Heft 106

W. Hoffmann, R. Wein, A.-W. Scheer

Konzeption eines Steuerungsmodells für Informationssysteme - Basis für die Real-Time-Erweiterung der EPK (rEPK)

Dezember 1993

Inhaltsverzeichnis

1. Anwendungsgebiete der Real-Time-Modellierung

Der Begriff Real-Time ist traditionell in der Begriffswelt der Informatik und Elektrotechnik anzusiedeln und wurde fast ausschließlich in Verbindung mit technischen Anwendungssystemen (z. B. SPS, Prozeßleitsysteme etc.) verwendet. Eine Zusammenfassung von Real-Time- oder auch Echtzeit-Charakteristika bieten in diesem Zusammenhang Ward und Mellor.[1]

Echtzeit-Systeme führen Unterstützungsfunktionen in einer automatisierten Umgebung aus, deren technische Komponenten als "Sinne" des Systems fungieren. Dementsprechend überlappen sich die kontinuierlichen Eingaben der Sensoren eines Echtzeit-System mit dessen kontinuierlichen Ausgaben. Dagegen sind Nicht-Echtzeit-Systeme eher auf Eingaben beschränkt, die zu diskreten Zeitpunkten erfolgen.

Die parallele Verarbeitung von mehreren auftretenden Ereignissen sowie die nach menschlichen Ermessen in sehr kurzer Zeit durchgeführte Verarbeitung sind ebenfalls Merkmale von Echtzeit-Systemen. Auch die von einem Echtzeit-System angestrebte Genauigkeit der Reaktionszeit ist größer als die durch Nicht-Echtzeit-Systeme geforderte.

Raasch bezeichnet technische Anwendungssysteme im Gegensatz zu kommerziellen Anwendungssystemen als aktive Systeme, die ihre Umgebung kontrollieren, indem sie diese regeln, steuern und aktiv beeinflussen. Innerhalb des Real-Time-Systems wird neben der daten- und zeitlich-kalendergesteuerten Auslösung auch noch die sofortige Aktivierung eines Prozesses benötigt, falls sich ein bestimmter Systemzustand ändert oder ein systeminternes Ereignis eintritt.[2]

Systemintern erkannte Bedingungen resultieren aus einer Kombination von mehreren Ereignissen und deren systeminternen Auswertungen. Dies führt dazu, daß das System seinen Zustand und somit seine Arbeitsweise ändert. Dabei übernehmen Kontrollflüsse die Steuerungsfunktion der zu aktivierenden Prozesse.

Für die weiteren Ausführungen soll der Zustandsbegriff verwendet werden, wie ihn Raasch definiert:

"Jeder Zustand repräsentiert eine Zeitperiode, innerhalb derer das System ein beobachtbares Verhalten zeigt. Der Zustand eines Systems kann sich nur durch äußere Einflüsse ändern. Das System wartet auf ein Ereignis in der Umgebung oder auf eine Aktivität der Umgebung, die das System in einen anderen Zustand versetzt."[3]

Im Gegensatz zu den konventionellen Ansätzen, die sich auf die Betrachtung technischer Anwendungssysteme beschränken, wird die Real-Time-Modellierung im folgenden zur

---

1 vgl. Ward, P. T.; Mellor, S. J.: Strukturierte Systemanalyse von Echtzeitsystemen, München Wien 1991, S. 13-16.

2 vgl. Raasch, J.: Systementwicklung mit strukturierten Methoden, München Wien 1991, S. 199.

3 Raasch, J.: Systementwicklung mit strukturierten Methoden, a.a.O., S. 210.

Beschreibung eines neu entwickelten Ansatzes zur Steuerung von Informationssystemen verwendet. Diese ist nicht nur zur Beschreibung von Echtzeitsystemen geeignet, sondern auch auf beliebige kommerzielle Systeme anwendbar.

2. Aufbau von Informationssystemen

Bei der Analyse heutiger Informationssysteme werden folgende Schwachstellen ersichtlich:

- Starre Abläufe der Prozesse

Die mangelnde Anpassungsfähigkeit von Anwendungssystemen an unternehmensspezifische Prozesse resultiert aus den traditionellen Methoden der Softwareentwicklung. Bei historisch gewachsenen Systemen führt die starre Verknüpfung von Teilmodulen zu festgelegten Abläufen. Dieses Problem läßt sich nur mit einen hohen Aufwand beseitigen.

- Komplexität der Software

Die gegenseitige Abhängigkeite von Modulen führen zu schwer überschaubaren Auswirkungen auf das Gesamtsystems bei Änderung einzelner Teilmodule. Eine ausreichende Entkopplung von Daten und Funktionen ist in der Regel nicht gegeben, da die Datenzugriffe innerhalb der Applikation fest verankert sind.

- Hoher Ressourcenbedarf

Durch die oben aufgeführten Punkte müssen zur Laufzeit eines Anwendungssystems sämtliche Module in den Hauptspeicher geladen werden. Dies führt zu langen Reaktionszeiten des Systems und erzwingt hohe Speicherkapazitäten.

- Unzureichende Dokumentation

Die Entwicklung und damit auch die Dokumentation eines Informationssystems beginnt häufig erst auf der DV-Konzeptebene. Daraus resultiert eine unzureichende Übersichtlichkeit des Gesamtsystems. Auch die inhaltliche Beschreibung der Teilmodelle ist ungenügend.

Folgende Maßnahmen schaffen Abhilfe:

- Modularer Aufbau der Software

Jedes Modul muß so programmiert werden, daß es allein für sich "lebensfähig" ist. Es darf kein Modul von einem anderen Modul direkt aufgerufen werden. Die Daten werden über temporäre Datenstrukturen oder über Datenbanktransaktionen zur Verfügung gestellt.

❒ Trennung der Systemkomponenten

Daten und Funktionen werden entkoppelt. Der Zugriff auf die Daten muß über ein separates, konfigurierbares Transaktionsmodul erfolgen. Der Ablauf der Transaktionen und Prozesse wird von einem separaten, frei konfigurierbaren Steuerungsmodul geregelt.

❒ Verlagerung des Systementwurfs auf Fachkonzeptebene

Für die inhaltliche Beschreibung des Systems existieren auf Fachkonzeptebene formale, grafikbasierte Beschreibungsmethoden, die in DV-Konzepte überführbar sind. Änderungen in der Software sind auf Fachkonzeptebene leichter nachvollziehbar. Mittels geeigneter, toolunterstützter Prozeßmodellierungsmethoden lassen sich die Systeme konfigurieren und ihre Abläufe modifizieren.

Der idealtypische Aufbau eines Informationssystems ist in Abb. 1 beschrieben.

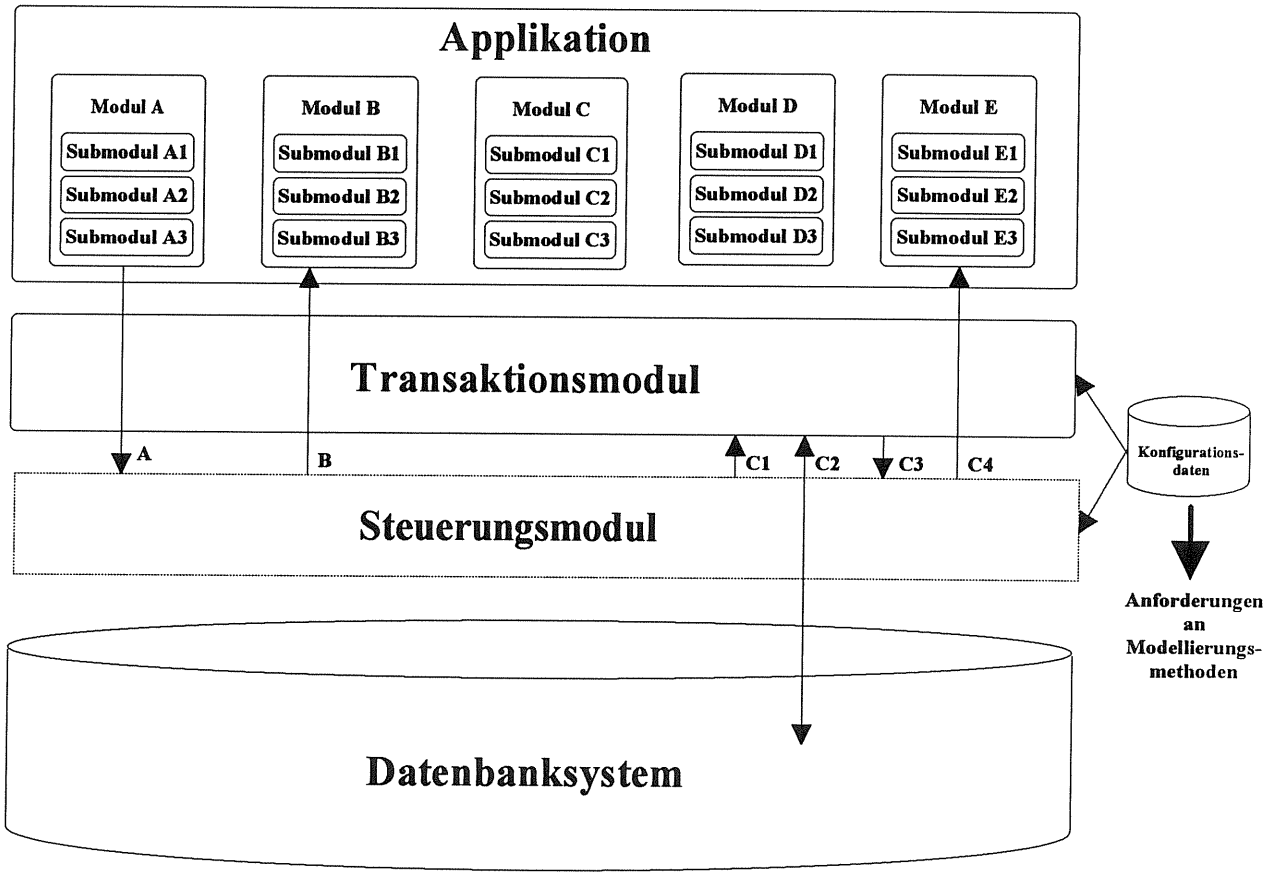

Abb. 1: Aufbau eines Informationssystems

Die Komponenten eines Informationssystems sind:

❒ Applikation

Die Applikationssoftware erfüllt die fachlichen Anforderungen eines Informationssystems. Sie besteht aus voneinander *unabhängigen* Modulen. Die Submodule innerhalb eines Moduls

können miteinander kommunizieren (z. B. Interprozeß-Kommunikation), müssen aber nicht. Die Module sind entkoppelt von der Datenbank.

- Transaktionsmodul

Sämtliche Datenbankzugriffe innerhalb des Informationssystems erfolgen ausschließlich über das Transaktionsmodul. Es ist verantwortlich für die Erhaltung der Datenbankintegrität. Das Transaktionsmodul ist frei konfigurierbar.

- Steuerungsmodul

Die Steuerung der Applikation und des Transaktionsmoduls ist Aufgabe des Steuerungsmoduls. Dieses ist ebenfalls frei konfigurierbar.

- Datenbanksystem

Es ist verantwortlich für die Verwaltung und Speicherung der Daten.

Ein beispielhafter Ablauf kann wie folgt aussehen (s. Abb. 1):
Modul A sendet einen Trigger an das Steuerungsmodul. Dies kann beispielsweise durch Eingaben eines Benutzers an einer Bildschirmmaske und anschließender Datenfreigabe erfolgen (Pfeil A). Im Steuerungsmodul wird die dazugehörende Aktivitätenkette ermittelt und deren Ausführung veranlaßt. Zum einen kann direkt ein Modul gestartet werden (Pfeil B), zum anderen können Datenbankzugriffe erforderlich sein, da ein auszuführendes Modul Informationen aus der Datenbank benötigt. Hierzu aktiviert das Steuerungsmodul eine Transaktion im Steuerungsmodul (Pfeil C1), das die notwendigen Informationen aus der Datenbank extrahiert (Pfeil C2). Das Transaktionsmodul benachrichtigt das Steuerungsmodul über den Abschluß der Transaktion und stellt die Informationen dem Steuerungsmodul in einer temporären Datenstruktur zur Verfügung (Pfeil C3). Dieses aktiviert daraufhin das auszuführende Modul und übergibt die notwendigen Inputdaten (C4). Nach Abschluß ihrer Aktivitäten senden die Module Nachrichten an das Steuerungsmodul.
Da die Verarbeitungsmechanismen des Steuerungsmodul u. U. sehr schnell reagieren müssen, sind bei dessen Spezifikation Real-Time-Anforderungen zu berücksichtigen.
Das System ist durch höchste Flexibilität gekennzeichnet. Dies findet sich in der freien Konfigurierbarkeit des Transaktions- und Steuerungsmoduls wieder. Dazu ist es notwendig, unternehmensspezifische Konfigurationsparameter zu ermitteln. Verschiedene Konzepte für Transaktionsmodule wurden bereits entwickelt und realisiert (s. EG-Projekte CIDAM, IMPACCT). Die Vorgehensweise für die Ermittlung der Konfigurationsparameter für die vom IWi im ESPRIT II Projekt CIDAM entwickelte neutrale Datenbankschnittstelle INMAS, die in

modifizierter Form ein Transaktionsmodul darstellen könnte, ist in [IDS92][4] und [CIDAM93][5] beschrieben. Auch hier müsen Real-Time-Aspekte berücksichtigt werden.
Da das Steuerungsmodul, wie dargestellt, von zentraler Bedeutung ist, wird im folgenden ein den Anforderungen entsprechendes Steuerungsmodell für Informationssysteme entwickelt.

3. Entwicklung eines Steuerungsmodells für Informationssysteme

Steuern eines Systems bedeutet die Aktivierung und Kontrolle von Prozessen. Diese können sich aus einer Kombination von Modulablauffolgen und Datenbanktransaktionen zusammensetzen.

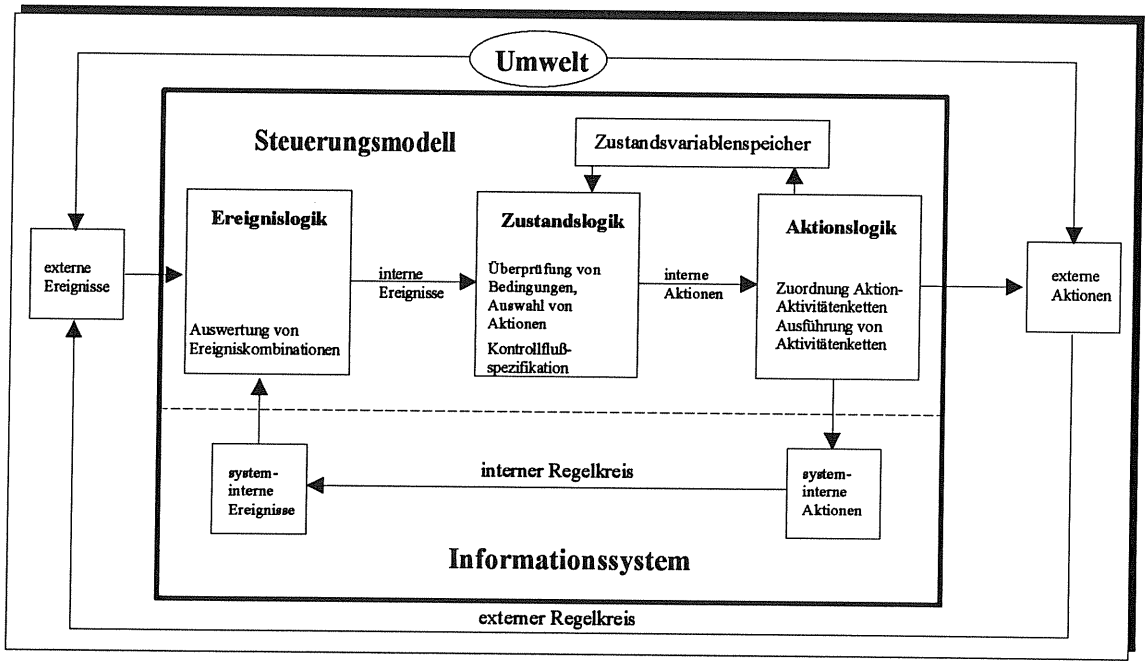

Abb. 2: Steuerungsmodell eines Informationssystems

Zur Aktivierung eines Prozesses bedarf es eines Auslösemechanismus, der dem System in Form von (internen oder externen) *Ereignissen* zugeführt wird. Hierzu müssen zunächst alle für das System wesentlichen Ereignisse systematisch erfaßt werden. Für diese Ereignisse werden die zugehörigen *Aktionen* des Systems spezifiziert. Die exakte Bestimmung der auszuführenden Aktionen ist jedoch nicht nur von aktuellen, sondern auch von vergangenen

---

4 Heß, H.; Hoffmann, W.; Herterich, R.; Houy, C.; Jung, S.: INMAS - ein Tool zur Datenintegration über eine neutrale Datenschnittstelle, in Scheer A.-W. (Hrsg.): Tagungsband der 4. Fachtagung: Datenbanken 1992, Saarbrücken 1992.

5 Hoffmann, W.; Koch, A.: Validation and Optimization of Testbed Application, Deliverable 6.2.4 of CIDAM-Project 2527, Commission of the European Communities, Saarbrücken 1993.

Ereignissen abhängig. Dieser Zusammenhang wird als *Zustand* des Systems bezeichnet. Im hier verwendeten Zustandsbegriff sind auch die Zustände relevanter Informationsobjekte enthalten.[6]

Aus diesen Ausführungen ergeben sich folgende elementare Komponenten des Steuerungsmodells:

- Ereignislogik,
- Zustandslogik und
- Aktionslogik.

Der Zusammenhang zwischen den Komponenten ist aus Abbildung 2 ersichtlich und wird im folgenden näher erläutert.

3.1. Ereignislogik

Im folgenden soll zunächst der Begriff *Ereignis*, wie er hier verwendet wird, definiert werden. In der Regel wird in der semantischen Prozeßmodellierung das Ereignis mit Zustandsänderungen von Informationsobjekten definiert.

"Ereignisse referenzieren auf Attribute, welche im Rahmen der Datensicht den Informationsobjekten des Datenmodells zugeordnet sind. Ein Ereignis beschreibt somit das Eingetreten sein von Ausprägungen (Werten) von Attributen, das eine Funktion auslöst. Somit exisitiert zwischen den Ereignissen und den Informationsobjekten des Datenmodells ein Zusammenhang. Ist ein vollständig attributiertes Datenmodell vorhanden, so können über die Identifizierung von Attributen und die Analyse möglicher Ausprägungen der Attribute potentielle Ereignisse erarbeitet werden. Ist kein Datenmodell vorhanden, so sind signifikante Ereignisse aus der Praxis zu identifizieren.[7]"

Ergänzend hierzu wird der Begriff Ereignis auf Zustandsänderungen innerhalb des betrachteten Systems angewendet, die nicht auf Informationsobjekte referenzieren. Solche, für die spätere Beschreibung von DV-Konzepten relevanten Ereignisse können beispielsweise sein: "Datensatz AB ist gesperrt" oder "Prozeß XY ist aktiviert".

Um aus der Gesamtheit der für ein System identifizierten Ereignisse diejenigen zu selektieren, die für die Ereignislogik relevant sind, kann eine Vorauswahl durch eine Klassifikation der Ereignisse erfolgen. Das Konzept der Ereignisklassifikation basiert auf einer

---

6 Beispielsweise können im Bereich der Fertigungssteuerung je nach Zustand des Informationsobjekts "Arbeitsgang" unterschiedliche Aktionen ausgeführt werden. Hat der Arbeitsgang den Zustand "Ressourcen zugeteilt", kann die Aktion "Arbeitsgangfreigabe" ausgeführt werden. Ansonsten müssen Aktionen zur Zuteilung von Ressourcen durchgeführt werden. Würde der Zustand "Ressourcen zugeteilt" im Steuerungsmodell nicht spezifiziert werden, erfolgt in jedem Fall direkt die Aktion "Arbeitsgangfreigabe".

7 Hoffmann, W.; Maldener, B.; Nüttgens, M.; Scheer, A.-W.: Das Integrationskonzept am CIM-TTZ Saarbrücken (Teil 2: Produktionssteuerung), Saarbrücken 1992.

Facettenklassifikation, die beliebig erweiterbar ist. Abbildung 3 zeigt Möglichkeiten, wie die Facetten des Klassifikationssystem mit ihren Ausprägungen aussehen können[8].
Interessant sind die Facetten F und G. Facette F charakterisiert, ob sich das Ereignis auf einen Zustand eines Informationsobjektes oder des Systems bezieht. Facette G beschreibt die Beteiligung an einem Zustandsübergang. Bei einer direkten Beteiligung an einem Zustandsübergang leitet die Ereignislogik das eingetretene Ereignis direkt an die Zustandslogik weiter. Eine indirekte Beteiligung eines Ereignisses an einem Zustandsübergang impliziert die Verwendung eines Regelmechanismus zur Sammlung und Auswertung von möglichen Ereigniskombinationen. Das durch Aggregation resultierende "Ergebnisereignis" wird an die Zustandslogik weitergeleitet. Ausprägung G3 kennzeichnet Ereignisse, die für die Ereignislogik nicht relevant sind.

Klassifikationssystem für Ereignistypen (Facettenklassifikation)	
Facette A: Abstraktionsebene	A1 übergeordneter Prozeß (Vorgang) A2 untergeordneter Prozeß (Vorgang)
Facette B: Modellierungsaspekt	B1 Startereignis B2 Endereignis B3 Start-/Endereignis B4 Zwischenereignis
Facette C: Systemeigenschaft	C1 systemintern C2 systeminterdependent/ intern C3 systemextern
Facette D: Eintrittsmerkmal (erzeugender Auslösemechanismus)	D1 automatisiert D2 interaktiv D3 manuell
Facette E: Eintrittszeitpunkt	E1 bestimmt E2 unbestimmt
Facette F: Zustandsbezug	F1 Informationsobjekt F2 System
Facette G: Beteiligung an Zustandsübergang	G1 direkte G2 indirekte G3 keine

Abb. 3: Klassifikationssystem für Ereignisse

[8] vgl Hoffmann, W.; Scheer, A.-W.; Backes, R.: Konzeption eines Ereignisklassifikationssystems, Saarbrücken 1992.

Abbildung 4 zeigt ein Beispiel für eine Ereignisaggregation. Aggregationen werden ausschließlich bei logischen "und"-Verknüpfungen ausgeführt. Solche Ereignisse, die indirekt an einer Zustandsänderung beteiligt sind, werden im *Ereignispuffer* zwischengespeichert. Diese Auswertung wird aus Performancegründen in der Ereignislogik behandelt, um die Zustandslogik von unnötigen Auswertungs- und Verwaltungsaufgaben zu entlasten.

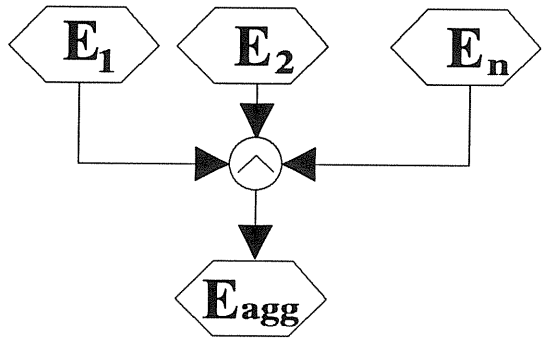

Abb. 4: Aggregation von Ereignissen

Die Ereignislogik hat also folgenden Aufgaben:

- Definition von Ereignissen,
- Umwandlung von externen zu systeminternen Ereignissen,
- Identifikation von Ereignissen,
- Pufferung und Aggregation von Ereignissen sowie
- Weiterleiten von Ereignissen an die Zustandslogik.

3.2. Zustandslogik

Der Begriff des Zustands wurde bereits in Kapitel 1 definiert. Weiterführend läßt sich diese Definition in dem hier betrachteten Zusammenhang um folgenden Sachverhalt ergänzen:

"Ein Zustand repräsentiert das Intervall zwischen Ereignissen und spezifiziert den Kontext, in dem die Ereignisse interpretiert werden."

Zusätzlich wird betont, daß unter dem Begriff Zustand nicht nur der Gesamtzustand eines Systems in seiner Beziehung zur Umwelt betrachtet wird, sondern es wird eine detailliertere Spezifikation der systeminternen Zustände durchgeführt. Es erfolgt also eine Auflösung der Gesamtreaktion des Systems in einzelne Teilschritte, die zur Erreichen des Zielzustandes erforderlich sind. Diese "Zwischenzustände" werden durch interne Ereignisse verursacht. Sie sind notwendig, um den korrekten Ablauf des Systems zu gewährleisten. So kann ein System z. B. aus mehreren Prozessen und Informationsobjekten bestehen, wobei für jeden einzelnen

Prozeß und jedem einzelnen Informationsobjekt zu einem diskreten Zeitpunkt ein eindeutiger Zustand zugeordnet werden kann.

Die Ausgangsvariablen a_i der Zustandslogik hängen nicht nur von den Eingangsereignissen e_i, sondern auch vom vorherigen Zustand des Systems ab. Alle Variablen (vergangene Ereignisse) des Systems, die neben den Eingangsereignissen (aktuelle Ereignisse) den Übergang in den nächsten Zustand beeinflussen, heißen Zustandsvariablen z_n. Ihre Speicherung erfolgt im Zustandsvariablenspeicher.
Die Menge der eintreffenden Ereignisse e_i heißt Eingangsvektor E:

$$E = \{e_1, e_2, \ldots e_n\}$$

Die Menge der Ausgangsvariablen a_i (Aktionsliste) heißt Ausgangsvektor A:

$$A = \{a_1, a_2, \ldots a_m\}$$

Die Menge der Zustandsvariablen z_n heißt Zustandsvektor Z:

$$Z = \{z_1, z_2, \ldots z_n\}$$

Der Vektor Z(t) beschreibt den dynamischen Zustand eines Systems zu jedem Zeitpunkt.
Im allgemeinen Fall führt die Lösung des Gleichungssystems

$$E \times Z$$

zu dem Ausgabevektor

$$A^9.$$

Um die Komplexität dieses Gleichungssystems zu reduzieren, erfolgen die exakten Zuordnungen der Ereignis-Zustandskombinationen, die Aktionen auslösen, durch die in Kapitel 4 entwickelte Modellierungsmethode. Diese basiert auf den funktionalen Abhängigkeiten:

$\delta(Z_a, e_i) = Z_n$	Z_a = aktueller Zustand, Z_n = neuer Zustand, δ = Zustandsübergangsfunktion
$\lambda(Z_n, e_i) = A$	λ = Funktion zur Ermittlung der auszuführenden Aktionen

---

9 Detaillierte Ausführungen zu dem Themengebiet liefert z. B. Föllinger, O.: Regelungstechnik, 5, verbesserte Auflage, Heidelberg 1985.

Durch diese Auslegungen werden für die Zustandslogik folgende Aufgaben festgelegt:
Eine Aufgabe der Zustandslogik ist es, aktuelle Zustände zu speichern. Dies bedeutet auch das Speichern vergangener, zustandsübergangsrelevanter Ereignisse. Die Speicherung der Zustände erfolgt im Zustandsvariablenspeicher. Durch das Abfragen (Zustandsübergangsfunktion) solcher vergangener Ereignisse in Zustandsübergangsbedingungen werden ergänzende Kontrollspezifikationen bereitgestellt.
Die Erfüllung einer Zustandsübergangsbedingung erfolgt durch das Eingetretensein von aktuellen Ereignissen, u. U. auch in Kombination mit vergangen Ereignissen. Dabei werden eine Reihe definierter Aktionen, die mit einer Übergangsbedingung verknüpft sind, an die Aktionslogik übertragen. Dies ist eine Aufgabe der λ -Funktion.

3.3 Aktionslogik

Die von der λ-Funktion übermittelten Funktionen werden in der Aktionslogik ausgewertet. Jeder Aktion können eine oder mehrere (applikationsspezifische) Aktivitätenkette(n) hinterlegt werden. Die Aktionslogik erlaubt eine freie Konfiguration der Aktivitätenketten. Eine Zentralfunktion der Aktionslogik ist die Ausführung und Kontrolle der Aktivitätenketten.
Sie erlaubt auch die Definition betriebssystemspezifischer Parameter. Dies beinhaltet beispielsweise die Festlegung der Maximalanzahl auszuführender Prozesse sowie die Möglichkeit einer Prioritätenregelung von Aktivitäten.

3.4. Real-Time-Aspekte des Steuerungsmodells

Durch die Existenz eines derart aufgebauten Steuerungsmodells in einem Informationssystem, läßt sich nicht nur die Applikationssoftware unternehmensspezifisch anpassen, sondern auch die Portierung auf unterschiedliche Hardware- und Betriebssystemplattformen mit geringem Aufwand realisieren (*Open System*).
Dieses Konzept eines Steuerungsmodells entspricht der Philosophie von Real-Time-Anwendungen (vgl. Kapitel 1). Um eine eindeutige und übersichtliche Beschreibung der Anforderungen, die solche Systeme stellen, zu gewährleisten, wird im folgenden Kapitel eine Methode zur Real-Time-Modellierung entwickelt. Diese basiert auf den Grundlagen der Ereignisorientierten Prozeßkette.

4. Herleitung einer Methode zur Real-Time-Modellierung

4.1. Basismethode: Die Ereignisorientierte Prozeßkette (EPK)

Die EPK ist eine Modellierungsmethode zur Darstellung von Prozessen. Sie wird im Fachkonzept der Steuerungssicht von ARIS angewendet und stellt eine Verknüpfung des Fachkonzeptes der Funktionssicht mit dem der Datensicht dar, wenn man das Ereignis der Datensicht zurechnet. Die EPK besteht aus den Komponenten:

- Ereignis,
- Funktion,
- Kanten und
- Verknüpfungsoperatoren.

Mit der EPK wird der zeitlich-logische Ablauf von Funktionen dargestellt. Funktionen werden von einem Auslösemechanismus gestartet, der als Ereignis bezeichnet wird. Ereignisse starten somit Funktionen und können wiederum ein Ergebnis von Funktionen sein[10]. Hierdurch lassen sich zusammenhängende Prozeßketten abbilden. Zur Begriffsbestimmung von Ereignissen vgl. Kapitel 3.1. Eine detaillierte Beschreibung der EPK und ihrer Konstrukte geben Hoffmann et al.[11]

Um den Anforderungen der Real-Time-Modellierung gerecht zu werden, müssen zusätzliche Objekte definiert, zueinander in Beziehung gesetzt und in die EPK eingeordnet werden.

4.2. Objekte der Methode

Aus den Anforderungen zur strukturierten Abbildung des Steuerungsmodells ergeben sich folgende Objekte:

- Systemzustände,
- Zustandsübergangsbedingungen,
- Aktionen,
- Zustandsübergang,
- Kontrollflüsse,
- Ereignisse.

Ereignisse sind expliziet in der EPK enthalten und werden, da sie von dort unverändert übernommen werden, in den folgenden Ausführungen nicht beschrieben.

---

[10] vgl. Scheer, A.-W.: Architektur integrierter Informationssysteme - Grundlagen der Unternehmensmodellierung. Berlin et al. 1992, S. 90-92.

[11] Hoffmann, W.; Kirsch, J.; Scheer, A.-W.: Modellierung mit Ereignisgesteuerten Prozeßketten. in: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 101, Saarbrücken 1992.

4.1.1. Der Systemzustand

Alle für die Steuerung des Systems relevanten Stati werden durch Ausprägungen von Variablen definiert. Wenn man zu einem diskreten Zeitpunkt t_n die Werte aller Zustandsvariablen festhält, so erhält man den eindeutigen Zustand $Z(t_n)$ des Systems. Die Kombination von Zustandsvariablenwerten wird als Zustandsmuster bezeichnet. Das Zustandsmuster wird im Zustandsvariablenspeicher abgebildet. Änderungen des Zustandsmusters werden von der Aktionslogik direkt im Zustandsvariablenspeicher durchgeführt.

Als grafisches Symbol wird in der Methode für den Zustandsvariablenspeicher, der den aktuellen Systemzustand repräsentiert, ein Kreis verwendet.

Abb. 5: Grafisches Symbol des Zustandsvariablenspeichers

4.1.2. Die Zustandsübergangsbedingung

Zustandsübergangsbedingungen können aus einer Kombination von in der Vergangenheit aufgetretenen Ereignissen resultieren. Sie werden durch aktuelle Ereignisse ergänzt und bewirken durch das Erfülltsein ihrer logischen Verknüpfungen (Boolsche Operatoren) den Übergang des Systems vom aktuellen Zustand in den Folgezustand. Die Zustandsübergangsbedingung wird der Zustandslogik zugeordnet.

Als grafisches Symbol wird in der Methode für die Zustandsübergangsbedingung ein gestrichelt umrandetes Softrectangle verwendet.

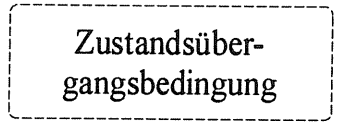

Abb. 6: Grafisches Symbol der Zustandsübergangsbedingung

4.1.3. Aktionen

4.1.3.1. Aktionsauswahl

Wird eine Zustandsübergangsbedingung erfüllt, müssen die zum Erreichen des Folgezustands notwendigen Aktionen ermittelt werden. Unter dem Begriff Aktion wird hierbei ein logischer Name verstanden, der sich in der Aktionslogik wiederfindet, in der die auszuführenden Funktionsaufrufe spezifiziert sind.
Diese Aktionen sind fest mit der Zustandsübergangsbedingung verknüpft und werden an die Aktionslogik übertragen. Die Aktionsauswahl wird ebenfalls der Zustandslogik zugeordnet.
Als grafisches Symbol wird ein Pfeil verwendet.

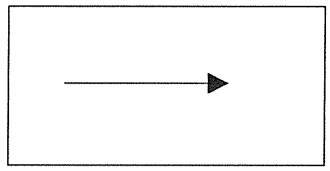

Abb. 7: Grafisches Symbol der Aktionsauswahl

4.1.3.2. Aktionsdurchführung

Den von der Zustandslogik übertragenen Aktionen werden die jeweiligen Aktivitäten zugeordnet und diese ausgeführt. Gleichzeitig nach jeder einzelnen Aktivität erfolgt die Änderung der entsprechenden Zustandsvariable (vgl. Kapitel 4.1.5). Die Aktionsdurchführung ist Bestandteil der Aktionslogik.
Als grafisches Symbol wird für die Aktionsdurchführung ein gestricheltes Sechseck verwendet.

Abb. 8: Grafisches Symbol der Aktionsdurchführung

4.1.4. Zustandsübergang

Wenn das System von einem aktuellen Zustand in den Folgezustand wechselt, bezeichnet man dies als Zustandsübergang des Systems. Dies nennt man auch "Schalten" des Systems. Der Zustandsübergang erfolgt nach der Ausführung der über die Aktionsauswahl spezifizierten Funktionsaufrufe durch die Aktionslogik. Nach dem Zustandsübergang ist der Kontext für die

folgende Zustandsübergangsbedingung spezifiziert. Der Zustandsübergang ist Bestandteil der Zustandslogik.
Als grafisches Symbol wird ein Doppelpfeil verwendet

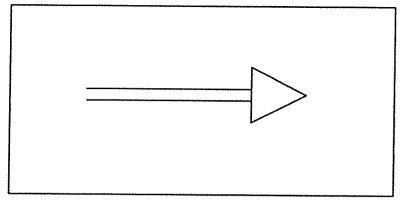

Abb. 9: Grafisches Symbol des Zustandsübergangs

4.1.5. Kontrollflüsse

Aufgabe von Kontrollflüssen ist die Verarbeitung und Bereitstellung von Systemzuständen.
In der vorliegenden Methode wird eine zusammengesetzte Kontrollflußspezifikation verwendet. Kontrollflüsse werden von Zustands- und Aktionslogik verwendet. Der Zustandslogik wird über den Kontrollfluß der jeweilige aktuelle Systemzustand zur Verfügung gestellt. Von der Aktionslogik wird über den Kontrollfluß parallel zu den ausgeführten Aktivitäten das hierdurch entstehende Zustandsmuster in den Zustandsvariablenspeicher eingetragen.
Als grafisches Symbol für Kontrollflüsse werden gepunktete Pfeile verwendet.

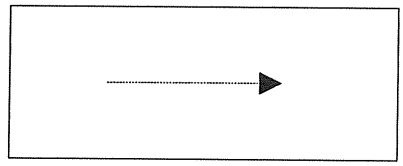

Abb. 10: Grafisches Symbol für Kontrollflüsse

4.1.6. Verknüpfung mit Ereignisorientierten Prozeßketten

Die aktive Schnittstelle von der neu entwickelten Methode zu der EPK ist die Aktionslogik. Von ihr aus müssen Funktionen und Prozesse aktiviert werden, die durch die Funktionen der EPK repräsentiert werden.
Ebenso läßt sich eine aktive Schnittstelle von der EPK zu der neuen Methode finden, nämlich die Ereignislogik. Ereignisse in der EPK werden in Form von Messages (z. B. Trigger) an die Ereignislogik übergeben.
Diese Schnittstellen werden als Steuerflüsse bezeichnet. Als grafisches Symbol wird ein strichpunktierter Pfeil verwendet.

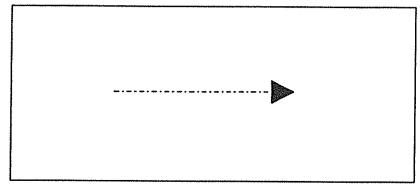

Abb. 11: Grafisches Symbol für Steuerflüsse

Die Integration der neuen Methode zur Real-Time-Modellierung in die EPK zeigt Abbildung 13. Die integrierte Methode wird als ***Real-Time erweiterte EPK (rEPK)*** bezeichnet. Die Ablauffolge wird durch Zahlen dargestellt. Parallel laufende Prozesse werden durch ein Hochkomma an derselben Zahl gekennzeichnet.

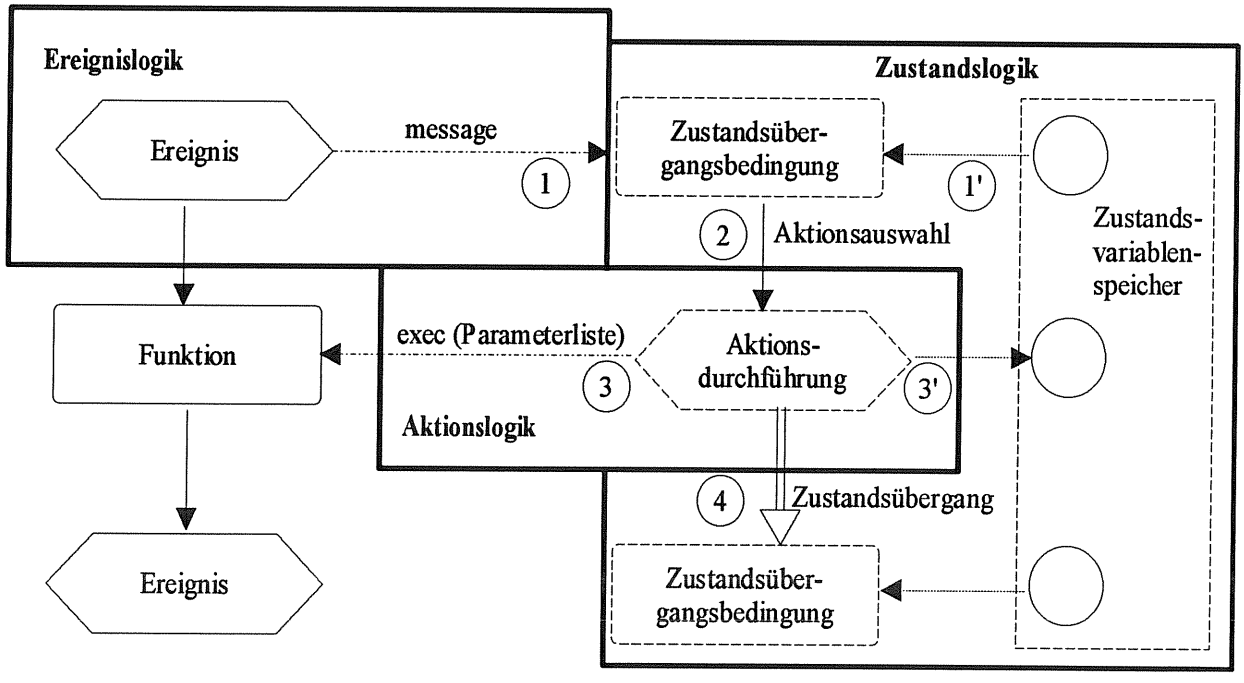

Abb. 12: Die Real-Time erweiterte EPK (rEPK)

4.2. Einordnung des Steuerungsmodells in die ARIS-Architektur

Das Steuerungmodell und die beschreibende Methode sind dem DV-Konzept der Steuerungssicht in ARIS zugeordnet. Abbildung 13 zeigt das dazugehörige erweiterte Informationsmodell.

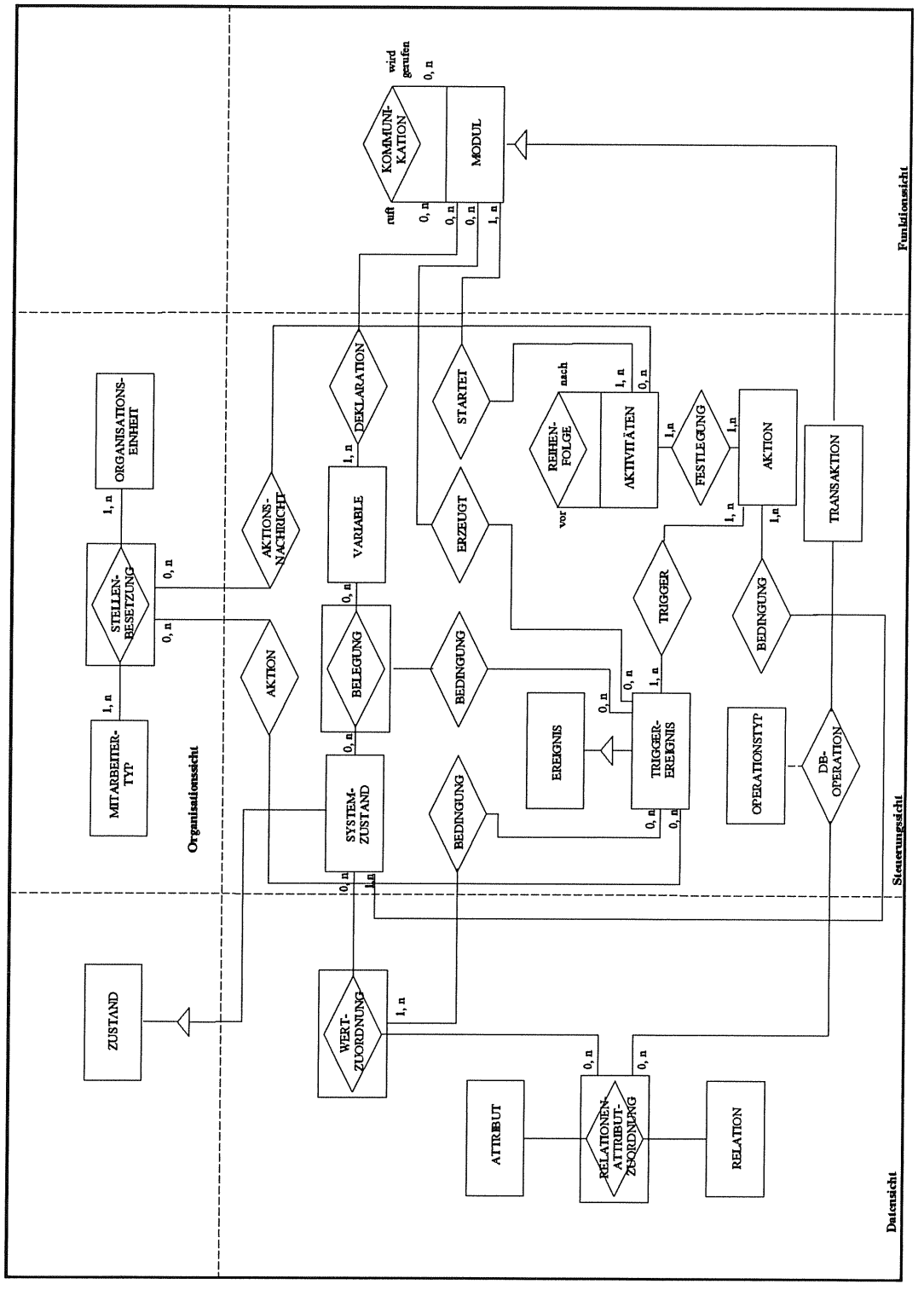

Abb. 13: Informationsmodell des Steuerungsmodells

4.3. Vergleich der rEPK mit Petri-Netzen

Als Vergleichsmethode mit der rEPK dienen Petri-Netze. Im folgenden wird ein Einblick in den statischen Aufbau und das dynamische Verhalten von Petri-Netzen gegeben.

❑ Die statische Struktur

Ein Petri-Netz ist ein gerichteter Graph, der durch gerichtete Kanten (Pfeile) und zwei Klassen von Knoten ('Platz' und 'Transition') dargestellt wird. In der graphischen Darstellung sind als Symbole Kreise für Plätze und Rechtecke für Transitionen eingeführt. Petri-Netze sind bipartitive Graphen, d. h. die Kanten laufen entweder von Plätzen zu Transitionen oder umgekehrt. Es werden aber niemals Transitionen oder Plätze miteinander verknüpft. Mehrfachkanten oder isolierte Kanten sind ausgeschlossen[12].

❑ Die dynamische Struktur

Um dynamische Abläufe kenntlich zu machen, benutzt die Petri-Netz-Darstellung Marken. Jeder Platz, der den aktuellen Zustand des Netzes repräsentiert, wird mit einer Marke belegt. Bedingungs-Ereignis-Netze, die im folgenden betrachtet werden, sind Einmarkensysteme, d. h. pro Platz ist nur eine Marke zulässig[13]. Sind alle Eingangsplätze ausnahmslos mit einer Marke belegt, so ist eine Transition aktiviert, sofern ihre Ausgangsplätze ausnahmslos markenfrei sind. Unter Voraussetzung der Aktivierung kann eine Transition schalten. Dabei werden die Marken von ihren Eingangsplätzen entfernt und die Ausgangsplätze mit je einer Marke belegt.
Im folgenden Abschnitt erfolgt eine kurze Einführung in das Bedingungs-Ereignis-Netz. Eine detailliertere Darstellung gibt beispielsweise REISIG[14].

4.3.1. Das Bedingungs-Ereignis-Netz

In Bedingungs-Ereignis-Netzen (B/E-Netz) werden die Plätze als Bedingungen und die Transitionen als Ereignisse interpretiert. Das B/E-Netz läßt eine Aufnahmekapazität von nur einer Marke pro Platz und nur eine fließende Marke pro Kante beim Schaltvorgang zu[15]. Die aktiven Komponenten des B/E-Netzes sind Ereignisse, die passiven Komponenten die Bedingungen.
Aus dem B/E-Netz lassen sich vereinfachte Netzformen ableiten. In Abbildung 14 sind die Herleitung eines vereinfachten ereignisorientierten Netzes und die Herleitung eines

---

12 Vgl.: Zuse, K.: Petri-Netze aus der Sicht des Ingenieurs; Braunschweig, Wiesbaden 1980.
13 Vgl.: Reisig, W.: Petri-Netze-eine Einführung, Berlin et al. 1982.
14 Reisig, W.: Petri-Netze-eine Einführung, Berlin et al. 1982.
15 Vgl.: Abel, D.; Rake, H.: Simulation von komplexen Steuerungssystemen mit Petri-Netzen, in: Proceedings of the 2nd European Simulation Congress, Antwerpen 1986.

vorgangsorientierten Netzes dargestellt[16]. Letzteres eignet sich besonders zum Vergleich mit einer Modellierungsmethode wie der EPK-Modellierung. Unter einem Vorgang werden hierbei der Start, die Ausführung und das Ende einer Aktion verstanden. Dadurch steigt der Abstraktionsgrad des Netzes und die Ähnlichkeit mit der EPK wird ersichtlich.

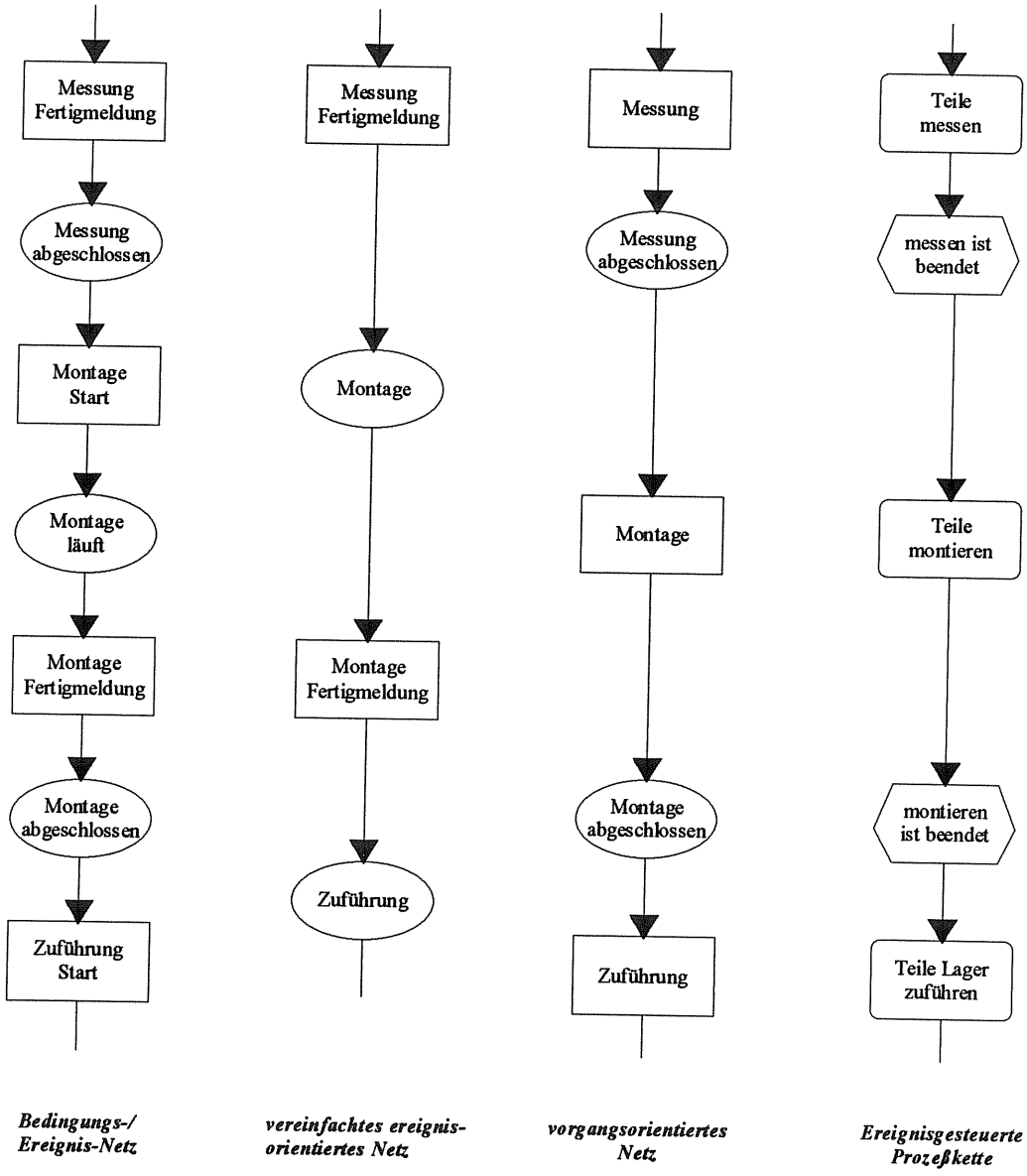

Abb. 14: Analogie der EPK zu Petri-Netzen

Die EPK verwendet ebenfalls gerichtete Kanten zur Abbildung des logischen Zusammenhangs der Systemkomponenten. Die Funktionen der EPK lassen sich mit den aktiven Komponenten des B/E-Netzes vergleichen, also den Ereignissen. Ein grundlegender Unterschied ist jedoch

16 Vgl.: Groha, A.: Universelles Zellenrechnerkonzept für flexible Fertigungssysteme, Berlin et al. 1988.

zwischen den Ereignissen der EPK und den Bedingungen des B/E-Netzes festzustellen. Ereignisse in der EPK repräsentieren das "Ergebnis der Zustandsänderung von Informationsobjekten". Das ist für die Bedingungen des B/E-Netzes nur dann der Fall, wenn sie mit einer Marke belegt sind. Dies bedeutet, daß Zustandsänderungen im B/E-Netz durch die Weitergabe von Marken abgebildet werden, in der EPK geschieht dies durch die Darstellung der Start- und Endereignisse einer Funktion. Dieses Endereignis kann dabei wiederum Startereignis einer weiteren Funktion sein. Dementsprechend bietet die EPK eine den Petri-Netzen vergleichbare Grundlage zur Simulation von Abläufen[17]. Ein Vorteil der EPK gegenüber dem B/E-Netzes ist die Verwendung von Verknüpfungsoperatoren, die Funktionen Entscheidungsbefugnisse über den weiteren Verlauf des Prozeßablaufes geben.

4.3.2. Unterschiede zwischen rEPK und Bedingungs-Ereignis-Netzen

Erster Abgrenzungspunkt ist die explizite Unterscheidung zwischen Prozeß-, RT- und Kontrollebenen in der rEPK, die in den Petri-Netzen nicht vorgesehen ist. Um diese Dreiteilung mit Hilfe von Petri-Netzen darstellen zu können müßten mindestens zwei verschiedene Netze erstellt werden, wobei die Beziehungen zwischen den Netzen verloren gehen.

Ein weiterer Unterschied liegt in der Organisation der Zustandsvariablen der rEPK bzw. der Marken des B-E-Netzes. In der rEPK stehen sie, sobald sie gesetzt wurden, während des gesamten Prozeßablaufes zur Verfügung, das System kann sich also selbständig an frühere Zustände erinnern. Im B-E-Netz müßte ein weiterer Pfeil vom Ereignis ausgehen, was zu immer komplexeren Netzstrukturen führen würde.

Durch die unterschiedlichen Bezeichnungen können die verschiedenen Zustandsvariablen in der rEPK unterschieden werden. Damit weist die rEPK eine gewisse Ähnlichkeit mit den Prädikaten-Transitionen-Netzen auf, die sich durch individuelle Marken auszeichnen. Desweiteren werden in der rEPK interne Verarbeitungsfunktionen von Funktionen des Systemumfeldes grafisch getrennt. Dies ist mit Hilfe von Petri-Netzen nicht möglich, zumindest existiert derzeit noch keine Netzerweiterung in diese Richtung.

Bei Netzarten mit anonymen Marken wie dem B-E-Netz führt die Darstellung komplexer Zustandsübergangsbedingungen zu einer unübersichtlicheren Darstellung als in der rEPK. Müssen z. B. zum Auslösen einer Aktion fünf Bedingungen erfüllt sein, so sind in der Petri-Netz-Technik fünf Bedingungskreise notwendig, in der rEPK ist hingegen ein Symbol ausreichend. Bei mit einem logischen "oder" verknüpften Bedingungen wird die Darstellung in Petri-Netzen noch komplexer.

Mit der rEPK lassen sich dieselben Sachverhalte abbilden wie mit Petri-Netzen. Sie bietet jedoch eine kompaktere und übersichtlichere Darstellungsweise des abzubildenden Problemfeldes.

---

17 Vgl. Spang, S.: Informationsmodellierung im Investitionsgütermarketing, Wiesbaden 1993, S. 86 ff.

Die Überführung der rEPK in Petri-Netze ist ohne größeren Aufwand möglich, da in der rEPK sämtliche zur Abbildung in Petri-Netze erforderlichen Informationen enthalten sind. Für die verschiedenen Petri-Netz-Derivate müssen geeignete Transformationsregeln bereitgestellt werden.
Eine solche Überführung ist von Interesse, da bereits eine große Anzahl von Quellcode-Generatoren und Simulationswerkzeugen auf der Basis von Petri-Netzen verfügbar sind.
Dieser Sachverhalt legt nahe, eine Dreiteilung bei der Entwicklung von Informationssystemen vorzunehmen. Wie in Abbildung 15 dargestellt werden den verschiedenen Spezifikationsstufen der Systementwicklung, die sich durch unterschiedliche Abstraktions- und Komplexitätsgrade der Modelle auszeichnen, die geeigneten Modellierungsmethoden zugeordnet.

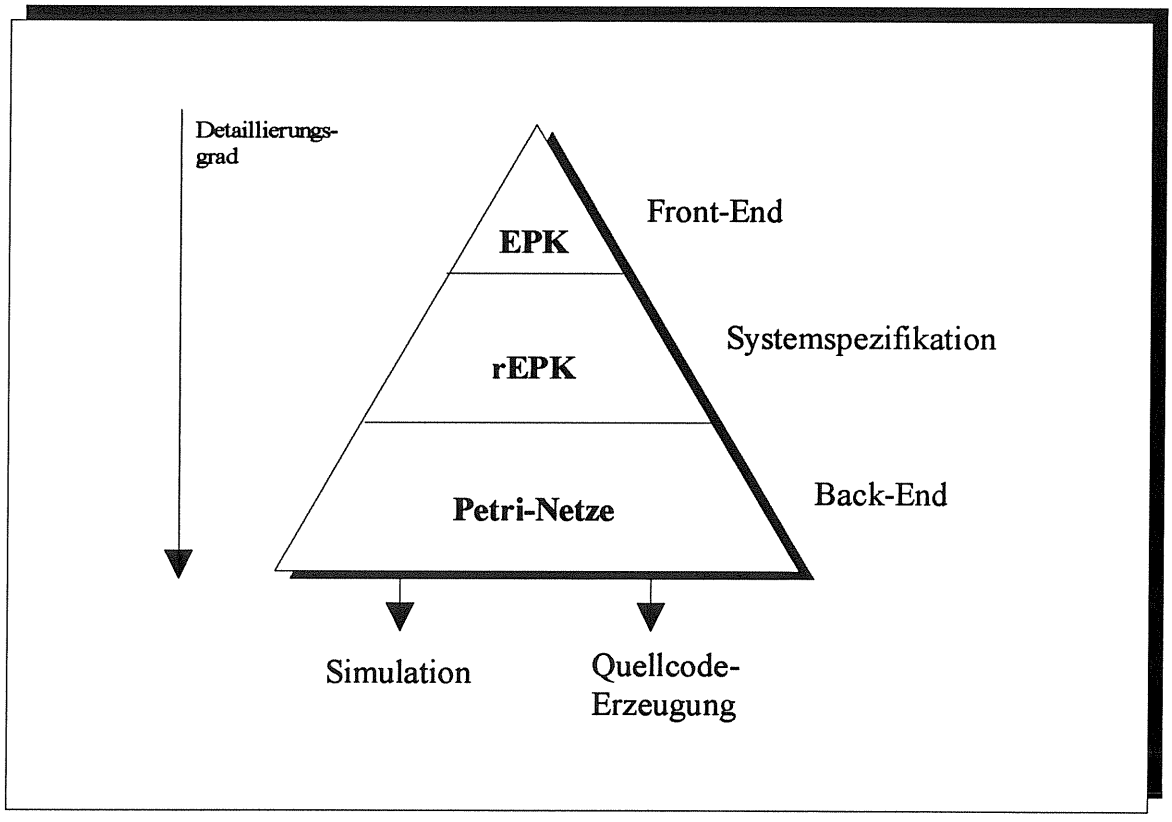

Abb. 15: Spezifikationsstufen der Systementwicklung

5. Anwendungsbeispiele

5.1. Teilauszug eines Real-Time-Systems

Zur Verdeutlichung der beschriebenen Sachverhalte wird ein konkreter Anwendungsfall modelliert. Es handelt sich hierbei um die Ansteuerung eines Lagersystems. Es handelt sich

hierbei um ein Real-Time-Informationssystem, repräsentiert durch eine Speicherprogrammierbare Steuerung (SPS), das mit einem Lagerverwaltungssystem verbunden ist. Die Zentraleinheit einer SPS hat eine ähnliche Struktur wie das entwickelte Steuerungsmodell. Die für das Steuerungsmodell benötigten Konfigurationsparameter werden im SPS-Programm abgebildet. Diese berücksichtigen sowohl statische als auch dynamische Systemaspekte. In Abbildung 16 wird ein Teilauszug eines Auslagervorgangs mit Hilfe der rEPK beschrieben. Abbildung 17 zeigt die Transformation in ein Petri-Netz.

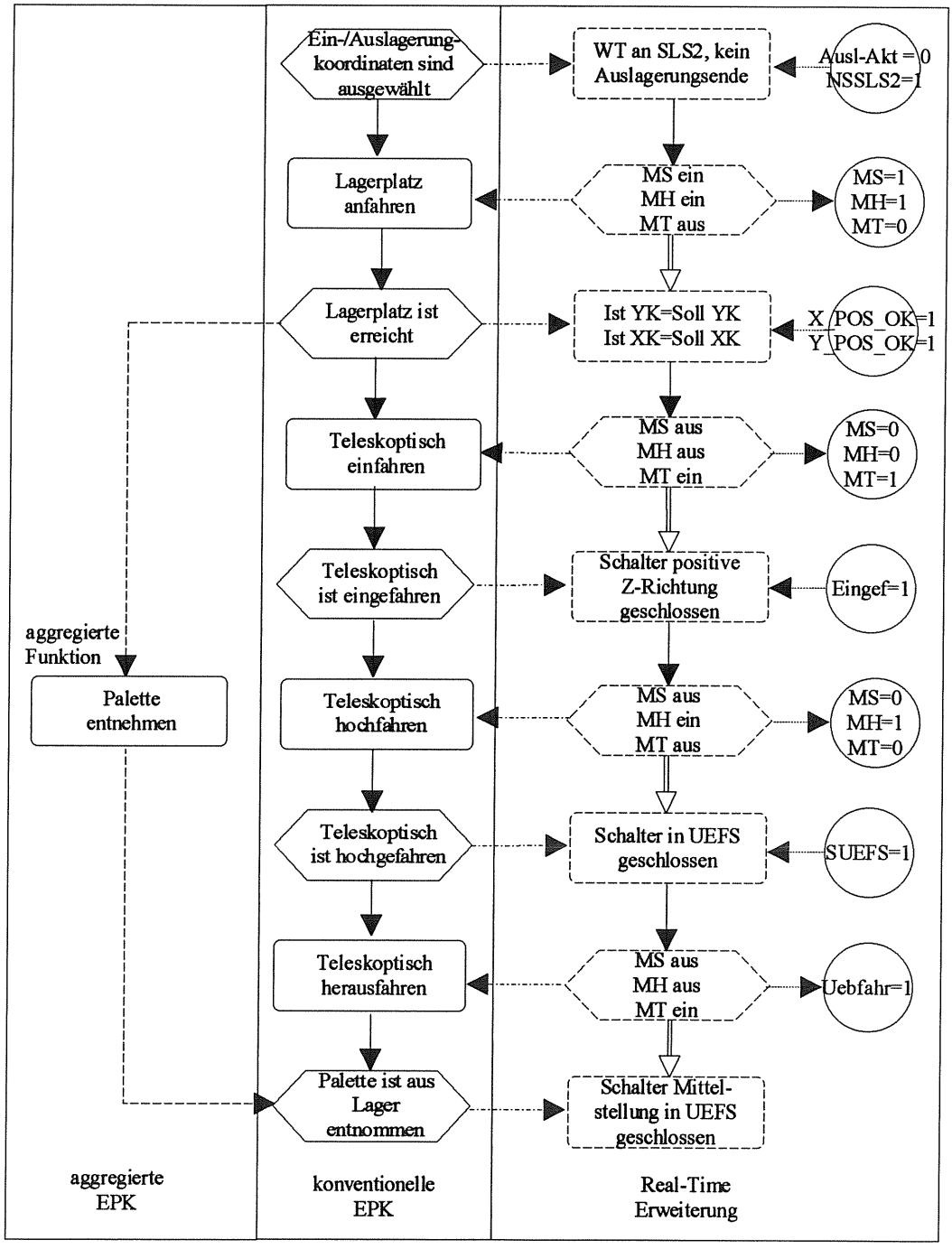

Abb. 16: Teilauszug eines Auslagervorgangs mit der rEPK.

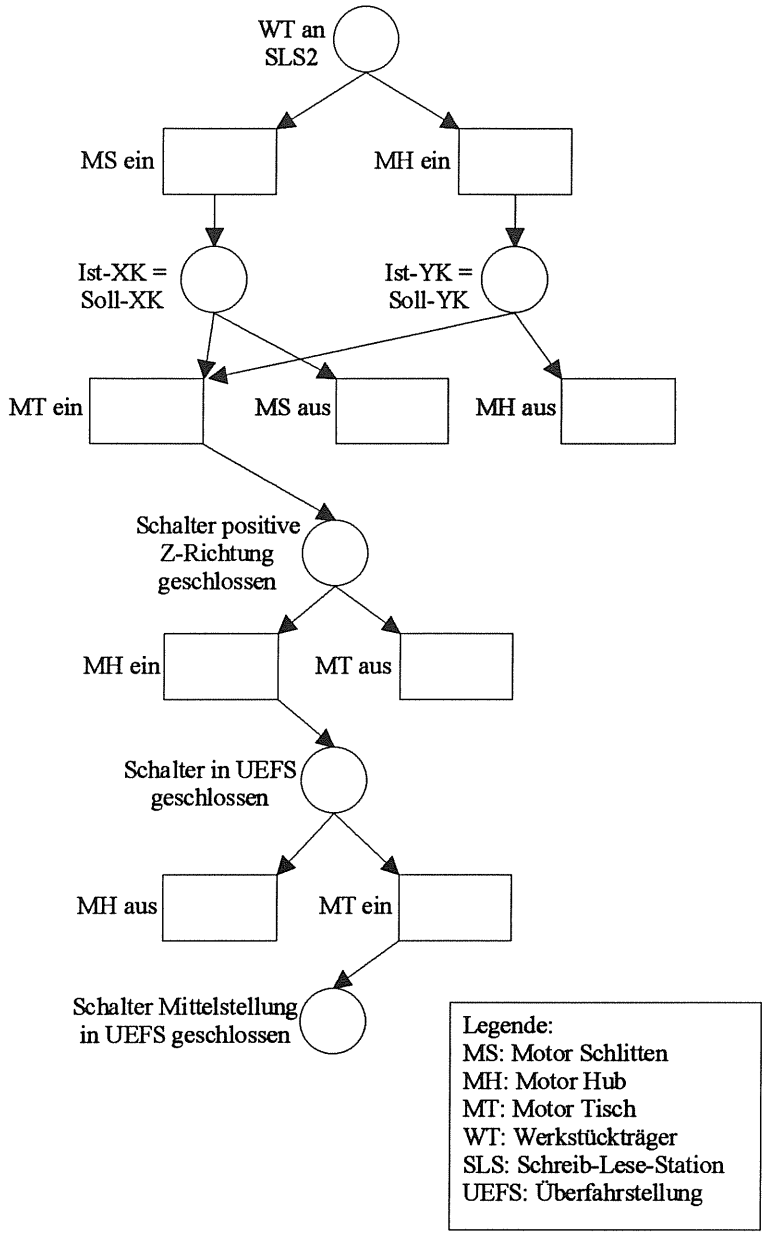

Abb. 17: Transformatierte Darstellung in einem Petri-Netz

5.2. Einsatz der rEPK bei der modellgestützten Konfiguration von Informationssystemen

In Abbildung 18 wird der Prozeß einer Unternehmensanalyse von der Ist-Analyse (Reverse-Engineering) über die Ermittlung unternehmensspezifischer Soll-Prozeßmodelle, abgeleitet aus Referenzmodellen (Customizing), bis zum Erhalt der restrukturierten Ist-Prozeßmodelle dargestellt. Diese können zur Auswahl von Standardsoftware durch Vergleich mit den jeweiligen Prozeßmodellen herangezogen werden oder aber die Notwendigkeit der Erstellung

von Individualsoftware aufzeigen. Durch eine konsequente Anwendung der rEPK können in diesem Kreislauf die erforderlichen Konfigurationsparameter des ausgewählten Informationssystems ermittelt werden und für die unternehmensspezifische Implementierung bereitgestellt werden.

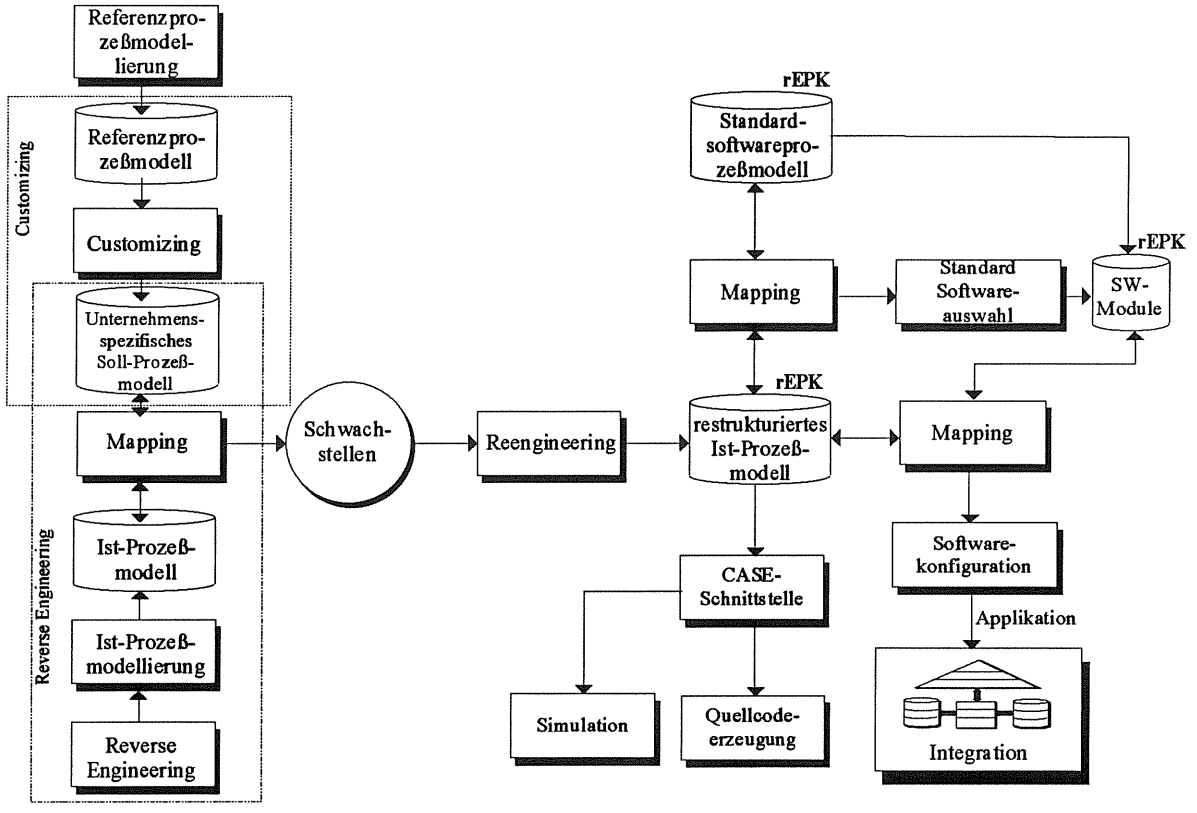

Abb. 18: Einordnung der rEPK in ein Gesamtkonzept zur Softwareauswahl, -generierung, konfiguration und Implementierung

Literaturverzeichnis

Abel, D.; Rake, H.: Simulation von komplexen Steuerungssystemen mit Petri-Netzen, in: Proceedings of the 2nd European Simulation Congress, Antwerpen 1986.

Föllinger, O.: Regelungstechnik, 5, verbesserte Auflage, Heidelberg 1985.

Groha, A.: Universelles Zellenrechnerkonzept für flexible Fertigungssysteme, Berlin et al. 1988.

Heß, H.; Hoffmann, W.; Herterich, R.; Houy, C.; Jung, S.: INMAS - ein Tool zur Datenintegration über eine neutrale Datenschnittstelle, in Scheer A.-W. (Hrsg.): Tagungsband der 4. Fachtagung: Datenbanken 1992, Saarbrücken 1992.

Hoffmann, W.; Kirsch, J.; Scheer, A.-W.: Modellierung mit Ereignisgesteuerten Prozeßketten. in: Scheer, A.-W. (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 101, Saarbrücken 1992.

Hoffmann, W.; Koch, A.: Validation and Optimization of Testbed Application, Deliverable 6.2.4 of CIDAM-Project 2527, Commission of the European Communities, Saarbrücken 1993.

Hoffmann, W.; Maldener, B.; Nüttgens, M.; Scheer, A.-W.: Das Integrationskonzept am CIM-TTZ Saarbrücken (Teil 2: Produktionssteuerung), Saarbrücken 1992.

Hoffmann, W.; Scheer, A.-W.; Backes, R.: Konzeption eines Ereignisklassifikationssystems, Saarbrücken 1992.

Raasch, J.: Systementwicklung mit strukturierten Methoden, München Wien 1991

Ward, P. T.; Mellor, S. J.: Strukturierte Systemanalyse von Echtzeitsystemen, München Wien 1991

Reisig, W.: Petri-Netze-eine Einführung, Berlin et al. 1982.

Scheer, A.-W.: Architektur integrierter Informationssysteme - Grundlagen der Unternehmens-modellierung. Berlin et al. 1992

Spang, S.: Informationsmodellierung im Investitionsgütermarketing, Wiesbaden 1993

Zuse, K.: Petri-Netze aus der Sicht des Ingenieurs; Braunschweig, Wiesbaden 1980.