Betriebliche Informationssysteme

Kurseinheit 7

Fallstudien

Autor:
Prof. Dr. Lars Mönch
unter Mitarbeit von:
Dr. Jens Zimmermann und
Dr. René Drießel

Fallstudien

Inhaltsübersicht

Lernziele

In diesem Kapitel werden zwei Fallstudien präsentiert. In der ersten Fallstudie wird ein verteiltes hierarchisches Produktionssteuerungssystem für Halbleiterfabriken beschrieben. Wir gehen auf die Architektur des Systems detailliert ein. Inbesondere wird auch die Implementierung des Systems mit Hilfe von Softwareagenten beschrieben.

Die zweite Fallstudie beschreibt einen Prototyp, der versucht, eine SOA für die Produktionsdomäne umzusetzen. Die beiden Fallstudien dienen dazu, die Konzepte und Techniken der vorherigen Kapitel zu illustrieren. Insbesondere wird auf Aspekte der Architektur und der Konstruktion eingegangen.

Nach dem Durcharbeiten von Abschnitt 7.1 sollen Sie verstanden haben, warum es sinnvoll ist, zur Produktionssteuerung in komplexen Produktionssystemen hierarchische Ansätze zu verwenden. Sie sind in der Lage, den Entwurf eines verteilten hierarchischen Produktionssteuerungssystems als Multi-Agenten-System zu erläutern.

Das Studium von Abschnitt 7.2 soll Sie in die Lage versetzen zu erkennen, dass das SOA-Konzept noch nicht ausgereift ist und dass die Entwicklung von Anwendungssystemen auf Basis von Webservices nicht trivial ist. Sie sollen weiterhin verstehen, wie Daten aus ERP-Systemen im Rahmen einer SOA genutzt werden.

Diese Kurseinheit soll Ihnen anhand von ausgewählten Beispielen zeigen, wie Inhalte dieses Kurses Eingang in aktuelle Forschungsfragestellungen finden. Wir weisen darauf hin, dass an der Entwicklung der beiden in dieser Kurseinheit vorgestellten Prototypen eine Reihe von Studenten der Wirtschaftsinformatik im Rahmen von Diplomarbeiten aktiv beteiligt waren. Insofern soll Ihnen diese Kurseinheit auch einen Eindruck von möglichen Abschlussarbeitsthemenstellungen vermitteln.

Die Übungsaufgaben innerhalb der Kurseinheit dienen der Überprüfung des erreichten Kenntnisstandes. Eine selbständige Bearbeitung der Aufgaben ist für das Erreichen eines tieferen Verständnisses des Stoffes wichtig.

Einige der in Kurseinheit 6 behandelten Konzepte aus dem Vertriebsbereich werden für Sie leichter verständlich sein, wenn Sie die praktischen Übungen am mySAP ERP-System im Rahmen der Einsendeaufgaben durchführen. Wir möchten Sie deshalb dazu ermutigen, sich an der Lösung der Einsendeaufgaben zu beteiligen.

7.1 FABMAS - ein agentenbasiertes Produktionssteuerungssystem

7.1.1 Motivation

komplexe Produktionsprozesse

Komplexe Produktionsprozesse [27] sind durch die folgenden Merkmale gekennzeichnet:

- einen zeitlich veränderlichen Produktmix,
- gleichzeitiges Auftreten unterschiedlicher Prozesstypen wie Batchprozesse, bei denen mehrere Lose gleichzeitig auf einer Maschine bearbeitet werden,
- vorgeschriebene Kundenendtermine,
- viele Arbeitsgänge umfassende Arbeitspläne,
- reihenfolgeabhängige Umrüstzeiten auf den Maschinen,
- parallele Maschinen sowie sekundäre Ressourcen,
- externe Störungen in Form von Veränderungen von Kundenendterminen und interne Störungen durch Maschinenausfälle.

Wir untersuchen in dieser Fallstudie ausschließlich diskrete Produktionssysteme (vergleiche dazu die Ausführungen in Abschnitt 6.1.2.2), die bezüglich der Organisationsform der Werkstattfertigung zuzuordnen sind und in denen eine Mischung aus kundenorientierter Fertigung (make to order) und anonymer Lagerfertigung (make to stock) vorliegt. In den betrachteten Produktionssystemen sind auf Grund von Kundenanforderungen Lose mit unterschiedlichem Gewicht vorhanden.

Basis- und Steuerungssystem

Ein Produktionssystem besteht aus dem durch Ressourcen gebildeten Basissystem und dem Steuerungssystem (vergleiche dazu die Ausführungen zum systemtheoretischen Grundmodell eines Unternehmens in Kurseinheit 1). Produktionsprozesse beschreiben die Inanspruchnahme von Ressourcen des Basissystems infolge von Entscheidungen des Steuerungssystems im zeitlichen Verlauf. Ein Produktionsprozess wird in einen Basis- und einen Steuerungsprozess unterteilt.

Komplexe Systeme werden in der Systemtheorie als Systeme von Subsystemen definiert [17]. Komplexe Produktionssysteme setzen sich, einer räumlichen und arbeitsorganisatorischen Dekomposition folgend, aus Produktionsbereichen zusammen. Ein Produktionsbereich ist durch eine bestimmte Anzahl von Gruppen paralleler Maschinen gekennzeichnet.

Halbleiterfertigungssysteme sind Beispiele für komplexe Produktionssysteme [27, 13, 18]. Der Einsatz zentral orientierter Planungs- und Steuerungsmethoden ist auf Grund der NP-Vollständigkeit der zu lösenden Ablaufplanungsprobleme

(vergleiche [13]), der häufigen Störungen, des zu bewältigenden Datenvolumens und der räumlichen und arbeitsorganisatorischen Dekomposition erschwert bzw. nicht möglich [9]. Stark vereinfacht bedeutet die **NP-Vollständigkeit** eines Ablaufplanungsproblems, dass keine Verfahren bekannt sind, die das Problem mit polynomialem Aufwand in Abhängigkeit von der Problemgröße lösen [30].

Ein verteiltes hierarchisches Rahmenwerk zur Steuerung komplexer Produktionssysteme, im Weiteren als Framework bezeichnet, wird vorgeschlagen, das die Vorteile dezentraler Ansätze mit den Vorteilen einer zentralen Steuerung verbindet. Wir betrachten dazu das folgende Beispiel.

Beispiel 7.1.1 (Dezentrale vs. zentrale Produktionssteuerung) *Eine dezentrale Produktionssteuerung ermöglicht die Verwendung von lokalen Daten zur Entscheidungsfindung. Der Vorteil einer zentralen Produktionssteuerung liegt in einer Berücksichtigung der globalen Situation bei der Entscheidungsfindung.*

Hierarchie

Wir definieren zunächst den Begriff der hierarchischen Abhängigkeit in Anlehnung an Schneeweiss [33].

Definition 7.1.1 (Hierarchie) *Zwei Systemobjekte befinden sich in einer hierarchischen Beziehung, wenn eine Asymmetrie bezüglich Entscheidungsrechten, Informationsstatus oder des Zeitpunkts der Entscheidungsfindung besteht.*

hierarchischer Ansatz

Hierarchische Ansätze zerlegen das Gesamtproblem sukzessive in leichter zu lösende Unterprobleme. Hierarchien können im Kontext einer modernen verteilten Entscheidungsfindung diskutiert und als konzeptioneller Rahmen für den Entwurf verteilter Entscheidungsunterstützungssysteme betrachtet werden (dieser Standpunkt wird unter anderem in [33] vertreten). Mit der vorliegenden Fallstudie werden drei Ziele verfolgt:

1. Die Anwendung der Theorie der verteilten hierarchischen Entscheidungsfindung für die Steuerung komplexer Produktionsprozesse wird dargestellt. Während in der Literatur das Problem der Steuerung einzelner Maschinengruppen oder Produktionsbereiche eines komplexen Produktionssystems häufig diskutiert wird, wird das Problem der Steuerung eines ganzen Produktionsunternehmens in dieser Fallstudie untersucht.

2. Es wird gezeigt, dass das vorgeschlagene hierarchische Steuerungssystem als Multi-Agenten-System implementiert werden kann. Insbesondere wird auch die Architektur und Konstruktion eines derartigen Multi-Agenten-Systems unter Verwendung des in Abschnitt 5.4.7.2 dargestellten agentenbasierten Rahmenwerks ManufAg beschrieben.

3. Operations Research (OR)-Methoden und Verfahren der Künstlichen Intelligenz (KI) werden vorgeschlagen, die optimierende und koordinierende Eigenschaften des vorgeschlagenen Steuerungsansatzes sicherstellen und als Dienstagenten im Multi-Agenten-System fungieren.

Die Fallstudie ist wie folgt aufgebaut. In Abschnitt 7.1.2 wird das methodische Vorgehen erläutert. Eine Problembeschreibung erfolgt in Abschnitt 7.1.3. Das vorgeschlagene Rahmenwerk wird in Abschnitt 7.1.4 diskutiert. Fragen der Realisierung der Steuerungsfunktionalität im Rahmen von Multi-Agenten-Systemen werden in 7.1.5 untersucht. Die Fallstudie endet mit der Beschreibung der simulationsbasierten Leistungsbewertung des entwickelten Prototyps.

7.1.2 Methodische Vorgehensweise

Da Modelle auf Annahmen beruhen, die von der Realität abweichen können, sind nach Ansicht von Porter [31] Rahmenwerke besser geeignet, Praxisunterstützung zu geben.

Rahmenwerk

Wir haben in Kurseinheit 2 und 5 bisher Software-Rahmenwerke kennengelernt. Wir verwenden an dieser Stelle einen umfassenderen Rahmenwerkbegriff. Dazu definieren wir:

Definition 7.1.2 (Allgemeines Rahmenwerk) *Unter einem allgemeinen Rahmenwerk wird eine Rahmenstruktur verstanden, die mit dem Ziel entworfen wird, eine bestimmte Betrachtungsweise auf eine zu lösende Problemklasse zu unterstützen. Die Konstruktion von Rahmenwerken erfolgt unter Verwendung von Modellen und Konzepten. Rahmenwerke bieten eine breite, umfassende Problembeschreibung gemeinsam mit Strukturierungsinstrumenten an, die geeignet sind, der Komplexität in Unternehmungen besser gerecht zu werden als das bei Verwendung von mit vielen Annahmen behafteten Modellen möglich ist.*

Im weiteren Verlauf sprechen wir anstelle von allgemeinen Rahmenwerken stets nur von Rahmenwerken.

Übungsaufgabe 7.1 (Allgemeines vs. Software-Rahmenwerk) *Begründen Sie, warum Software-Rahmenwerke auch allgemeine Rahmenwerke sind.*

Der Rahmenwerkgedanke wird in der Fallstudie aufgegriffen. In einer Analysephase wird das zu lösende Produktionssteuerungsproblem untersucht und eine auf der Theorie der verteilten hierarchischen Steuerung [33, 17] basierende Lösungsidee entwickelt. In einem weiteren Schritt erfolgt die Abstraktion zu einem Rahmenwerk für komplexe Produktionssysteme und dessen konzeptionelle softwaretechnische Umsetzung mit Softwareagenten.

Das Rahmenwerk wird empirisch auf die Steuerung der Herstellung integrierter Schaltkreise angewandt, indem ein entsprechender Prototyp implementiert wird und dann simulationsbasiert die Wirkung des Steuerungsansatzes untersucht wird.

7.1.3 Problemstellung

Trotz der sich in den letzten Jahren vollzogenen Entwicklungen in Wissenschaft und Praxis (vergleiche u. a. [34, 3, 25, 10, 40, 1] für eine Beschreibung des er-

reichten Forschungs- und Anwendungsstandes) sind für komplexe Produktionssysteme die nachfolgenden Fragen von außerordentlichem Interesse. Ausgehend von einem Produktionsplan, untersetzt in Kundenaufträge und Aufträge für eine anonyme Lagerfertigung zur Sicherstellung einer Grundlast, findet eine weitere Untersetzung der so gebildeten Aufträge in Lose statt. Das Produktionssteuerungspersonal muss rechnergestützt entscheiden:

Entscheidungen

1. wann welche Lose als unteilbare, dynamische Systemeinheiten eingesteuert werden,

2. wie die Zuteilung von Losen zu alternativen Maschinen erfolgen soll,

3. in welcher Reihenfolge Lose bearbeitet werden sollen, um Ziele wie hohe Liefertermineinhaltung bei höchstmöglicher Kapazitätsauslastung unter Berücksichtigung aller technologischen Restriktionen komplexer Produktionssysteme zu erreichen.

In der Praxis sind zentrale Implementierungen von fast vollständig "heterarchischen" Steuerungen, d. h. Anwendung von geeigneten Prioritätsregeln, anzutreffen. Prioritätsregelansätze berücksichtigen keine globalen Zielsetzungen und arbeiten räumlich und zeitlich lokal (vergleiche dazu die Ausführungen in Abschnitt 6.1.2.4).

Ansätze des OR adressieren häufig Produktionssteuerungsprobleme für ausgewählte Subsysteme eines Produktionssystems. Wir betrachten dazu das folgende Beispiel.

Subsystem

Beispiel 7.1.2 (Produktionssteuerung für Subsysteme) *Ein Modell zur Optimierung der Maschinenbelegung für identische parallele Maschinen wurde bereits in Beispiel 1.2.19 vorgestellt. Derartige Modelle können zum Einsatz kommen, wenn die Maschinen eine Engpassmaschinengruppe bilden.*

Die im Rahmen der "Intelligent Scheduling"-Initative entwickelten Produktionssteuerungssysteme [7, 9] waren auf Grund der fehlenden effizienten Implementierungsmöglichkeiten und der daraus resultierenden mangelnden softwaretechnischen Umsetzung Anfang der 90er Jahre nicht erfolgreich und haben zu keiner nachhaltigen Veränderung der betrieblichen Praxis geführt. Das folgende Beispiel illustriert die fehlenden Implementierungsmöglichkeiten.

Intelligent Scheduling

Beispiel 7.1.3 (Fehlende effiziente Implementierungsmöglichkeiten) *In Kurseinheit 4 wurde gezeigt, dass heute Middleware leicht anwendbar ist. Anfang der 90er Jahre standen Middlewareansätze nur hochspezialisierten Softwareentwicklern zur Verfügung. Aufgrund der damals fehlenden Automatisierung in der Produktion waren die Möglichkeiten zur Datenerfassung und Datenhaltung geringer ausgeprägt als heute.*

Zum damaligen Zeitpunkt waren bereits hierarchisch organisierte Modellsysteme vorhanden, allerdings waren Koordinationsverfahren, die rechnerübergreifende Interaktionen voraussetzen, nicht realisierbar. Wir betrachten dazu das folgende Beispiel.

Beispiel 7.1.4 (Hierarchisch organisiertes Modellsystem) *Die Architektur des Systems ReDS [7], das in einer Halbleiterfabrik von Siemens zum Einsatz kam, ist hierarchisch.*

Die Batch- und Rüstoptimierung, die Steuerung von geplanten und dynamischen Engpässen und die Behandlung von internen und externen Störungen sind typische Koordinationsprobleme [18]. Der Einsatz von OR- und KI-Methoden befindet sich in der Domäne komplexer Produktionssysteme am Anfang. Diese Situation wird insbesondere unter dem Aspekt einer zunehmenden Automatisierung der Produktionssysteme als unbefriedigend empfunden [34] und steht der von Mertens in [14] geforderten Vision der sinnhaften Vollautomatisierung im Weg.

Die Steuerung komplexer Produktionsprozesse kann durch räumlich und zeitlich verteiltes Lösen einer Folge von Entscheidungsproblemen im zeitlichen Ablauf beschrieben werden. Die verteilten Entscheidungen haben einen unterschiedlichen Stellenwert im System der Produktionssteuerung, aus diesem Grund erscheint eine hierarchische, asymmetrische Strukturierung der zu lösenden Entscheidungsprobleme als angemessen. Die Steuerung von Produktionssystemen kann folglich als **Design und Koordinierung von miteinander verbundenen Entscheidungen** aufgefasst werden [33].

In der Vergangenheit wurden hierarchische Entscheidungssysteme überwiegend für die mittelfristige Planung vorgeschlagen (vergleiche u. a. [5, 6, 39]). Anwendungen auf Produktionssteuerungsprobleme [35, 36] erfolgten bisher überwiegend im Rahmen der Kontrolltheorie. Die **Kontrolltheorie** versucht, vereinfacht gesagt, Steuerungsprozesse durch Differentialgleichungen abzubilden. Die mit der Kontrolltheorie verbundene Betrachtung von kontinuierlichen anstelle von diskreten Systemen erlaubt keine Behandlung einer kundenspezifischen Produktion. Weitere Defizite vorhandener hierarchischer Steuerungsansätze für komplexe Produktionssysteme bestehen in einer fehlenden Berücksichtigung terminorientierter Steuerungsziele, der nicht vorhandenen Möglichkeit einer nebenläufigen Lösung der Steuerungsprobleme und einer entsprechenden softwaretechnischen Unterstützung. Heute ist es möglich, für die im Zuge der sukzessiven Dekomposition des Gesamtproblems auftretenden Unterprobleme geeignete effiziente OR- und KI-Heuristiken bereitzustellen und so verteilt prozessnahe Entscheidungen zu treffen. Heuristiken sind Näherungsverfahren, die falls geeignet entworfen, gute Lösungen für eine Problemstellung liefern.

Multi-Agenten-System

Multi-Agenten-Systeme (vergleiche dazu die Ausführungen in Abschnitt 4.1) sind zur Implementierung von Systemen zur verteilten Entscheidungsfindung geeignet, wenn es gelingt, die Optimierungs- und Koordinationsfähigkeiten von

Multi-Agenten-Systemen durch Ergebnisse der Theorie verteilter hierarchischer Entscheidungen und durch OR- und KI-Verfahren anzureichern [33]. Die Theorie verteilter hierarchischer Entscheidungen liefert Vorschläge für die Gestaltung und Nutzung hierarchischer Systeme. Eine Analyse der in [2, 28, 29] beschriebenen Multi-Agenten-Systeme für die Produktionsdomäne zeigt, dass dabei noch große Defizite bestehen.

Übungsaufgabe 7.2 (Verteilte Systeme zur Entscheidungsfindung) *Begründen Sie die Aussage, dass Multi-Agenten-Systeme zur Implementierung von Systemen zur verteilten Entscheidungsfindung gut geeignet sind, wenn die oben beschriebene Anreicherung vorgenommen wird.*

Aufbauend auf der in [18] durchgeführten Domänenanalyse wurden die folgenden Thesen aufgestellt:

1. Ein Verzicht auf jede Form von Hierarchie liefert keine besseren Ergebnisse als die derzeit dominierenden prioritätsregelbasierten Ansätze. Um ein unter globalen Gesichtspunkten günstiges Verhalten zu gewährleisten, sind zentrale Elemente in die Steuerung einzubeziehen. Dabei stellen Entscheidungen übergeordneter Ebenen lediglich einen Rahmen für die Entscheidungen untergeordneter Steuerungsebenen dar, d. h., untergeordnete Systemeinheiten treffen autonom Steuerungsentscheidungen.

2. Die in klassischen hierarchischen Steuerungsansätzen vorhandene Stratifikation, Staffelung und zeitliche Dekomposition der entscheidungsfähigen Einheiten kann als Basis für die Verteilung der Steuerungsfunktionalität benutzt werden. Dadurch ist eine Verteilung der Steuerungsaufgaben auf mehrere Computer möglich.

3. Die Koordinationstheorie muss weiterentwickelt werden. Eine Koordinationssprache ist zu entwickeln, welche die im Zuge der Stratifikation und Staffelung entstandenen Entscheidungseinheiten informationstechnisch miteinander verbindet.

4. Neben entscheidungsfähigen Einheiten ("Entscheideragenten") müssen Softwareeinheiten ("Dienstagenten") betrachtet werden, die geeignet weiterzuentwickelnde OR- und KI-Methoden für komplexe Produktionssysteme kapseln. Die Trennung in Entscheider- und Dienstagenten ermöglicht eine leichte Austauschbarkeit von Steuerungsalgorithmen und bietet Vorteile bei der Anpassung des Steuerungssystems an neuartige Situationen.

7.1.4 Rahmenwerk zur verteilten hierarchischen Steuerung

In diesem Abschnitt wird ein Modellsystem für eine verteilte hierarchische Steuerung komplexer Produktionsprozesse vorgeschlagen. Die drei verwendeten Steue-

rungsebenen und die benutzten Steuerungsverfahren werden beschrieben. Das verteilte hierarchische Steuerungsverfahren wird vorgestellt.

7.1.4.1 Modellsystem für eine verteilte hierarchische Produktionssteuerung

Modellsystem

Hierarchische Steuerungssysteme ordnen die zu lösenden Entscheidungsaufgaben unterschiedlichen Ebenen zu, die in einer Vorgabe- und gegebenenfalls Rückkopplungsbeziehung stehen. Eine in der Hierarchie übergeordnete Ebene beeinflusst umfangreichere Teile des Gesamtprozesses. Übergeordnete Ebenen haben längere Entscheidungsperioden und beschäftigen sich mit langsameren Aspekten des zu steuernden Prozesses. Die Entscheidungsprobleme übergeordneter Ebenen sind weniger strukturiert, enthalten in stärkerem Maße Unsicherheiten und sind schwieriger quantitativ zu formalisieren [17]. Hierarchieebenen werden durch ein Modell des zu steuernden Prozesssegmentes, eine Menge von Entscheidungskriterien, einen Alternativenraum sowie einen bestimmten Informationsstatus beschrieben [33].

Hierarchieebenen

Ein aus drei Ebenen bestehendes Steuerungssystem wird vorgeschlagen. Die einzelnen Ebenen dienen in dieser Reihenfolge:

- der Terminierung von Losen,
- der detaillierten Ablaufplanung auf Basis der Terminierungsvorgaben,
- der Implementierung der ermittelten Ablaufpläne auf der untersten Hierarchieebene.

Die Ebenen werden in dieser Reihenfolge als Produktionssystem-, Produktionsbereichs- und Maschinengruppensteuerungsebene bezeichnet.

Zeithorizonte

Die zu lösenden Entscheidungsprobleme der drei Ebenen besitzen unterschiedliche Zeithorizonte. Außerdem ist die Tragweite der auf jeder Ebene zu treffenden Entscheidungen unterschiedlich. Die durch die vertikale Dekomposition entstandenen Produktionsbereichs- und Maschinengruppensteuerungsebenen werden zusätzlich horizontal dekomponiert. Im Gegensatz zur vertikalen Dekomposition sind die auftretenden Entscheidungsprobleme für die Organisationseinheiten Produktionsbereich und Maschinengruppe durch einen identischen Zeitbezug, einen identischen Abstraktionsgrad und eine identische Struktur gekennzeichnet.

Jede der drei Steuerungsebenen verfügt über ein eigenes Datenmodell. Die Granularität der Modelle ist für die obere Steuerungsebene gröber als für die beiden nachfolgenden Steuerungsebenen. Jede einem Produktionsbereich zugeordnete Entscheidungseinheit arbeitet auf den Bewegungsdaten des Produktionsbereichs. Die Stammdaten werden redundant für jede einem Produktionsbereich zugeordnete Entscheidungseinheit vorgehalten. Ähnliches gilt für die den Maschinengruppen zugeordneten Einheiten.

Wir beschreiben nun der Reihe nach die einzelnen Steuerungsebenen. Die **Produktionssystemsteuerungsebene** verwendet zur Entscheidungsfindung ein aggregiertes Modell. Die Aggregation wird erreicht, indem aufeinanderfolgende in einem Produktionsbereich auszuführende Arbeitsgänge zu Makroarbeitsgängen zusammengefasst werden. Das Terminierungsproblem kann als Optimierungsproblem mit binären Variablen und periodengenauer Auflösung zur Minimierung der gewichteten Verspätung der Lose formuliert werden. Zur Lösung des Problems werden Vorwärts- und Rückwärtsterminierungsmethoden (vergleiche dazu die Ausführungen in 6.1.2.3) verwendet.

Produktionssteuerungsebene

Die obere Ebene des hierarchischen Ansatzes liefert Vorgaben für den zeitlichen Durchlauf der Lose durch das Produktionssystem entsprechend der Segmentierung in Produktionsbereiche. Ein Modell des geplanten Engpassproduktionsbereichs findet bei der Entscheidungsfindung der oberen Ebene Berücksichtigung, d. h., es wird implizite reaktive Antizipation [33] angewandt. Als Koordinationsform wird somit wechselseitige Abstimmung zwischen der Produktionssystem- und Produktionsbereichssteuerungsebene gewählt [17, 16]. Das Interaktionsvorhersageprinzip [17, 16] wird als Koordinationsprinzip angewandt, da die Produktionssystemsteuerungsebene durch die Losterminierung abschätzt, wann Lose infolge der Bearbeitung in einem Produktionsbereich in anderen Produktionsbereichen eintreffen.

Die **Produktionsbereichssteuerungsebene** dient der Steuerung einzelner Produktionsbereiche unter Verwendung von disjunktiven Graphenmodellen. In diesen Modellen werden die Starttermine und die geplanten Endtermine für die Lose aus den Terminvorgaben der oberen Ebene ermittelt. Auf diese Weise wird der Alternativenraum des Entscheidungsproblems für einen einzelnen Produktionsbereich eingeschränkt. Die Verwendung von Terminen der Produktionssystemsteuerungsebene bei der Berechnung von detaillierten Ablaufplänen ähnelt der als Verfeinerung von Plänen bezeichneten Koordinationsform [39].

Produktionsbereichssteuerungsebene

Das gewählte Vorgehen ermöglicht die Lösung von Koordinationsproblemen der Batch- und Rüstoptimierung. Eine Lösung des Problems einer koordinierten Steuerung von Engpassmaschinengruppen wird durch die maschinengruppenübergreifende Betrachtungsweise ebenfalls beträchtlich erleichtert. Die Maschinenbelegungsprobleme für die Produktionsbereiche werden mit Hilfe der Shifting-Bottleneck-Heuristik [13] gelöst. Die Shifting-Bottleneck-Heuristik dekomponiert das Ablaufplanungsproblem in eine Folge von Ablaufplanungsproblemen für einzelne Maschinengruppen. Dazu werden zeitliche Abhängigkeiten sowie Abhängigkeiten zwischen Arbeitsgängen, die sich aus den Arbeitsplänen ergeben, durch einen disjunktiven Graphen modelliert. Jeder Knoten des Graphen repräsentiert die Ausführung eines Arbeitsgangs auf einer Maschine einer Maschinengruppe. Der genaue Verfahrensablauf ist Gegenstand des Kurses „Entscheidungsmethoden in unternehmensweiten Softwaresystemen“.

Die Ablaufplanungsprobleme für die Produktionsbereiche werden verteilt gelöst, da die Vorgaben der oberen Steuerungsebene in Form von Start- und End-

terminen für eine Entkopplung der zu lösenden Probleme sorgen. Eine produktionsbereichsübergreifende Koordination findet statt, indem durch die Ablaufpläne ermittelte früheste Verfügbarkeitstermine der Lose zwischen den Produktionsbereichen ausgetauscht werden und so auf iterative Weise günstigere Maschinenbelegungen bezüglich der absoluten gewichteten Verspätung der Lose des betrachteten Produktionsbereichs ermittelt werden. Das Vorgehen zur sukzessiven Verbesserung der Ablaufpläne für drei Produktionsbereiche, mit PB1, PB2 und PB3 bezeichnet, ist in Abbildung 7.1 dargestellt und wird als verteilte Shifting-Bottleneck-Heuristik (DSBH) bezeichnet.

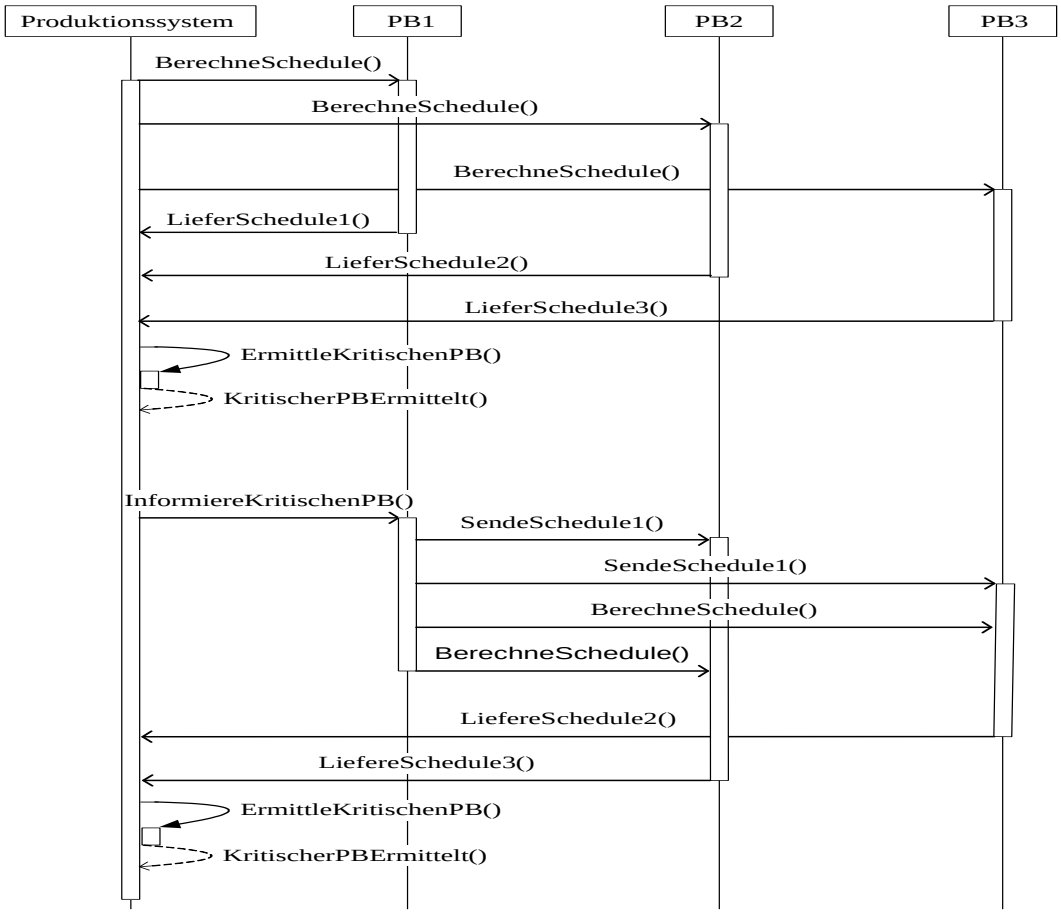

Abbildung 7.1: Sequenzdiagramm für produktionsbereichsübergreifende Koordination

Das iterative, produktionsbereichsübergreifende Verfahren kann als „nachträgliche“ Koordination [39] interpretiert werden. Die obere und mittlere Steuerungsebene realisieren eine kooperative Steuerung des Produktionssystems. Die Steuerung der einzelnen Produktionsbereiche erfolgt autonom, da die Berechnung der Ablaufpläne lediglich grobe terminliche Vorgaben der oberen Steuerungsebene berücksichtigen muss.

Maschinengruppensteuerungsebene

Die **Maschinengruppensteuerungsebene** dient der Implementierung der Ablaufpläne der mittleren Ebene durch Maschinengruppen. Ein ereignisdiskretes Modell, das wesentliche Ereignisse wie Statusänderungen von Losen und Maschinen enthält, wird in regelmäßigen Abständen aktualisiert. Vorgaben der mittleren Ebene in Form von Ablaufplänen werden zur Maschinenbelegung verwendet.

Falls die Ablaufpläne nicht gültig sind, wird ein kontraktnetzartiges Allokationsverfahren zur Maschinenbelegung [22] verwendet (vergleiche für Kontraktnetzansätze die Ausführungen in Abschnitt 3.4.3.2 dieses Kurses). In Simulationsstudien konnte gezeigt werden, dass sich das Allokationsverfahren in Produktionssystemen mit Störungen besser bezüglich terminorientierter Leistungsmaße verhält als Prioritätsregeln [19]. Auf dieser Steuerungsebene wird ebenfalls die Wahrung der Autonomie der entscheidungsfähigen Einheiten deutlich. Den Ablaufplänen wird nur solange gefolgt, wie diese gültig sind. In der gegenteiligen Situation werden autonom Maschinenbelegungsentscheidungen getroffen. Die verwendeten Koordinationsformen und -prinzipien für den vorgeschlagenen Ansatz sind in Tabelle 7.1 zusammengefasst. Die Produktionssystem-, Produktionsbereichs- und Maschinengruppensteuerungsebene werden durch ein „T“, „M“ und „B“ gekennzeichnet.

Tabelle 7.1: Koordination im verteilten hierarchischen Steuerungssystem

Ebene	Koordinationsform		Koordinationsprinzip
	vertikal	horizontal	
T	(primale) hierarchische Koordination mit Vorgaben und Rückkopplung bei nicht-reaktiver Antizipation	entfällt	Interaktionsvorhersageprinzip
T	wechselseitige Abstimmung bei impliziter reaktiver Antizipation	entfällt	Interaktionsvorhersageprinzip
M	(primale) hierarchische Koordination mit Vorgaben und Rückkopplung	Planverfeinerung, nachträgliche Koordination	nicht benutzt
B	entfällt	(primale) dezentrale Koordination	nicht benutzt

7.1.4.2 Verteiltes hierarchisches Produktionssteuerungskonzept

verteilter hierarchischer Steuerungsalgorithmus

Aus Platzgründen wird in diesem Abschnitt lediglich eine vereinfachte Version des verteilten hierarchischen Steuerungsalgorithmus angegeben, die keine Reschedulingaktivitäten enthält und in der die Losterminierung und die Berechnung von Ablaufplänen für die Produktionsbereiche jeweils zum Zeitpunkt T_0 stattfinden. Beide Bedingungen können entsprechend abgeschwächt werden (vergleiche dazu [24]).

Die Länge des Terminierungshorizontes wird mit T und die Länge des Schedulinghorizontes mit τ bezeichnet. Das Steuerungsverfahren Distributed-Hierarchical-Production-Control-Approach (DHPCA) kann wie folgt beschrieben werden:

DHPCA: Verteilter hierarchischer Steuerungsalgorithmus für komplexe Produktionssysteme

1. Terminiere die im Produktionssystem zum Zeitpunkt t_0 befindlichen bzw. innerhalb des Terminierungszeitraums $(t_0, t_0 + T)$ eintreffenden Lose.

2. Antizipiere das Verhalten des geplanten Engpassproduktionsbereichs, indem ein Ablaufplan für diesen Produktionsbereich mit Schedulinghorizont $(t_0, t_0 + \tau)$ berechnet wird.

3. Falls der in Schritt 2 ermittelte Ablaufplan zu einer hinreichend kleinen antizipierten gewichteten Verspätung der Lose führt, verzweige zu Schritt 4. Andernfalls verzweige zu Schritt 1. und ändere die von der Produktionssystemsteuerungsebene angewandte Losterminierungsstrategie.

4. Iteration über alle Produktionsbereiche, für die noch kein auszuführender Ablaufplan berechnet wurde: Bestimme für jeden dieser Produktionsbereiche einen Ablaufplan für den Schedulinghorizont $(t_0, t_0 + \tau)$ unter Verwendung von Vorgaben für früheste Starttermine und geplante Fertigstellungstermine.

5. Wähle einen Produktionsbereich aus der Menge der Produktionsbereiche, für die noch kein auszuführender Ablaufplan ermittelt wurde, unter Verwendung bestimmter Kriterien wie Engpasscharakteristik aus. Nehme diesen Produktionsbereich in die Menge der Produktionsbereiche mit auszuführendem Ablaufplan auf. Teile die tatsächlich notwendigen Start- und Endtermine der Lose, die für die Implementierung dieses Ablaufplans notwendig sind, den Entscheidungseinheiten der Produktionsbereiche mit, für die noch kein auszuführender Ablaufplan vorliegt. Falls die Menge der Produktionsbereiche, für die kein auszuführender Ablaufplan vorhanden ist, leer ist, verzweige zu Schritt 6, andernfalls verzweige zu Schritt 4.

6. Nehme eine Maschinenbelegung zum Zeitpunkt $t_0 \leq t_1 \leq t_0 + \tau$ vor, falls zum Zeitpunkt t_1 eine Bearbeitung von Losen auf dieser Maschine möglich ist. Berücksichtige dabei den in Schritt 4 bzw. 5 ermittelten Ablaufplan. Verwende ein Kontraktnetzverfahren zur Maschinenbelegung, falls der Ablaufplan ungültig ist.

7. Falls für die aktuelle Zeit t_{act} gilt $t_{act} = t_0 + \tau$, setze $t_0 = t_0 + \tau$ und verzweige zu Schritt 1.

Der gewählte Ansatz unterscheidet sich von anderen neueren hierarchischen Steuerungsansätzen [35, 36] dadurch, dass die Entscheidungseinheiten der mittleren Steuerungsebene koordiniert werden, dass terminorientierte Entscheidungskriterien verwendet und dass die Entscheidungsprobleme verteilt gelöst werden.

7.1.5 Softwaretechnische Umsetzung des Rahmenwerks durch Multi-Agenten-Systeme

Dieser Abschnitt beschreibt den Multi-Agenten-System-Ansatz, der zur Implementierung des Rahmenwerks verwendet wird. In diesem Abschnitt werden im Sinne einer Spezifikation innerhalb des generischen Architekturrahmens von Sinz (vergleiche Abschnitt 2.2) die Sichten

Sichten

1. Komponentensicht,
2. Datensicht,
3. Sicht auf Nachbarsysteme,
4. Laufzeitsicht,
5. Funktionsmodellsicht

beschrieben. Das zu bauende Softwaresystem besteht aus fachlichen und technischen Komponenten. Zu den technischen Komponenten für das zu spezifizierende verteilte Steuerungssystem gehören die Laufzeitumgebung des Multi-Agenten-Systems und die blackboardartige Datenschicht. Die Datensicht wird bei der Spezifikation des Datenhaushalts des verteilten Steuerungssystems eingenommen. Gleichzeitig wird die Interaktion des Steuerungssystems mit anderen betrieblichen Informationssystemen zur Datenbereitstellung beschrieben. Die Laufzeitsicht wird eingenommen, um das Kommunikationsverhalten des Systems zu spezifizieren. Die Funktionssicht wird bei der Beschreibung der generischen Funktionalität für Entscheider- und Dienstagenten sowie von Mustern für das Zusammenwirken von Entscheider- und Dienstagenten benutzt.

Nach einer Identifizierung der benötigten Agenten skizzieren wir eine Infrastruktur für Multi-Agenten-Systeme. Die Diskussion von Kommunikations- und Koordinationsfragestellungen erfolgt.

7.1.5.1 Motivation eines agentenbasierten Ansatzes

Softwareagenten

Agenten sind in der Lage, mit anderen Agenten zu kommunizieren und auf diese Art und Weise in Interaktion zu treten. Dadurch ist es möglich, Koordinationsmechanismen in Multi-Agenten-Systemen zu verwenden und auch softwaretechnisch rechnerübergreifend zu implementieren. Agenten lösen Entscheidungsprobleme (vergleiche dazu Definition 4.1.1 sowie die Ausführungen in Abschnitt 4.1). Da in verteilten hierarchischen Steuerungssystemen die Entscheidungsprobleme ebenfalls eine herausragende Bedeutung haben, wird in Tabelle 7.2 der Zusammenhang zwischen diesen beiden Systemarten untersucht. Es wird deutlich, dass verteilte hierarchische Systeme und Multi-Agenten-Systeme in den folgenden Punkten übereinstimmen:

Analogie von verteilten hierarchischen Steuerungssystemen und MAS

- sowohl verteilte hierarchische Systeme als auch Multi-Agenten-Systeme ermöglichen eine verteilte Entscheidungsfindung,
- die Entscheidungsfindungskompetenz ist auf unterschiedliche Systemeinheiten verteilt,
- in beiden Arten von Systemen ist Kommunikation zwischen den Entscheidungseinheiten notwendig,
- auf Grund der verteilten Entscheidungsfindung ist Koordination zur Harmonisierung erforderlich.

Multi-Agenten-Systeme können als Implementierungsrahmen für verteilte hierarchische Systeme verstanden werden [33]. Die Problemlösungsfähigkeiten sind in verteilten hierarchischen Systemen auf Grund der dort anzutreffenden funktionalen Dekomposition stärker ausgeprägt. Der Koordinationsgedanke ist direkt in der hierarchischen Dekomposition verankert. Multi-Agenten-Systeme betonen stärker den Implementierungsgesichtspunkt.

7.1.5.2 Identifizierung der benötigten Softwareagenten

PROSA

Unter Benutzung der Referenzarchitektur PROSA für holonische Produktionssysteme (vergleiche dazu die Ausführungen in Beispiel 2.2.6 und in Aufgabe 4.1) wird die zur softwaretechnischen Realisierung des geforderten Steuerungsverhaltens notwendige Agentenhierarchie abgeleitet. Entscheideragenten werden den Stratifikations- und Staffelungselementen zugeordnet und sind für die Entscheidungsfindung der durch sie repräsentierten Systembestandteile zuständig. Die folgenden Typen von Entscheideragenten werden unterschieden:

Entscheideragenten

- Produktionssystemagenten,
- Produktionsbereichsagenten,
- Maschinengruppenagenten,

Tabelle 7.2: Gegenüberstellung von verteilten hierarchischen Steuerungssystemen und Multi-Agenten-Systemen

Dimension	Verteiltes hierarchisches Steuerungssystem	Multi-Agenten-System
Systembestandteile	Entscheidungseinheiten der Ebenen mit Entscheidungsmodell entsprechend der Stratifikation und Staffelung	Agenten, die zu Agentenorganisationen zusammengefasst werden
Verteilung der Entscheidungsfunktionalität	entsprechend räumlicher und zeitlicher Gesichtspunkte	Verteilung auf mehrere Rechner
Autonomie	Vorgaben von Entscheidungseinheiten übergeordneter Ebenen stellen einen Rahmen für die Entscheidungen der untergeordneten Ebene dar, iterative Abstimmungsprozesse	Agenten handeln autonom, Begrenzung der Autonomie bei der Entscheidungsfindung durch Koordinationsmechanismen möglich
Koordination	übergeordnete Entscheidungseinheiten koordinieren untergeordnete Entscheidungseinheiten, Koordination durch Vorgabe von Handlungsrahmen und iterative Abstimmungsprozesse	implizite Koordination durch Verhandlungen, Koordination von Plänen
Entscheidungsfähigkeit	jede Entscheidungseinheit verfügt über Entscheidungsmodell, Antizipation des Verhaltens anderer Entscheidungseinheiten für Entscheidungsfindung möglich	regelbasiert in reaktiven Agentenarchitekturen, planbasierte Architekturen ermöglichen Entscheidungsfindung unter Berücksichtigung einer symbolischen Umweltrepräsentation
Kommunikation	für Feedback- und Feedforwardprozesse sowie für iterative Abstimmungsprozesse zwischen den Entscheidungseinheiten erforderlich, nicht notwendigerweise rechnerübergreifend	prinzipiell rechnerübergreifend möglich, wesentlicher Bestandteil des Agentenkonzeptes
Verteilte Implementierung	Verteilung für nebenläufige Bearbeitung wird in der Literatur nur am Rande diskutiert [7]	Verteilung auf Grund rechnerübergreifender Kommunikationsfähigkeiten prinzipiell unbegrenzt

- Losagenten,
- Agenten zur Repräsentation von vorbeugenden Instandhaltungsaufträgen (PM-Agenten)
- Agenten zur Darstellung von sekundären Ressourcen.

Die Identifikation der Agenten erfolgt unter Berücksichtigung des zu lösenden Steuerungsproblems. Die identifizierten Agenten sind in Abbildung 7.2 dargestellt. Bei der Lösung der Entscheidungsprobleme werden die Entscheideragenten durch Dienstagenten unterstützt, welche die notwendige Ablaufplanungs- und Monitoringfunktionalität kapseln. Wir unterscheiden zwischen

Dienstagenten

- Schedulingagenten,
- Monitoringagenten.

Wesentlicher Vorteil der Betrachtung von Entscheider- und Dienstagenten ist eine Entkopplung der Struktur des Steuerungssystems von den Steuerungsalgorithmen. Die Dienstagenten dienen als Container für OR- und KI-Algorithmen.

Dienstagenten können dabei als ein Schritt in Richtung der Realisierung einer SOA (vergleiche für den SOA-Begriff die Ausführungen in Abschnitt 2.3.4) verstanden werden.

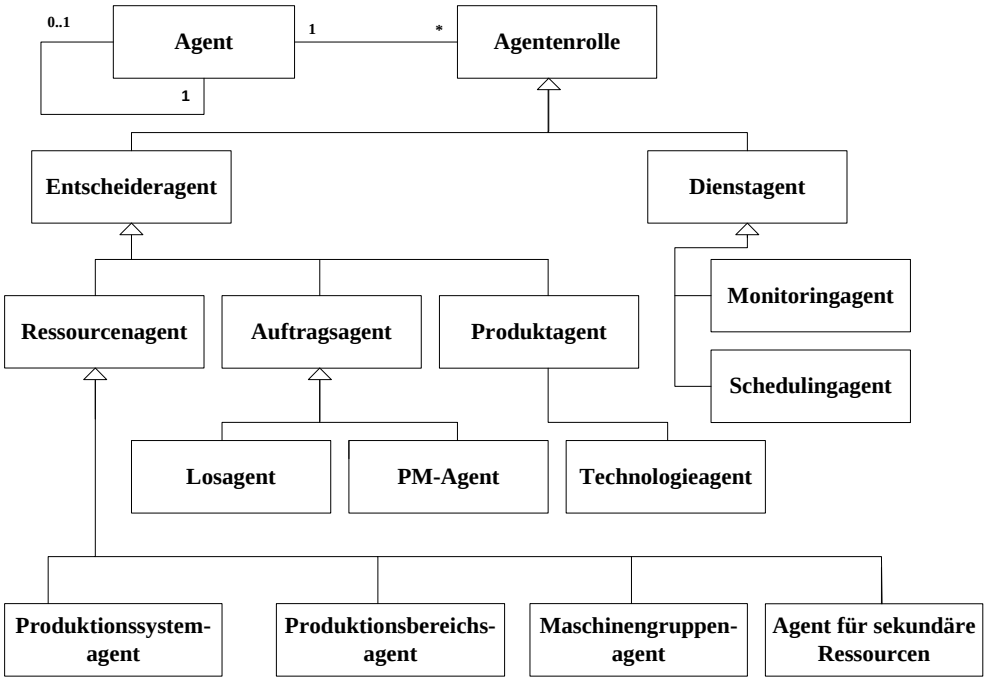

Abbildung 7.2: Agenten in FABMAS

Der Zusammenhang zwischen den Entscheidungseinheiten des Rahmenwerks und den Entscheideragenten ist in Abbildung 7.3 dargestellt.

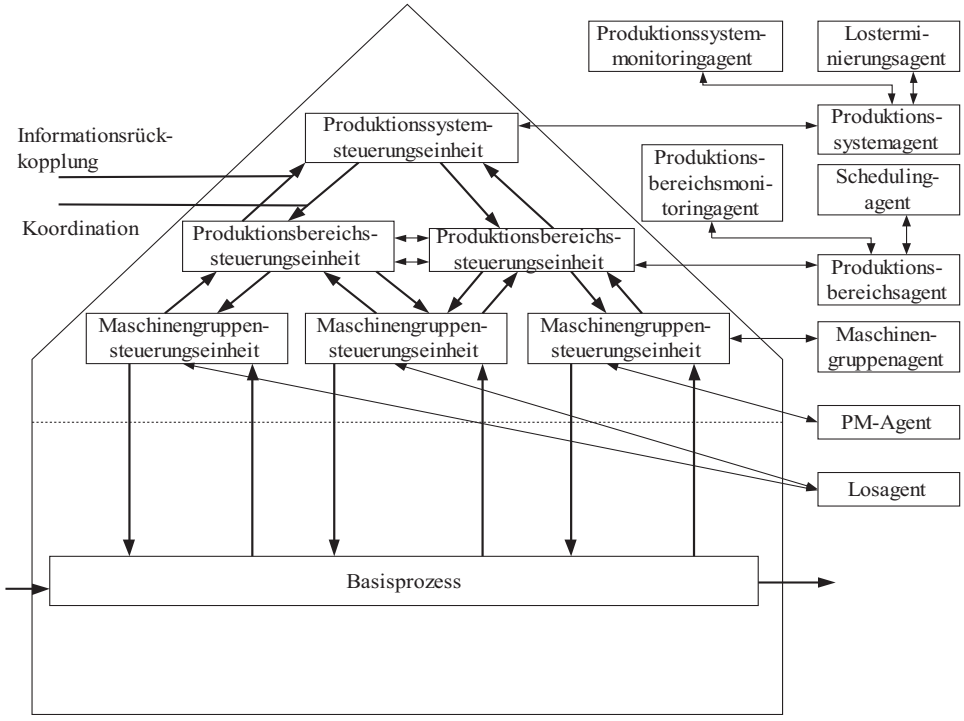

Abbildung 7.3: Zusammenhang Steuerungsansatz und Agententypen

Eine prototypische agentenbasierte Implementierung eines hierarchisch organisierten Steuerungssystems unter Verwendung des vorgeschlagenen Rahmenwerks für die Domäne der Fertigung integrierter Schaltkreise wurde in [23] beschrieben. Der Prototyp wurde unter Verwendung der Programmiersprachen C++ und C# sowie der .NET-Middleware entwickelt. Dabei wurde unter anderem für die Infrastruktur auf das ManufAg-Rahmenwerk (siehe Abschnitt 5.4.7.2) zurückgegriffen.

In Abbildung 7.4 wird gezeigt, wie FABMAS auf Basis von ManufAg entwickelt wird. Insbesondere wird deutlich, dass Entscheider- und Dienstagenten in der in Abschnitt 5.4.7.2 beschriebenen Art und Weise zusammenwirken. Die dort abgebildete blackboardartige Datenschicht ist Bestandteil der Datenhaltung in FABMAS und wird in Abschnitt 7.1.5.3 genauer dargestellt. Der Simulator AutoSched AP sowie das Simulationsmodell dienen der Leistungsbewertung von FABMAS und werden in Abschnitt 7.1.6 beschrieben. ManufAg

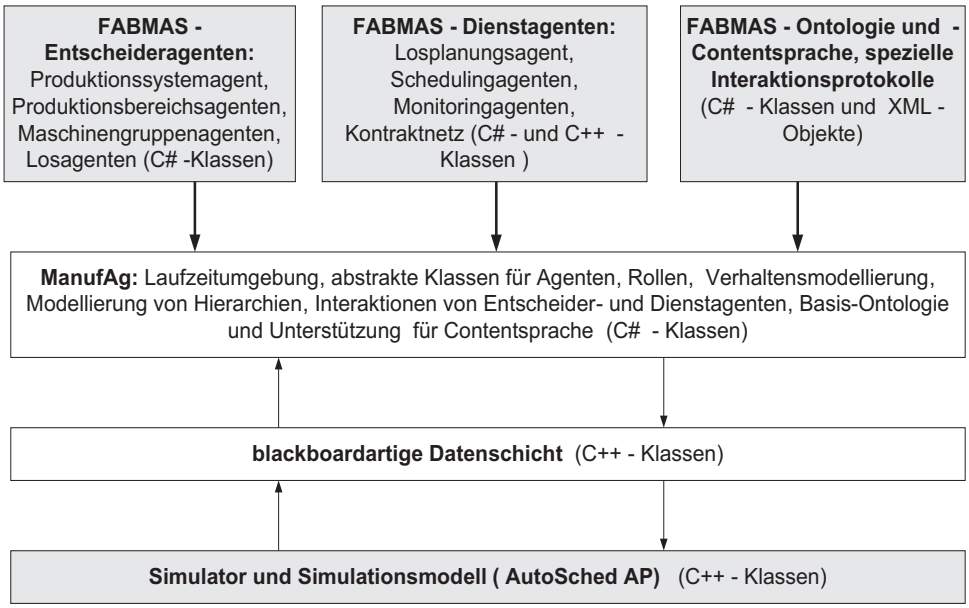

Abbildung 7.4: Zusammenhang FABMAS und ManufAg

7.1.5.3 Datenhaushalt des vorgeschlagenen Multi-Agenten-Systems

Datenhaltung

Die Konzeption des Datenhaushaltes erfolgt unter dem Gesichtspunkt der Online-Fähigkeit des verteilten hierarchischen Steuerungssystems. Es wird das Ziel verfolgt, den zur Übertragung von Daten notwendigen Nachrichtenaustausch zwischen einzelnen Agenten zu minimieren. Es werden die Prinzipien

- Redundanz,
- gemeinsame Nutzung von Daten durch mehrere Agenten,
- direkte Referenzierung von Datenobjekten

eingesetzt. Die für das Multi-Agenten-System notwendigen Daten werden in Datenobjekten abgelegt. Datenobjekte werden im Hauptspeicher des Rechners vorgehalten. Dadurch wird ein sehr schneller Zugriff auf die Datenobjekte ermöglicht und die Online-Fähigkeit des Multi-Agenten-Systems sichergestellt. Eine redundante Datenhaltung verringert die Anzahl von Informationen, die zwischen den Agenten ausgetauscht werden, und lässt eine Verteilung des Multi-Agenten-Systems auf mehrere Rechner zu. Die gemeinsame Nutzung von Daten durch mehrere Agenten verkleinert die Anzahl der Datenobjekte, die im Hauptspeicher vorgehalten werden müssen. Durch die direkte Referenzierung von Objekten wird vermieden, die in den Objekten enthaltene Information durch Nachrichten zu übermitteln.

Betriebliche Informationssysteme (MES, ERP-Systeme oder verschiedene Spezialdatenbanken) stellen die Daten bereit, die vom verteilten hierarchischen

Steuerungssystem für Steuerungsentscheidungen benutzt werden. Kern der Architektur ist eine blackboardartige Online-Datenschicht. Die Datenschicht enthält die zur Steuerung wesentlichen Daten in Form geeigneter Objekte, die durch die betrieblichen Informationssysteme ständig aktualisiert werden. Das Multi-Agenten-System besitzt eine Schnittstelle, die über die Online-Datenschicht mit Daten versorgt wird. Die Schnittstelle versendet Nachrichten an die Agenten. Die Agenten führen nach Erhalt einer entsprechenden Nachricht eine Aktualisierung der jeweiligen Datenobjekte durch. Die beschriebene Architektur ist in Abbildung 7.5 dargestellt.

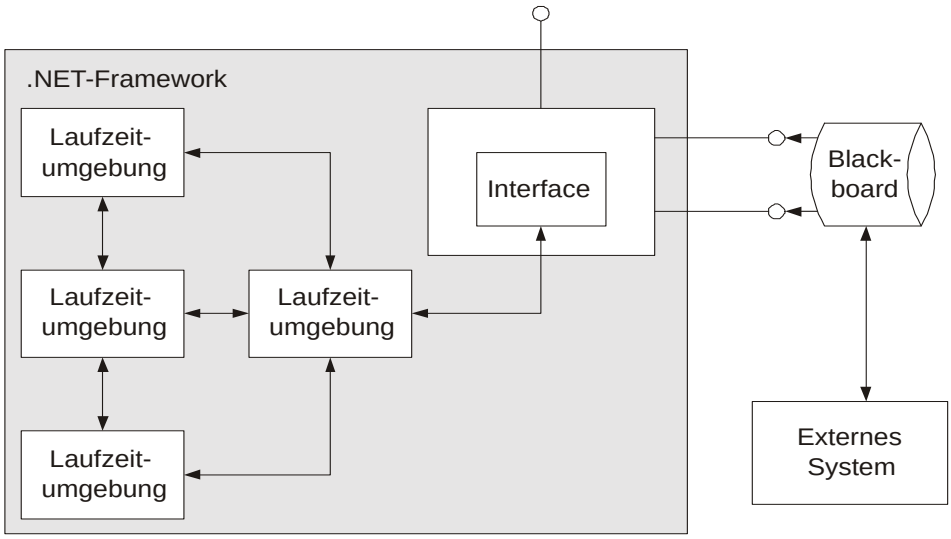

Abbildung 7.5: Datenversorgung durch Nachbarsysteme

7.1.5.4 Realisierung von Kommunikation und Koordination

Die für verteilte hierarchische Steuerungssysteme typischen Vorgabe- und Rückkopplungszyklen zwischen den stratifizierten Entscheidungseinheiten und die Koordinationsaktivitäten zwischen den gestaffelten Entscheidungseinheiten erfordern eine rechnerübergreifende Kommunikation auf Basis von Middleware.

Aufgrund der Aufteilung der Steuerungsfunktionalität auf mehrere Rechner wird eine nachrichtenbasierte Kommunikation realisiert. Dazu ist in einem ersten Schritt die Festlegung einer Ontologie erforderlich, die eine für alle Kommunikationspartner verfügbare Interpretationsvorschrift für die kommunizierten Daten zur Verfügung stellt. Der Entwurf einer domänendedizierten Ontologie für die Domäne der Fertigung integrierter Schaltkreise wird in [20] beschrieben. Die Ontologie enthält Domänenkonzepte wie beispielsweise Ressourcen in unterschiedlichem Aggregationsgrad, Produkte, Pläne für terminierte Lose und Ablaufpläne. Neben den Domänenkonzepten werden Agentenaktivitäten und Prädikate in die

Ontologie

Ontologie aufgenommen.

Contentsprache

Auf semantischer Ebene ist die Entwicklung einer Contentsprache erforderlich, die semantisch sinnvolle Kommunikationsinhalte unter Verwendung des durch die Ontologie bereitgestellten Vokabulars beschreibt. Eine Contentsprache wird in [21] beschrieben. Die grundlegende Idee dieser Sprache besteht darin, eine kontextfreie Grammatik zu entwickeln. Die Grammatik verwendet die in XML-Objekte konvertierten Instanzen der Ontologieklassen. Innerhalb von ManufAg wurde ein spezielles Codec-Element zur Unterstützung der Contentsprache von FABMAS entwickelt (vergleiche dazu die Ausführungen in Abschnitt 5.4.7.2).

Im letzten Schritt muss das dynamische Verhalten der an der Kommunikation beteiligten Einheiten durch Interaktionsprotokolle beschrieben werden. Dazu definieren wir den Begriff „Koordinationssprache“ wie folgt:

Koordinationssprache

Definition 7.1.3 (Koordinationssprache) *Das Tripel*

$$\langle O, C, IP \rangle, \tag{7.1}$$

bestehend aus einer Ontologie O, einer Contentsprache C und der Menge von Interaktionsprotokollen IP für die Einheiten des Steuerungssystems, wird als Koordinationssprache bezeichnet. Als Interaktionsprotokoll wird eine Menge von auf Sprechakten beruhenden Konversationen bezeichnet, die in einer wohldefinierten Ordnung durchlaufen werden. Jeder Sprechakt besitzt einen Contentausdruck, der entsprechend der Grammatik aufgebaut ist.

Wir weisen darauf hin, dass das Kontraktnetz als spezielles Interaktionsprotokoll bereits in Abschnitt 3.4.3.2 eingeführt wurde.

Mit Hilfe der Koordinationssprache können die Entscheideragenten unter Verwendung der Monitoringagenten Ausnahmesituationen erkennen und ein geeignetes verteiltes rollierendes Ablaufplanungsvorgehen realisieren.

Übungsaufgabe 7.3 (Interaktionsprotokoll) *Entwerfen Sie ein Interaktionsprotokoll für einen Produktionsbereichsagenten. Der Agent wird vom Produktionssystemagenten aufgefordert, einen Ablaufplan zu berechnen. Nach der Berechnung geeigneter Parameter wird der entsprechende Schedulingagent damit beauftragt. Der Ablaufplan wird an den Produktionssystemagenten übermittelt. Der Produktionsbereichsagent wartet auf Informationen über den Status des Produktionsbereichs (vergleiche dazu die Methode ErmittleKritischenPB() in Abbildung 7.1).*

Übungsaufgabe 7.4 (Visualisierung von Interaktionsprotokollen) *Überlegen Sie, welche UML-Diagrammarten geeignet sind, Interaktionsprotokolle zu visualisieren. Visualisieren Sie damit das Interaktionsprotokoll aus Aufgabe 7.3.*

7.1.6 Architektur zur Leistungsbewertung des Produktionssteuerungsansatzes

Leistungsbewertung

Eine generische Architektur für eine Leistungsbewertung des Rahmenwerks durch simulationsbasierte Emulation des zu steuernden Produktionsprozesses wurde in [19] vorgeschlagen. Kernidee dieses Ansatzes ist die Verwendung eines diskreten ereignisorientierten Simulationsmodells zur Identifikation des zu steuernden Produktionsprozesses.

Ein derartiges Vorgehen hat den Vorteil, dass zum einen das Basissystem des zu steuernden Produktionssystems in hinreichendem Detaillierungsgrad modelliert werden kann. Andererseits können auch dynamische Aspekte des Produktionsprozesses, die im Wesentlichen durch Arbeitspläne beschrieben werden, gut modelliert werden. Das stochastische Verhalten des Produktionssystems kann in die Leistungsbewertung einbezogen werden. Zur Realisierung des Ansatzes zur Leistungsbewertung wird eine objektorientierte blackboardartige Datenschicht in der Programmiersprache C++ implementiert, die alle relevanten Entitäten des Produktionssystems im Hauptspeicher des Computers vorhält (siehe Abbildung 7.4 und 7.5).

Existierende Produktionssteuerungsverfahren wie Prioritätsregeln (vergleiche dazu die Ausführungen in Abschnitt 6.1.2.3) können so zu Vergleichszwecken softwaretechnisch effizient in die Leistungsbewertung einbezogen werden.

7.1.7 Interpretation von FABMAS innerhalb von BCA und des Kern-Schalen-Modells

Kern-Schalen-Modell

Das beschriebene Rahmenwerk kann innerhalb des Kern-Schalen-Modells [15, 12] interpretiert werden (vergleiche dazu die Ausführungen in Abschnitt 5.3.5). Das Kern-Schalen-Modell unterscheidet zwischen allgemeinen und unternehmensspezifischen Funktionsclustern. Die Anwendung des Kern-Schalen-Modells auf das Rahmenwerk ist in Abbildung 7.6 dargestellt. Außerdem soll gezeigt werden, wie das vorgeschlagene Rahmenwerk zur Erweiterung von MES-Funktionalität verwendet werden kann.

Die gepunktete Linie wird in Abbildung 7.6 dazu verwendet, das Kern-Schalen-Modell in zwei Teile zu zerlegen. Der untere Teil bildet das MES ab, während der obere Teil das Rahmenwerk darstellt. Die innere Schale bildet den technischen Kern des Anwendungssystems. Das MES ermöglicht den Zugang zu relationalen Datenbanken und dem Betriebssystem. Auf der Ebene des Rahmenwerks werden das .NET-Remoting-Framework und die DCOM-Technologie zur Implementierung von Kommunikationsmöglichkeiten zwischen den unterschiedlichen Agenten verwendet. In der Sprache der BCA-Architektur (vergleiche hierzu die Ausführungen in Abschnitt 2.6.3) wird das als Komponenten-System-Rahmenwerk bezeichnet.

Die innere Schale wird zur Datenhaltung verwendet. Geeignete Datenstruktu-

ren werden innerhalb des Datenmodells im MES angelegt, um Lose, Maschinen, Arbeitspläne und die entsprechenden Zustandsdaten zu speichern. Auf der Ebene des Rahmenwerks findet die in Abschnitt 7.1.6 beschriebene blackboardartige Datenschicht Verwendung, um ähnliche Daten wie im Datenmodell des MES vorzuhalten. Diese Schale enthält immer noch relativ generische Funktionalität. Im Sinne des Kern-Schalen-Modells wird durch diese Schicht Basisfunktionalität zur Verfügung gestellt.

Die nächste Schale ist durch die Maschinenbelegungsfunktionalität des MES vorgegeben. Ablaufplanungsfunktionalität wird durch die Entscheider- und Dienstagenten des Rahmenwerks zur Verfügung gestellt. Diese Schale ist bereits in gewissem Maße unternehmensspezifisch.

Konkrete Ablaufplanungsalgorithmen sind in der äußeren Schale implementiert. Die Algorithmen sind dabei typischerweise durch Rechnenkerne wie ILOG [11] oder objektorientierte Rahmenwerke wie GaLib [38] vorgegeben. Im Sinne von BCA liegen Anwendungsrahmenwerke vor.

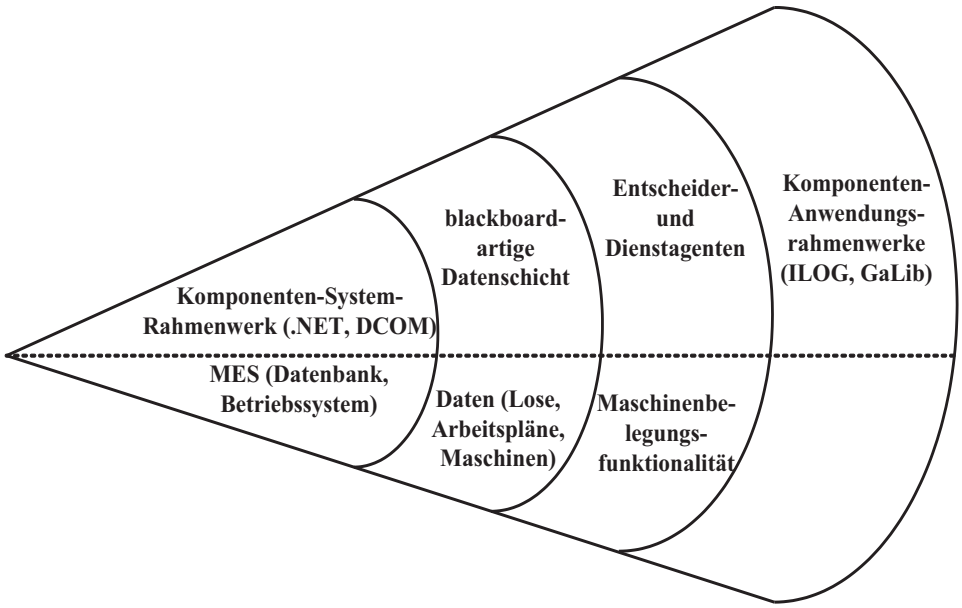

Abbildung 7.6: Interpretation von FABMAS innerhalb des Kern-Schalen-Modells

7.2 Prototyp einer SOA für die Produktionsdomäne

7.2.1 Motivation und Problemstellung

In der wissenschaftlichen Literatur gibt es nur relativ wenige Arbeiten, die sich mit der Realisierung einer SOA in der Praxis beschäftigen. Diese Arbeiten be-

ziehen sich hauptsächlich auf Anwedungen im Dienstleistungsbereich und nicht in der Produktion. Wir betrachten dazu das folgende Beispiel.

Beispiel 7.2.1 (SOA-Anwendungen) *In [8] wird die Realisierung einer SOA bei der Deutschen Post AG beschrieben. Eine Fallstudie aus dem Bankenbereich wird in [37] vorgestellt.*

Die Anwendung von Webservices in zukünftigen Produktionssystemen wird auf einem rein konzeptuellen Niveau in [4] dargestellt. Das Ziel dieser Fallstudie besteht darin, konkrete Dienste für die Produktionsdomäne auszuwählen und dann prototypisch zu implementieren. Insbesondere ist es auch notwendig zu untersuchen, wie innerhalb der SOA auf Stamm- und Bewegungsdaten in ERP-Systemen zugegriffen werden kann.

7.2.2 Identifizierung von Diensten

Diensteidentifizierung

Dienste für den Prototyp werden aufgrund ihrer Verwendung in verschiedenen Situationen und Kontexten identifiziert. Wie bereits aus Abschnitt 4.4 bekannt ist, kann dieser Ansatz aber nur ein erster Schritt bei der Identifizierung geeigneter Dienste sein.

Der erste Dienst beschäftigt sich mit Losplanungsansätzen, die eine Grobplanung der Lose in einem Produktionssystem ermöglichen. Wir unterscheiden (vergleiche dazu die Ausführungen in Abschnitt 6.1.2.3) zwischen:

- Vorwärtsterminierung,
- Rückwärtsterminierung,
- kombinierter Vorwärts-/Rückwärtsterminierung.

Vorwärtsterminierung bedeutet, dass wir geplante Fertigstellungstermine für jeden Arbeitsgang eines Loses unter Verwendung der rekursiven Beziehung

$$c_{jk} := \begin{cases} r_j + (1+h)p_{jk}, & \text{falls } k = 1 \\ c_{jk-1} + (1+h)p_{jk}, & \text{sonst} \end{cases} \tag{7.2}$$

berechnen. Wir bezeichnen dabei mit r_j den Einsteuertermin von Los j. Die Bezeichnung c_{jk} wird für den geplanten Fertigstellungstermin für Arbeitsgang k von Los j verwendet. Die Bearbeitungszeit von Arbeitsgang k für Los j wird mit p_{jk} bezeichnet. Die Größe $h \geq 0$ ist ein Faktor, der zur Abschätzung der Wartezeit auf die Bearbeitung von Arbeitsgang k verwendet wird.

Bei einer Rückwärtsterminierung gehen wir ähnlich vor. Ausgehend vom geplanten Fertigstellungstermin d_j des Loses berechnen wir Starttermine für die einzelnen Arbeitsgänge des Loses. Eine kombinierte Vorwärts-/Rückwärtsterminierung verwendet zuerst eine Rückwärts- und dann eine Vorwärtsplanung, um zu Zeitpuffern zu gelangen, in denen die Arbeitsgänge begonnen werden müssen. Die kombinierte Vorwärts-/Rückwärtsterminierung ist als

Dienst interessant, da sie durch Nacheinanderausführung eines Rückwärts- und eines Vorwärtsplanungsdienstes implementiert werden kann.

Der zweite Dienst dient der Bedarfsplanung. Wir verwenden die Methode der gleitenden Durchschnitte und die exponentielle Glättung. Die Methode der gleitenden Durchschnitte basiert auf der Idee, dass auf Basis der historischen Bedarfe $d_{t-n+1}, \ldots, d_t$ der Bedarf $f(d_{t+1})$ für die zukünftige Periode $t+1$ durch die folgende Beziehung

$$f(d_{t+1}) := \frac{1}{n} \sum_{k=t-n+1}^{t} d_k \tag{7.3}$$

gegeben ist. Die Methode der exponentiellen Glättung funktioniert ähnlich. Der zukünftige Bedarf wird dabei durch

$$f(d_{t+1}) := \alpha d_t + (1-\alpha) f(d_t) \tag{7.4}$$

bestimmt, wobei $0 \leq \alpha \leq 1$ einen Glättungsparameter darstellt, der festlegt, wie stark historische Bedarfe in die Bedarfsvorhersage eingehen. Mit $f(d_t)$ bezeichnen wir einen Bedarfsvorhersagewert für die Periode t.

Die beiden Dienste benötigen Stamm- und Bewegungsdaten des Produktionsprozesses. Aus diesem Grund ist es notwendig, auf ein ERP-System, in unserem Fall mySAP ERP, zuzugreifen. Die identifizierten Dienste sind in Abbildung 7.7 in Form eines Use-Case-Diagramms gezeigt.

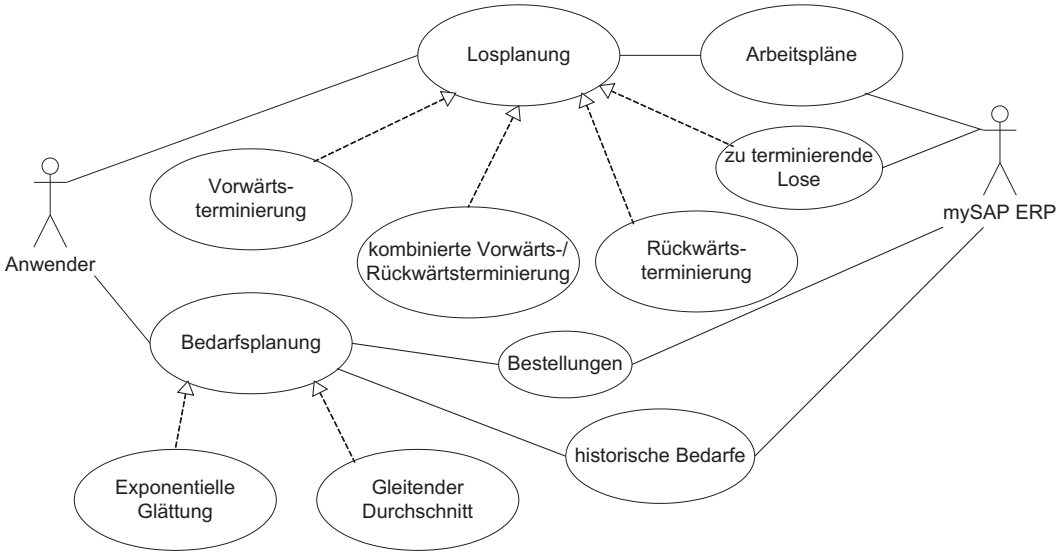

Abbildung 7.7: Use-Case-Diagramm für die identifizierten Dienste

7.2.3 Architektur des Prototyps

7.2.3.1 Übersicht

Prototyp-Architektur

Der Prototyp umfasst mehrere Softwaresysteme:

- eine Clientanwendung, die den Anwendern eine graphische Benutzeroberfläche zur Verfügung stellt und einen Zugriff auf die Webservices ermöglicht,
- Webservices, die zur Implementierung der Dienste der SOA dienen,
- einen BPEL-Business-Process-Manager zur Orchestrierung der unterschiedlichen Dienste,
- ein mySAP ERP-System, das die für die Produktionsplanung notwendigen Daten vorhält.

Die Webservices werden auf einem ASP .NET-Entwicklungsserver ausgeführt. Der BPEL-Process-Manager läuft auf einem J2EE-Anwendungsserver. Die BPEL-Engine des Oracle-BPEL-Process-Managers [26] wird verwendet.

Die Architektur des Prototyps ist in Abbildung 7.8 dargestellt.

7.2.3.2 Zugriff auf das ERP-System

Die Dienste benötigen Daten aus einem ERP-System. Wir benötigen:

- Arbeitspläne für die einzelnen Endprodukte (vergleiche dazu die Ausführungen in Abschnitt 6.1.2.2),
- die Menge der bereits in das Produktionssystem eingelasteten Lose,
- den aktuellen Arbeitsgang der einzelnen Lose,
- die geplanten Einlastungstermine für Lose, die zu prognostizierten Produktionsaufträgen gehören,

für die Losplanung. Für die Bedarfsplanung werden die nachfolgend aufgezählten Daten gebraucht:

- historische Bedarfe für eine bestimmte Anzahl von Vergangenheitsperioden,
- bereits eingegangene Bestellungen.

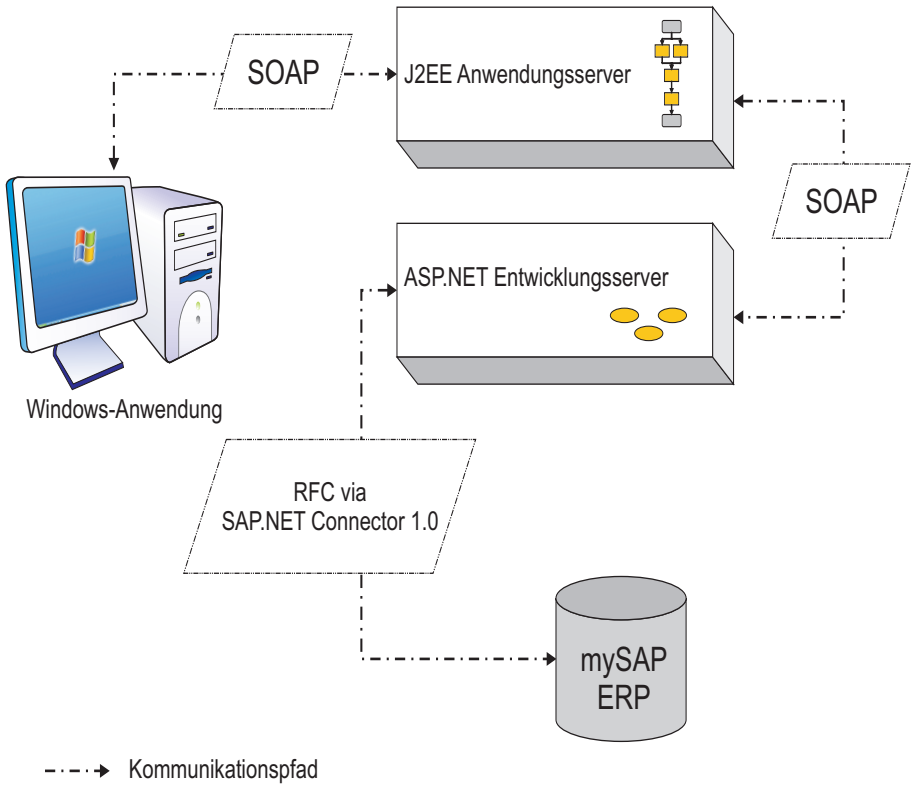

Abbildung 7.8: Architektur des Prototyps

Um auf die Daten innerhalb eines mySAP ERP-Systems zuzugreifen, wird der SAP Connector für Microsoft .NET 1.0 verwendet (vergleiche dazu [32]). Dieser Connector erzeugt C#-Proxy-Klassen für Datenstrukturen in mySAP ERR, die innerhalb von C#-Anwendungen verwendet werden können.

Ein ABAP-Funktionsbaustein wird dazu implementiert, der RFC-fähig ist (vergleiche die Ausführungen in Abschnitt 5.3.9.3). Gleichzeitig wird ein Webservice bereitgestellt, der die Daten aus dem mySAP ERP-System ausliest. Dieser Webservice ruft den Funktionsbaustein unter Verwendung des SAP-RFC-Protokolls auf. Für diese Kommunikation wird der SAP-Connector verwendet. Die Proxy-Klasse dient dazu, die im Data-Dictionary von mySAP ERP vorgehaltenen Tabelleninhalte der C#-Anwendung zugänglich zu machen. Die Tabelle enthält den Arbeitsplan und Losdaten. Wir erhalten den Tabelleninhalt als Ergebnis des Funktionsbausteinaufrufes.

SAP-Connector

7.2.3.3 Implementierung und Orchestrierung der Dienste

Die identifizierten Dienste werden als Webservices in C# unter Verwendung des .NET-Rahmenwerks (vergleiche dazu die Ausführungen in Abschnitt 4.2.5) implementiert. Die folgenden vier Dienste sind umgesetzt:

- LotPlanning,
- DemandPlanning,
- GetProductionData,
- ProductionPlanningSync.

Der Webservice „LotPlanning“ umfasst die folgenden drei Operationen:

- ForwardPlanning,
- BackwardPlanning,
- Forward/BackwardPlanning.

Als Ergebnis der Ausführung der Operationen erhält man Datenstrukturen, die Start- und geplante Fertigstellungstermine für die einzelnen Arbeitsgänge der betrachteten Lose zur Verfügung stellen.

Der Webservice „DemandPlanning“ bietet die folgenden Operationen an:

- MovingAverage,
- ExponentialSmoothing.

Dieser Webservice erzeugt Datenstrukturen, welche die Bedarfsvorhersagewerte enthalten.

Der dritte Webservice „GetProductionData“ dient der Datenbeschaffung aus dem mySAP ERP-System. Die Verbindung zum ERP-System wird aufgebaut. Der entsprechende Funktionsbaustein wird aufgerufen. Die entsprechenden Daten aus dem mySAP ERP-System sind das Ergebnis dieses Webservices.

Orchestrierung

Die drei Webservices sind geeignet zu orchestrieren (vergleiche die Ausführungen in Abschnitt 2.3.4.3 zum Orchestrierungsbegriff). Dafür wird BPEL verwendet. Der erstellte BPEL-Prozess stellt sicher, dass sich die drei Webservices logisch und zeitlich korrekt verhalten. Der erhaltene BPEL-Prozess wird durch den Webservice „ProductionPlanningSync“ ausgeführt. Die drei Webservices und ihre Orchestrierung werden in Abbildung 7.9 dargestellt.

Die unterschiedlichen Webservices werden durch partnerLinks in den BPEL-Prozess eingebunden. Der BPEL-Prozess enthält die Teilprozesse „LotPlanning“ und „DemandPlanning“. Der Teilprozess „LotPlanning“ ruft zuerst den Webservice „GetProductionData“ auf. Anschließend wird eine Vorwärtsterminierung unter Verwendung der Operation „ForwardPlanning“ durchgeführt. Unter Verwendung von Beziehung (7.2) können wir den erwarteten Fertigstellungstermin c_j mit dem geplanten Fertigstellungstermin d_j vergleichen. Das Los j ist verspätet, wenn $d_j < c_j$ gilt. In diesem Fall führen wir eine Rückwärtsterminierung unter Verwendung der Operation „BackwardPlanning“ durch, um einen neuen Einsteuertermin r_j zu ermitteln.

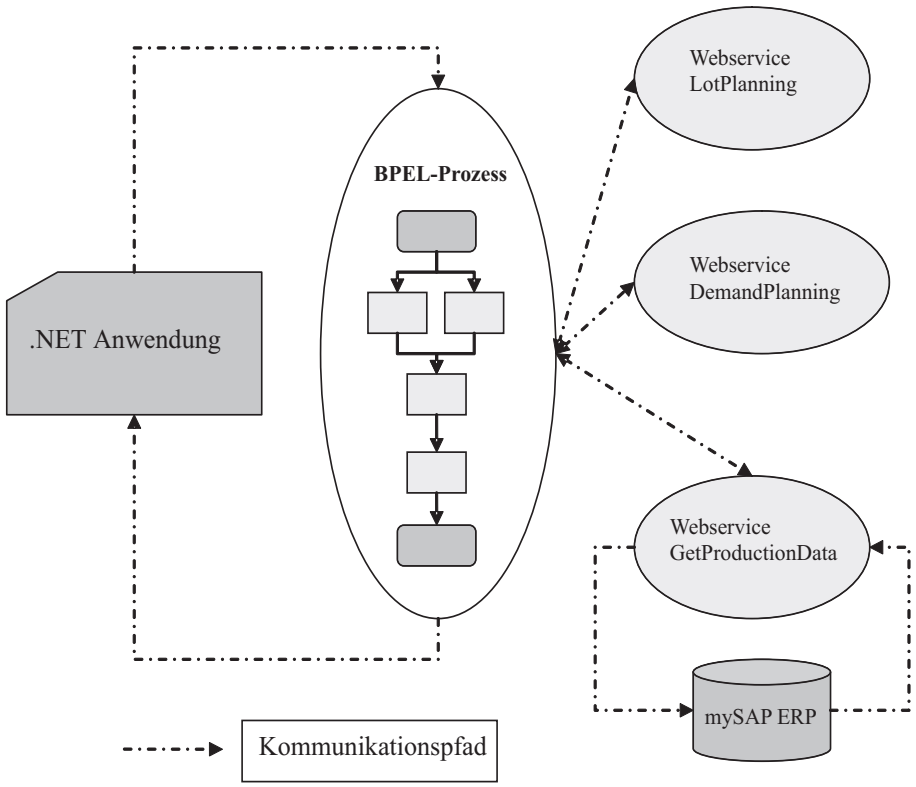

Abbildung 7.9: Dienste des Prototyps und ihre Orchestrierung

Falls $c_j \leq d_j$ gilt, ist das Los nicht verspätet. Dann ist es sinnvoll, eine kombinierte Vorwärts-/Rückwärtsterminierung durchzuführen, um Zeitpuffer für die Arbeitsgänge zu ermitteln. Außerhalb des Labors sind jetzt die Planungsergebnisse noch an das ERP-System zu übertragen. Der Teilprozess „LotPlanning“ ist in Abbildung 7.10 gezeigt.

Auf die Details zum Teilprozess „DemandPlanning“ verzichten wir an dieser Stelle.

7.2.3.4 Client-Anwendung

Die Client-Anwendung dient dazu, den BPEL-Prozess, der durch den Webservice „ProductionPlanningSync“ gegeben ist, unter Verwendung der WSDL-Datei des Webservices (vergleiche Abschnitt 4.3.2.2) einzubinden. Die Client-Anwendung enthält eine graphische Benutzeroberfläche, die drei unterschiedliche Sichten umfasst:

- Parametrisierung des BPEL-Prozesses,
- „LotPlanning“,
- „DemandPlanning“.

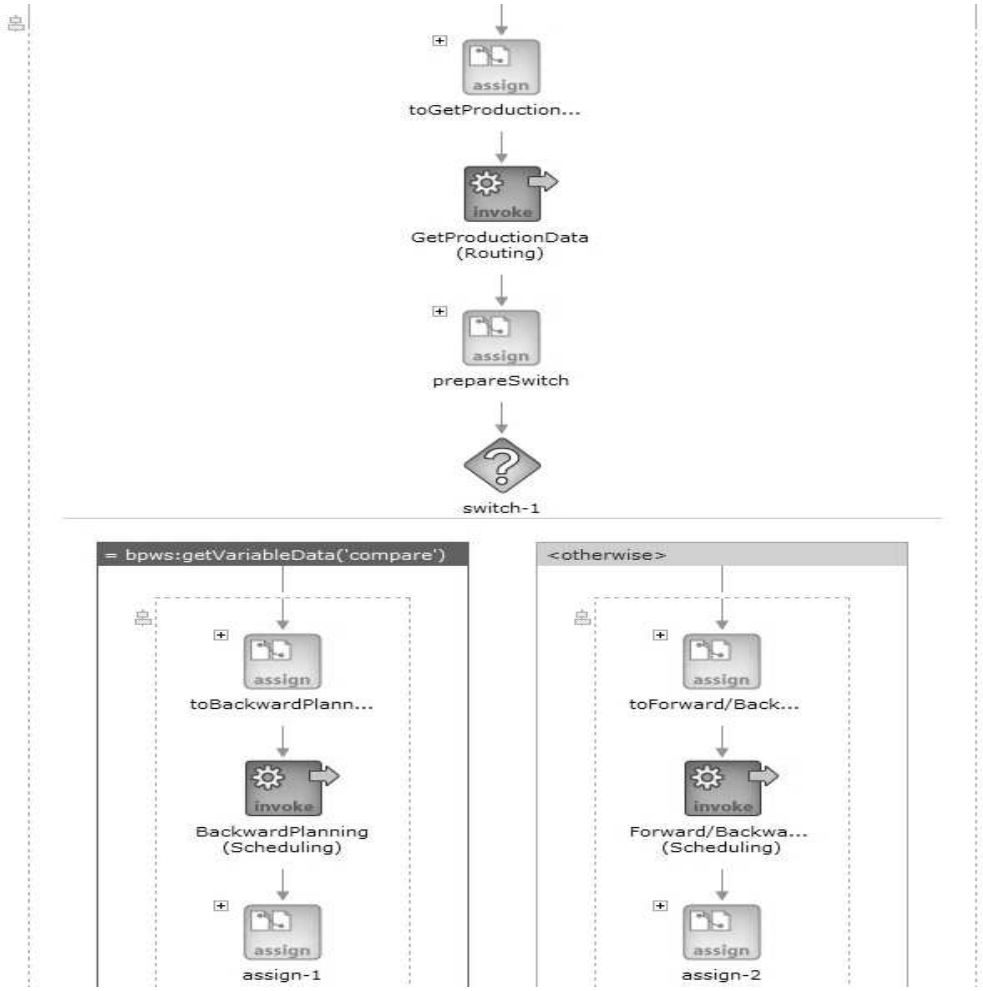

Abbildung 7.10: BPEL-Teilprozess Losplanung

Bei der Parametrisierung des BPEL-Prozesses werden Default-Werte für „Lot-Planning“ und „DemandPlanning“ gesetzt. Die Client-Anwendung ruft den BPEL-Prozess auf und zeigt dessen Ergebnisse an. Die Client-Anwendung ist ebenfalls in C# implementiert.

7.2.3.5 Gesammelte Erfahrungen

Die folgenden Erfahrungen konnten während der Entwicklung des Prototyps gemacht werden: Erkenntnisse

- Webservices können mit den verfügbaren Entwicklungswerkzeugen leicht erstellt werden.
- Es ist ohne Schwierigkeiten möglich, Daten aus existierenden ERP-Systemen innerhalb der SOA zu verwenden und umgekehrt Daten, die

innerhalb einer SOA ermittelt werden, in die ERP-Systeme zurückzuschreiben. Technisch wird das durch Adapter der ERP-Hersteller ermöglicht.

- Implementierungsdetails können gut in Webservices gekapselt werden.

Die folgenden Grenzen des gegenwärtig anzutreffenden SOA-Konzeptes wurden sichtbar:

- Die Identifizierung von Diensten ist noch nicht ausreichend unterstützt (vergleiche dazu auch die Ausführungen in Abschnitt 4.4).
- Unterschiedliche kommerziell verfügbare BPEL-Engines wurden getestet. Viele sind nicht stabil und laufen langsam.
- Anwender ohne eine sehr gute IT-Ausbildung sind nicht in der Lage, unter Verwendung einer BPEL-Engine bereits existierende Dienste zu komplexeren Diensten zu kombinieren.
- Der Test und die Überprüfung der Korrektheit der Funktionalität einer SOA ist schwierig.

Forschungsbedarf

Wie bereits in Abschnitt 4.4 in anderem Kontext diskutiert, sind im SOA-Umfeld noch viele Fragestellungen unter Verwendung wissenschaftlicher Methoden zu klären. Von der Beantwortung dieser Fragestellungen wird unter anderem abhängen, ob das SOA-Konzept die IT-Landschaft in Unternehmen nachhaltig verändern wird.

Lösungen zu den Übungsaufgaben

Übungsaufgabe 7.1
Ein allgemeines Rahmenwerk wird mit dem Ziel entworfen, eine bestimmte Betrachtungsweise auf eine zu lösende Problemklasse zu unterstützen. Auch Software-Rahmenwerke haben diesen Anspruch. Sie bieten eine generische (Software-)Lösung für ähnliche Probleme in einem bestimmten Kontext. Beim allgemeinen Rahmenwerk erfolgt das unter Verwendung von Modellen und Konzepten. Die Modelle und Konzepte sind beim Software-Rahmenwerk Klassen und das Zusammenspiel der einzelnen Klassen bzw. Komponenten. Es wird zudem der Kontrollfluss für die Anwendung vorgegeben.

Übungsaufgabe 7.2
Multi-Agenten-Systeme bieten die Möglichkeit, komplexe Probleme, die verteilbar sind, zu lösen. Dazu lösen einzelne Agenten des Multi-Agenten-Systems einen Teil des Gesamtproblems. Über die Interaktion der Agenten, vor allem die Kommunikation der Agenten untereinander, erfolgt dann unter Verwendung der Lösungen der Teilprobleme die Lösung des Gesamtproblems. Die Kommunikation der Agenten untereinander wird durch die Infrastruktur des Multi-Agenten-Systems ermöglicht. Auch Hierarchien lassen sich in Multi-Agenten-Systemen abbilden. Für die Lösung der Teilprobleme können OR- und KI-Verfahren eingesetzt werden. Durch die Hierarchien ist eine zeitliche und räumliche Dekomposition des Gesamtproblems und der Teilprobleme möglich.

Übungsaufgabe 7.3
Das Interaktionsprotokoll zwischen dem Produktionssystemagenten, dem Produktionsbereichsagenten und dem Schedulingagenten kann wie folgt dargestellt werden.

Produktionssystemagent :
Sende an Produktionsbereichsagenten: „Berechne Ablaufplan“
Produktionsbereichsagent :
Empfange Nachricht „Berechne Ablaufplan“ des Produktionssystemagenten
Berechne geeignete Parameter
Sende an Schedulingagent: „Berechne Ablaufplan“
Schedulingagent :
Empfange Nachricht „Berechne Ablaufplan“ des Produktionsbereichsagenten
Berechne Ablaufplan
Sende an Produktionssystemagent: Ablaufplan
Produktionssystemagent :
Empfange Ablaufplan des Schedulingagenten
Ermittle kritischen Produktionsbereich
Sende an Produktionsbereichsagenten: Informationen über kritischen Produk-

tionsbereich
Produktionsbereichsagent :
Empfange Informationen über kritischen Produktionsbereich

Übungsaufgabe 7.4
Interaktionsprotokolle lassen sich durch Sequenzdiagramme visualisieren. Das Interaktionsprotokoll für die Bereichsagenten ist in Abbildung 7.11 dargestellt.

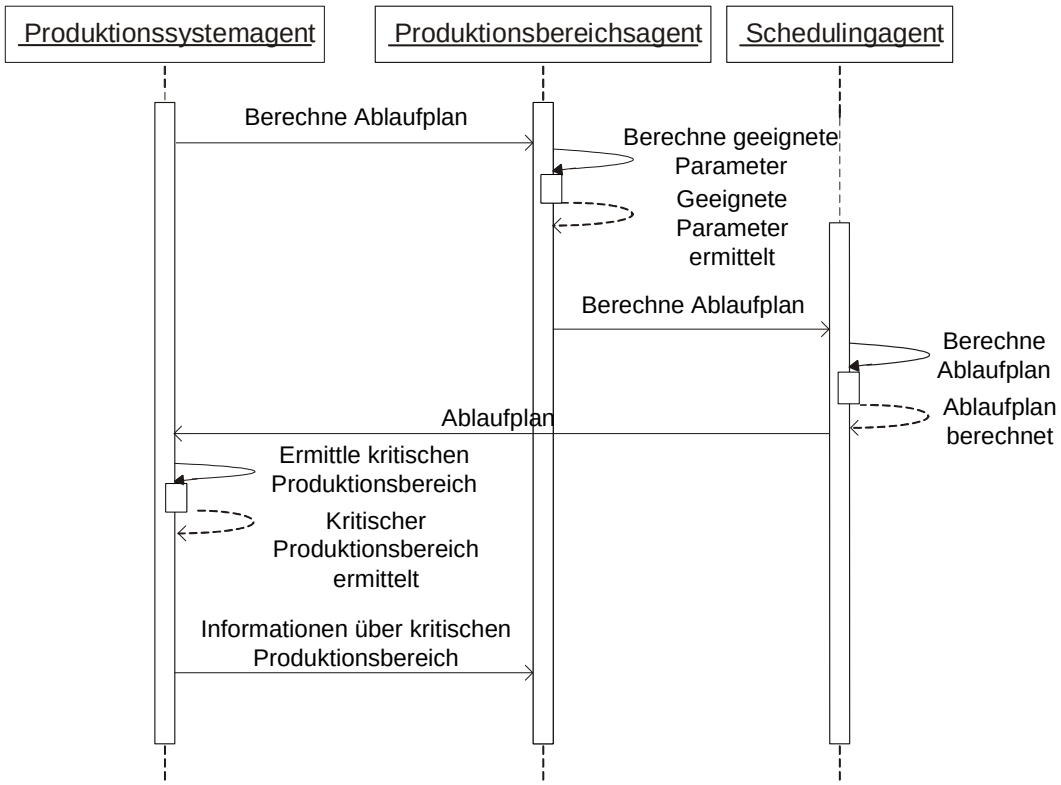

Abbildung 7.11: Interaktionsprotokoll für Produktionsbereichsagenten

Literatur

[1] Adexa. http://www.adexa.com/, 2006.

[2] M. Caridi; S. Cavalieri. Multi-agent systems in production planning and control: An overview. *Production Planning and Control*, 15(2), 106–108, 2004.

[3] H. Corsten; R. Gössinger. Advanced planning systems - Anspruch und Wirklichkeit. *PPS Management*, 6, 32–39, 2001.

[4] W. A. Estrem. An evaluation framework for deploying web services in the next generation manufacturing enterprise. *Robotics and Computer Integrated Manufacturing*, 19, 509–519, 2003.

[5] L. F. Gelders; L. Van Wasserhove. Hierarchical integration in production planning: Theory and practice. *Journal of Operations Management*, 3, 27–35, 1982.

[6] H. Gferer; G. Zäpfel. Hierarchical model for production planning in the case of uncertain demand. *European Journal of Operational Research*, 86, 142–161, 1995.

[7] K.C. Hadavi. ReDS: A Real Time Production Scheduling System from Conception to Practice. M. Zweben; M. Fox (Hrsg.), *Intelligent Scheduling*, Morgan Kaufmann, San Francisco, CA, 581–604, 1994.

[8] M. Herr; U. Bath; A. Koschela. Implementation of a service oriented architecture at deutsche post mail. *Proceedings ECOWS*, Erfurt, 227–238, 2004.

[9] Karl G. Kempf. Intelligently scheduling semiconductor wafer fabrication. M. Zweben; M. S. Fox (Hrsg.), *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 517–544, 1994.

[10] S. Kreipl; M. Pinedo. Planning and scheduling in supply chains: An overview of issues in practice. *Production and Operations Management*, 13(1), 77–92, 2004.

[11] C. Le Pape. Implementation of resource constraints in ILOG SCHEDULE: A library for the development of constraint - based scheduling systems. *Intelligent Systems Engineering*, 3(2), 55–66, 1994.

[12] M. Lohmann; B. Schmitzer; P. Mertens. Kern-Schalen-Modell mit Fokus auf E-Commerce. K. Turowski (Hrsg.), *Proceedings 3. Workshop „Komponentenorientierte betriebliche Anwendungssysteme“*, Frankfurt am Main, 23–38, 2001.

[13] S. Mason; J. W. Fowler; M. W. Carlyle. A modified shifting bottleneck heuristic for minimizing thetotal weighted tardiness in a semiconductor wafer fab. *Journal of Scheduling*, 5(3), 247–262, 2002.

[14] P. Mertens. Zufriedenheit ist die Feindin des Fortschritts - ein Blick auf das Fach Wirtschaftsinformatik. Arbeitspapier, Universität Erlangen-Nürnberg, Fachbereich Wirtschaftsinformatik I, 2004. Erweiterte Fassung eines Vortrags anlässlich des 15-jährigen Jubiläums des Instituts für Wirtschaftsinformatik der Universität St. Gallen.

[15] P. Mertens; M. Braun; A. Engelhardt; J. Holzner; T. Kaufmann; H. Ließmann; P. Ludwig; S. Möhle. Formen integrierter betrieblicher Anwendungssysteme zwischen Individual- und Standardsoftware: Erfahrungen und Zwischenergebnisse bei Experimenten mit branchen- und betriebstyporientierten Anwendungsarchitekturen. FORWISS Report, Bayerisches Forschungszentrum für Wissensbasierte Systeme, Universität Erlangen-Nürnberg, Erlangen, 1997.

[16] M.D. Mesarović; D. Macko; Y. Takahara. *Theory of Hierarchical, Multilevel, Systems*. Academic Press, New York, London, 1970.

[17] M.D. Mesarović; Y. Takahara. *Abstract Systems Theory*. Lecture Notes in Control and Information Sciences. Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo, 1989.

[18] L. Mönch. Analyse und Design für ein agentenbasiertes System zur Steuerung von Produktionsprozessen in der Halbleiterindustrie. *Proceedings Verbundtagung: Verteilte Informationssysteme auf der Grundlage von Objekten, Komponenten und Agenten (vertIS 2001)*, Bamberg, 99–112, 2001.

[19] L. Mönch; M. Stehli. Ein simulationsbasierter Ansatz zur Leistungsbewertung von Systemen der verteilten künstlichen Intelligenz zur Produktionssteuerung. *Künstliche Intelligenz*, 4, 43–49, 2003.

[20] L. Mönch; M. Stehli. An ontology for production control of semiconductor manufacturing processes. *Proceedings First German Conference on Multi Agent Technologies (MATES 2003)*, LNAI 2831, Springer, Erfurt, 156–167, 2003.

[21] L. Mönch; M. Stehli. A content language for a hierarchically organized multi-agent-system for production control. *Proceedings Multi-Konferenz Wirtschaftsinformatik, Teilkonferenz "Coordination and Agent Technology in Value Networks"*, Essen, 197–212, 2004.

[22] L. Mönch; M. Stehli; R. Schulz. An agent-based architecture for solving dynamic resource allocation problems in manufacturing. *Proceedings 14-th European Simulation Symposium (ESS 2002)*, Dresden, 331–337, 2002.

[23] L. Mönch; M. Stehli; J. Zimmermann. FABMAS - an Agent Based System for Semiconductor Manufacturing Processes. *Proceedings First International Conference on Industrial Application of Holonic and Multi-Agent Systems (HoloMas 2003)*, LNAI 2744, Springer, Prague, Czech Republic, 258–267, 2003.

[24] L. Mönch. *Agentenbasierte Produktionssteuerung komplexer Produktionssysteme*. Gabler, DUV-Verlag, Wiesbaden, 2006.

[25] U. Mussbach-Winter; H.-H. Wiendahl. Was leisten MES-Lösungen heute? - Merkmale ihrer Planungs- und Steuerungskonzepte. *Industrie Management*, 2, 14–18, 2003.

[26] Oracle. *Oracle BPEL Process Manager 10.1.3.3.0*. http://www.oracle.com/technology/products/ias/bpe, 2007.

[27] I. M. Ovacik; R. Uzsoy. *Decomposition Methods for Complex Factory Scheduling Problems*. Kluwer, Norwell, MA, 1997.

[28] H.V.D. Parunak. Applications of distributed artificial intelligence in industry. G. M. P. O'Hare; N. R. Jennings (Hrsg.), *Foundations of Distributed Artificial Intelligence*, Wiley, New York, 1994.

[29] H.V.D. Parunak. A practitioner's review of industrial agent applications. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(4), 389–407, 2000.

[30] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. 2. Auflage. Prentice Hall, Englewood Cliffs, 2001.

[31] M.E. Porter. Towards a dynamic theory of strategy. *Strategic Management Journal*, 12, 95–117, 1991.

[32] SAP. *SAP Connector for Microsoft .NET 1.0*. http://websmo2006-sap-ag.de/connectors, 2006.

[33] C. Schneeweiss. *Distributed Decision Making*. Springer, Berlin, Heidelberg, New York, 2003.

[34] A. Schömig; J. Fowler. Modelling semiconductor manufacturing operations. *Proceedings of the 9th ASIM Dedicated Conference Simulation in Production and Logistics*, Berlin, 55–64, 2000.

[35] N. Srivatsan; S.X. Bai; S.B. Gershwin. Hierarchical real-time integrated scheduling of a semiconductor fabrication facility. *Control and Dynamic Systems*, 61, 197–241, 1994.

[36] F. D. Vargas-Villamil; D. E. Rivera; K. G. Kempf. A hierarchical approach to production control of reentrant semiconductor manufacturing lines. *IEEE Transactions on Control Systems Technology*, 11(4), 578–587, 2003.

[37] R. von Ammon; W. Pausch; M. Schimmer. Realization of sevice-oriented architecture (SOA) using enterprise portal platforms taking the example of multi-channel sales in banking domain. *Proceedings Internationale Tagung Wirtschafsinformatik*, Bamberg, 1503–1518, 2005.

[38] M. Wall. GaLib: A C++ library of genetic algorithm components, 1999. http://lancet.mit.edu/ga/.

[39] R. Winter. *Mehrstufige Produktionsplanung in Abstraktionshierarchien auf der Basis relationaler Informationsstrukturen.* Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[40] M. Zweben; M. S. Fox (Hrsg.). *Intelligent Scheduling.* Morgan Kaufmann, San Francisco, 1994.

Index

V

W