 2008, 33 pages
41 Rüggeberg, Harald / Burmeister, Kjell: Innovationsprozesse in kleinen und mittleren Unternehmen. June 2008, 37 pages
42 Domke, Nicole / Stehr, Melanie: Ignorieren oder vorbereiten? Schutz vor Antitrust Verstößen durch „Compliance"-Programme. June 2008, 25 pages
43 Ripsas, Sven / Zumholz, Holger / Kolata, Christian: Der Businessplan als Instrument der Gründungsplanung – Möglichkeiten und Grenzen. December 2008, 34 pages

Special Edition:
Ben Hur, Shlomo: A Call to Responsible Leadership. Keynote Speech at the FHW Berlin MBA Graduation Ceremony 2006. November 24th, 2006, Berlin City Hall, April 2007, 13 pages