Zweitveröffentlichung

Schissler, Martin; Mantel, Stephan; Ferstl, Otto K.; Sinz, Elmar J.

Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen und ihre Realisierung mit SAP R/3

Datum der Zweitveröffentlichung: 28.08.2024

Akzeptiertes Manuskript (Postprint), Zeitschriftenartikel

Persistenter Identifikator: urn:nbn:de:bvb:473-irb-976033

Erstveröffentlichung

Schissler, Martin; Mantel, Stephan; Ferstl, Otto K.; Sinz, Elmar J. (2002): „Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen und ihre Realisierung mit SAP R/3“. In: Wirtschaftsinformatik : WI, Jg. 44, Nr. 5, S. 459-468, Berlin ; Heidelberg: Springer, doi: 10.1007/BF03250867.

Verlagshinweis

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature’s AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/BF03250867.

Rechtehinweis

Dieses Werk ist durch das Urheberrecht und/oder die Angabe einer Lizenz geschützt. Es steht Ihnen frei, dieses Werk auf jede Art und Weise zu nutzen, die durch die für Sie geltende Gesetzgebung zum Urheberrecht und/oder durch die Lizenz erlaubt ist. Für andere Verwendungszwecke müssen Sie die Erlaubnis der Rechteinhaberinnen und Rechteinhaber einholen.

Für dieses Dokument gilt das deutsche Urheberrecht.

Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen und ihre Realisierung mit SAP R/3

Martin Schissler, Stephan Mantel, Otto K. Ferstl, Elmar J. Sinz[1]

[1] Dipl.-Wirtsch.Inf. Martin Schissler, Dipl.-Wirtsch.Inf. Stephan Mantel, Prof. Dr. Otto K. Ferstl, Prof. Dr. Elmar J. Sinz, Bayerischer Forschungsverbund Wirtschaftsinformatik (FORWIN), Projekt OASYS. Fakultät Wirtschaftsinformatik und Angewandte Informatik, Otto-Friedrich-Universität Bamberg, D-96045 Bamberg. E-Mail: {martin.schissler | stephan.mantel | otto.ferstl | elmar.sinz}@wiai.uni-bamberg.de

Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen und ihre Realisierung mit SAP R/3

Kernpunkte für das Management

Die Integration von Anwendungssystemen durch Kopplungsarchitekturen wird zur Zeit vor Allem unter dem Schlagwort EAI diskutiert. Vor dem Hintergrund des E-Business hat dabei die überbetriebliche Integration verstärkte Bedeutung erlangt. Der vorliegende Artikel leistet einen Beitrag zur Systematisierung und Strukturierung von Kopplungsarchitekturen zur überbetrieblichen Integration von Anwendungssystemen.

- Reihenfolgebeziehung, partielle Identität von Aufgabenobjekt-Instanzen, partielle Gleichheit von Aufgabenobjekt-Typen und partielle Gleichheit von Lösungsverfahren sind Aufgabenbeziehungen, die eine Integration von Anwendungssystemen erforderlich machen.
- Zur Umsetzung dieser Integration dienen ereignis-, daten- und funktionsorientierte Kopplungsarchitekturen.
- SAP R/3 Release 4.6C bietet mit BAPI, ALE und SAP Business Workflow / WebFlow drei Kopplungsmechanismen, die als Basis für die Realisierung von Kopplungsarchitekturen eingesetzt werden können.

Stichworte

Überbetriebliche Geschäftsprozesse, Aufgabenbeziehungen, überbetriebliche Kopplung von Anwendungssystemen, Kopplungsarchitektur, Kopplungsmechanismus, SAP R/3

Zusammenfassung

Unternehmensübergreifende Geschäftsprozesse erfordern eine überbetriebliche Integration der diese Geschäftsprozesse unterstützenden Anwendungssysteme. Die Integration erfolgt auf der Grundlage überbetrieblicher Kopplungen von Anwendungssystemen, die in Form von Kopplungsarchitekturen spezifiziert werden. Eine Kopplung von Anwendungssystemen ist immer dann notwendig, wenn die von diesen Anwendungssystemen zu unterstützenden betrieblichen Aufgaben in Beziehung stehen. Ausgehend von den relevanten Aufgabenbeziehungen werden drei Klassen von Kopplungsarchitekturen identifiziert und konkrete Ausprägungen vorgestellt. Es wird untersucht, inwieweit die Kopplungsarchitekturen mithilfe von Kopplungsmechanismen, wie sie von SAP R/3 bereitgestellt werden, realisiert werden können.

Keywords

Business-to-business processes, task relationships, inter-company coupling of application systems, coupling architecture, coupling mechanism, SAP R/3

Title

Architectures for Inter-Company Integration of Application Systems and their Implementation using SAP R/3

Abstract

Business-to-business processes require an inter-company integration of the application systems supporting these business processes. Integration is based on inter-company couplings of application systems, which are specified in coupling architectures. The necessity of coupling of application systems arises from relationships between business tasks which are to be supported by these application systems. Starting with relevant relationships between business tasks, three classes of coupling architectures are identified and illustrated by concrete instances. It is investigated to which extent coupling mechanisms as provided by SAP R/3 are suitable for an implementation of these coupling architectures.

1. Einleitung

Die Gestaltung und Durchführung unternehmensübergreifender Geschäftsprozesse erfordert eine überbetriebliche Integration der diese Geschäftsprozesse unterstützenden Anwendungssysteme (AwS). Diese Integration erfolgt dabei auf der Grundlage einer überbetrieblichen Kopplung von AwS, die in Form von zugehörigen Kopplungsarchitekturen spezifiziert wird.

Die Notwendigkeit einer Kopplung von AwS ergibt sich immer dann, wenn die von diesen AwS unterstützten betrieblichen Aufgaben miteinander in Beziehung stehen. Im Falle der überbetrieblichen Kopplung von AwS sind die zu koppelnden AwS unterschiedlichen Unternehmen zugeordnet.

Der vorliegende Artikel leistet einen Beitrag zur Systematisierung und Strukturierung von Kopplungsarchitekturen. Es werden vier für die Kopplung von AwS relevante Aufgabenbeziehungen und drei zur Unterstützung dieser Beziehungen geeignete Klassen von Kopplungsarchitekturen identifiziert. Die Kopplungsarchitekturklassen werden anhand konkreter Ausprägungen von Kopplungsarchitekturen beschrieben, die sich am State-of-the-Art orientieren, wie er sich in derzeit verfügbaren Produkten und Gestaltungsvorschlägen widerspiegelt. Dabei werden die wesentlichen Subsysteme herausgearbeitet, die zur Realisierung der jeweiligen Kopplungsarchitektur notwendig sind.

Die vorgeschlagene Strukturierung wird anhand der Standardsoftware SAP R/3 exemplarisch erprobt. Standardsoftware wie SAP R/3 unterstützt im Allgemeinen die Realisierung von Kopplungsarchitekturen durch Bereitstellung von mehr oder weniger elementaren Kopplungsmechanismen. Es wird untersucht, inwieweit die identifizierten Kopplungsarchitekturen mit Hilfe verschiedener von R/3 angebotener Kopplungsmechanismen realisiert werden können.

Der Beitrag ist wie folgt aufgebaut: In Kapitel 2 wird zunächst, ausgehend von betrieblichen Aufgaben und deren Beziehungen sowie von betrieblichen Aufgabenträgern (vgl. [FeSi01, 90]), die Notwendigkeit der Kopplung von AwS verdeutlicht. Es werden vier Arten von Beziehungen zwischen betrieblichen Aufgaben vorgestellt, die unterschiedliche Kopplungen von AwS erfordern. In Kapitel 3 werden drei Klassen von Kopplungsarchitekturen identifiziert. Es wird diskutiert, wie die einzelnen Aufgabenbeziehungen durch diese Architekturen unterstützt werden können. Für jede Klasse werden konkrete Ausprägungen vorgestellt. Kapitel 4 beschäftigt sich schließlich mit Kopplungseigenschaften von SAP R/3 Release 4.6C. Es werden drei Kopplungsmechanismen von R/3 hinsichtlich ihrer Eignung für die in Kapitel 3 vorgestellten Kopplungsarchitekturen diskutiert.

Fragen der Integration von AwS gehören zu den klassischen Problemstellungen der Wirtschaftsinformatik (vgl. z.B. [Beck91; Sche90; Mert00]). In letzter Zeit wird dieses Thema unter anderem unter dem Schlagwort EAI (Enterprise Application Integration) wieder ver-

stärkt diskutiert (vgl. z.B. [Lint01]). Vor dem Hintergrund des E-Business hat hierbei die überbetriebliche Integration von AwS verstärkte Bedeutung erlangt.

2. Betriebliche Aufgaben und ihre Beziehungen

In betrieblichen Informationssystemen wird zwischen der Aufgaben- und der Aufgabenträgerebene unterschieden [FeSi01, 2ff]. Aufgabenträger sind Personen und AwS. Eigenschaften von Personen werden in die folgende Beschreibung nicht einbezogen. Ein AwS ist ein maschineller Aufgabenträger zur automatisierten Durchführung von Aufgaben eines betrieblichen Informationssystems.

Eine betriebliche Aufgabe besteht in der Außensicht aus einem Aufgabenobjekt und Aufgabenzielen sowie aus Vor- und Nachereignissen [FeSi01, 90]. Vorereignisse lösen eine Aufgabendurchführung aus, Nachereignisse sind das Ergebnis einer Aufgabendurchführung. Bei Aufgabenobjekten wird im Folgenden zusätzlich zwischen Typ (AO-Typ) und Instanz (AO-Instanz) unterschieden. Der AO-Typ umfasst die zu einer Aufgabe gehörigen Attribute eines betrieblichen Systems. AO-Instanzen beinhalten Attributwerte. Die Innensicht einer Aufgabe beschreibt ein oder mehrere Lösungsverfahren der Aufgabe, die im Rahmen einer Aufgabendurchführung auf den AO-Instanzen ausgeführt werden (vgl. Bild 1~~Bild 1~~).

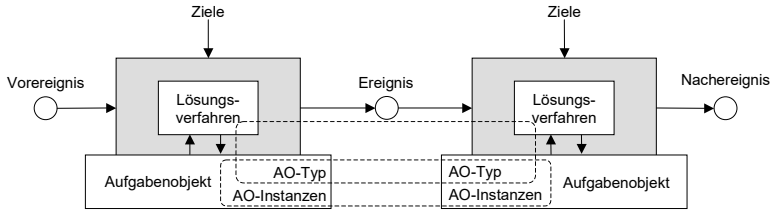

Bild 1 Aufgaben und ihre Beziehungen

Zwischen den Aufgaben eines betrieblichen Informationssystems können unterschiedliche Beziehungen bestehen, die durch entsprechende Kopplungen der AwS zu unterstützen sind. Es werden vier Arten von Beziehungen unterschieden, die einzeln oder in Kombination auftreten (vgl. Bild 1~~Bild 1~~).

hat formatiert: Schriftart: Nicht Fett

(1) Reihenfolgebeziehung zwischen Aufgaben

Ist das Nachereignis einer Aufgabe zugleich das Vorereignis einer weiteren Aufgabe, so besteht zwischen den beiden Aufgaben eine Reihenfolgebeziehung. Sind die beiden Aufgaben unterschiedlichen AwS zugeordnet, müssen die AwS so gekoppelt werden, dass zwischen ihnen das Ereignis übertragen werden kann.

(2) Partielle Gleichheit von AO-Typen / (3) Partielle Identität von AO-Instanzen

Operieren die Lösungsverfahren unterschiedlicher Aufgaben auf teilweise übereinstimmenden Attributmengen ihrer Aufgabenobjekte, so liegt eine partielle Gleichheit der AO-Typen dieser

Aufgaben vor. Sind darüber hinaus auch noch die entsprechenden Attributwerte identisch, so liegt auch eine partielle Identität von AO-Instanzen dieser Aufgaben vor. Falls die betrachteten Aufgaben wiederum unterschiedlichen AwS zugeordnet sind, müssen diese so gekoppelt werden, dass eine Abstimmung hinsichtlich der AO-Typen und AO-Instanzen möglich ist.

(4) Partielle Gleichheit von Lösungsverfahren

Die vierte Beziehungsart beschäftigt sich mit der partiellen Gleichheit von Lösungsverfahren unterschiedlicher Aufgaben. Aus der partiellen Gleichheit der Lösungsverfahren von Aufgaben folgt wegen der Verwendung gleicher Attribute auch eine partielle Gleichheit ihrer AO-Typen. Die weitergehende partielle Identität von AO-Instanzen ist dabei jedoch nicht zwingend. Sind die betrachteten Aufgaben wiederum unterschiedlichen AwS zugeordnet, müssen diese so gekoppelt werden, dass eine Abstimmung hinsichtlich des gemeinsamen Lösungsverfahrens möglich ist.

3. Kopplungsarchitekturen

Die Integration von AwS erfolgt gemäß einer Kopplungsarchitektur. Diese projiziert die Architekturen mehrerer integrierter AwS bezüglich ihrer Kopplungseigenschaften. Sie umfasst alle für die Kopplung erforderlichen Elemente der AwS-Architekturen sowie deren Beziehungen.

Kopplungsarchitekturen werden sowohl zur Kopplung von AwS innerhalb eines Unternehmens als auch zur Kopplung von AwS unterschiedlicher Unternehmen eingesetzt. Bei der überbetrieblichen Kopplung von AwS treten allerdings zusätzliche Probleme auf, die bei der Entwicklung von Kopplungsarchitekturen zu berücksichtigen sind. Ein häufig auftretendes Problem ist die Überwindung der technischen und fachlichen Heterogenität der AwS. Die beteiligten AwS differieren z.B. bezüglich Benennung, Darstellung und Handhabung gleicher Attribute. Eine weitere Forderung an Kopplungsarchitekturen, die insbesondere für die überbetriebliche Integration von AwS von Bedeutung ist, ist die Beachtung der Unabhängigkeit der AwS. Dabei wird zwischen Betriebs- und Entwicklungsunabhängigkeit unterschieden.

Beispiele:

- Betriebsunabhängigkeit: Eine Nachricht an ein AwS sollte zu einem beliebigen Zeitpunkt übermittelt werden können, auch dann, wenn dieses wegen Wartungsarbeiten heruntergefahren wurde.
- Entwicklungsunabhängigkeit: Änderungen im Programmcode eines Nachrichten empfangenden AwS sollten nach Möglichkeit keine Änderungen im Programmcode der sendenden AwS erforderlich machen.

Bei der Beschreibung von Kopplungsarchitekturen wird im Folgenden zwischen dem Kern und dem Kopplungs-Subsystem eines AwS unterschieden. Das Kopplungs-Subsystem enthält diejenigen Elemente eines AwS, die ausschließlich der Kopplung mit anderen AwS dienen. Der AwS-Kern enthält Elemente, die für die Kopplung der AwS nicht relevant sind oder nicht

ausschließlich im Kopplungskontext genutzt werden. Die Kopplungsarchitektur eines AwS umfasst dessen Kopplungs-Subsystem sowie die Elemente des AwS-Kerns, die im Kopplungskontext genutzt werden.

Beispiel:

> Ein AwS benötigt Zugriff auf eine Datenbanktabelle eines mit diesem AwS gekoppelten R/3-Systems. Das Kopplungs-Subsystem stellt die hierzu erforderlichen Zugriffsmechanismen in Form eines BAPI bereit (vgl. zu BAPI Abschnitt 4.1). Die genutzte Datenbanktabelle selbst ist Teil des R/3-AwS-Kerns. Die Kopplungsarchitektur des R/3-Systems umfasst hier die Zugriffsmechanismen als Teil des Kopplungs-Subsystems und die Tabelle als Teil des AwS-Kerns.

Die Beschreibungen der Kopplungsarchitekturen in den folgenden Abschnitten beschränken sich auf die Darstellung von strukturellen Aspekten. Verhaltensaspekte werden nicht betrachtet. Zur Beschreibung der Strukturen wird die in UML vorgesehene Notation für Subsysteme verwendet. Es werden die verschiedenen Subsysteme einer Kopplungsarchitektur, deren Funktionen und die Kommunikationsbeziehungen zwischen den Subsystemen dargestellt. Der zugehörige Kontrollfluss wird nicht dargestellt.

3.1 Ereignisorientierte Kopplungsarchitekturen

Eine ereignisorientierte Kopplungsarchitektur dient der Übertragung von Ereignissen und zugehörigen Daten[2] zwischen AwS durch Austausch von Nachrichten in Form einer losen Kopplung (vgl. [FeSi01, 225]). Dabei müssen die verwendeten Nachrichtenformate und Kommunikationsprotokolle festgelegt werden. Wünschenswert ist darüber hinaus eine Unterstützung bei der Abstimmung der Datenobjekttyp-Spezifikationen der ausgetauschten Daten in den sendenden und empfangenden AwS (vgl. zu Datenobjekttyp [FeSi01, 292]).

Ereignisorientierte Kopplungsarchitekturen können grundsätzlich zur Unterstützung folgender Aufgabenbeziehungen eingesetzt werden:

- Reihenfolgebeziehungen zwischen Aufgaben,
- Partielle Identität von AO-Instanzen,
- Partielle Gleichheit von AO-Typen.

Die ersten beiden Beziehungsarten werden von ereignisorientierten Kopplungsarchitekturen obligatorisch, die partielle Gleichheit von AO-Typen wird jedoch nur fakultativ unterstützt.

Im Folgenden wird eine typische ereignisorientierte Kopplungsarchitektur dargestellt. Hierzu wird eine Aufgabe A und eine Menge weiterer Aufgaben betrachtet. Alle Aufgaben sind unterschiedlichen Unternehmen zugeordnet. Das Nachereignis der Aufgabe A bildet das Vorer-

[2] Der Begriff Daten wird in diesem Beitrag im Sinne von Dateninstanzen verwendet.

eignis der weiteren Aufgaben. Zusätzlich liegt eine Identität im Bereich der AO-Instanzen vor, d.h. die nachfolgenden Aufgaben benötigen Ergebnisse von Aufgabe A als Input. Die beschriebenen Aufgabenbeziehungen werden durch eine ereignisorientierte Kopplungsarchitektur unterstützt, deren wesentliche Subsysteme in Bild 2~~Bild 2~~ dargestellt sind.

hat formatiert: Schriftart: Nicht Fett

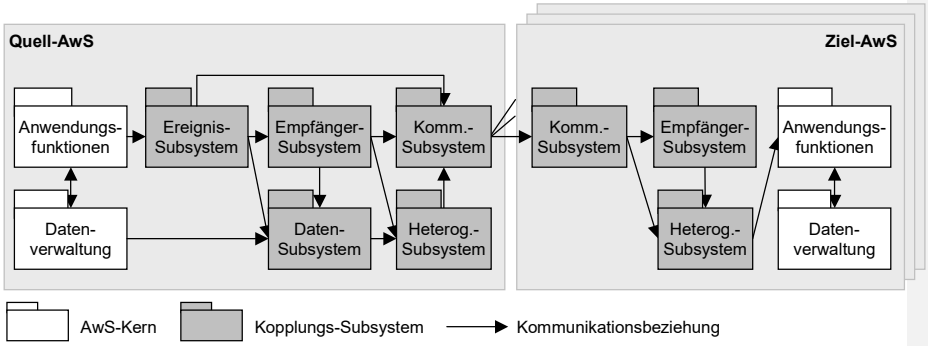

Bild 2 Ereignisorientierte Kopplungsarchitektur

Ausgangspunkt der Übermittelung eines Ereignisses ist dessen Eintreten im AwS-Kern des Quell-AwS, das die Vorgänger-Aufgabe A realisiert. Empfänger des Ereignisses sind ein oder mehrere Ziel-AwS, welche die Nachfolger-Aufgaben realisieren. Das **Ereignis-Subsystem** besitzt das Wissen darüber, welche im Quell-AwS eintretenden Ereignisse für Ziel-AwS relevant sind und beobachtet unter diesem Gesichtspunkt den Kern des Quell-AwS.

Die Verteilung der Ereignisse auf die Ziel-AwS übernimmt das **Empfänger-Subsystem**. Es bestimmt primär auf der Basis des Ereignistyps, welche AwS vom Eintreten eines Ereignisses informiert werden müssen. Sind zusätzliche Informationen zur Bestimmung der Ziel-AwS heranzuziehen, so können folgende Varianten unterschieden werden:

- zustandssensitives / zustandsloses Empfänger-Subsystem,
- inhaltssensitives / nicht-inhaltssensitives Empfänger-Subsystem.

Ein inhaltssensitives Empfänger-Subsystem greift auf die mit dem Ereignis zu übermittelnden Daten zu und wertet diese für die Bestimmung der Ziel-AwS aus. Ein zustandssensitives Empfänger-Subsystem bestimmt die Ziel-AwS in Abhängigkeit von vorher eingetretenen Ereignissen. Die in Bild 2~~Bild 2~~ dargestellten Empfänger-Subsysteme werden als zustandslos und nicht-inhaltssensitiv betrachtet.

hat formatiert: Schriftart: Nicht Fett

Sollen zusätzlich zu einem Ereignis auch Daten an die Ziel-AwS übertragen werden, so muss das **Daten-Subsystem** diese unter Zugriff auf den AwS-Kern ermitteln.

Das **Heterogenitäts-Subsystem** dient der Überwindung der fachlichen Heterogenität zwischen Quell- und Ziel-AwS. In Bild 2~~Bild 2~~ wird davon ausgegangen, dass das Quell-AwS die Daten in ein spezielles, von den AwS-internen Formaten abweichendes Übertragungsfor-

hat formatiert: Schriftart: Nicht Fett

mat umsetzt und die Ziel-AwS dieses Übertragungsformat wiederum in die dort benötigten Formate umformen. Dem Aufwand des zweifachen Umsetzens steht eine höhere Entwicklungsunabhängigkeit gegenüber, da ein AwS hierdurch nicht die internen Formate eines anderen AwS kennen muss.

Das **Kommunikations-Subsystem** des Quell-AwS übernimmt die Übertragung des Ereignisses und der Daten vom Quell- zu den Ziel-AwS und überwindet gleichzeitig die technische Heterogenität zwischen den AwS. Vorzugsweise wird eine asynchrone, gepufferte Kommunikationsform verwendet, bei der das Quell-AwS nicht von der Verfügbarkeit des Zielsystems abhängt. Hierdurch wird eine relativ hohe Betriebsunabhängigkeit von Quell- und Ziel-AwS ermöglicht.

In den Ziel-AwS werden Daten und Ereignisse durch das dortige **Kommunikations-Subsystem** entgegengenommen. Anschließend ermittelt das **Empfänger-Subsystem** die für die Verarbeitung des Ereignisses zuständige Funktion im jeweiligen Ziel-AwS. Eine direkte Adressierung einer Funktion durch das Quell-System ist unter dem Aspekt der Vermeidung von Entwicklungsabhängigkeiten zwischen den gekoppelten AwS nicht sinnvoll. Das **Heterogenitäts-Subsystem** setzt die übertragenen Daten in die Formate des Ziel-AwS um.

3.2 Datenorientierte Kopplungsarchitekturen

Datenorientierte Kopplungsarchitekturen dienen der Manipulation gemeinsamer Daten mehrerer AwS in Form einer engen Kopplung der auf den Daten operierenden Funktionen (vgl. [FeSi01, 225]).

Bei der datenorientierten AwS-Kopplung wird zwischen einer Kopplung auf Instanz- und einer Kopplung auf Typ-Ebene der Daten unterschieden. Ziel der Kopplung auf Instanz-Ebene, die von datenorientierten Kopplungsarchitekturen generell unterstützt wird, ist die gemeinsame Nutzung von Daten. Aus softwaretechnischer Sicht ist dabei z.B. zu klären, ob jedes AwS eine eigene Kopie der Daten verwaltet oder ob alle beteiligten AwS eine gemeinsame Kopie der Daten verwenden und in welchem AwS diese verwaltet wird.

Ziel der Kopplung auf Typ-Ebene ist die gemeinsame Nutzung von Datenobjekttyp-Spezifikationen.

Beispiel:

> Im Falle eines relationalen DBMS würde sich die Kopplung auf Typ-Ebene mit der gemeinsamen Nutzung von Datenbankschemata beschäftigen.

Grundsätzlich können mit datenorientierten Kopplungsarchitekturen folgende Aufgabenbeziehungen realisiert werden:

- Partielle Identität von AO-Instanzen,

- Partielle Gleichheit von AO-Typen.

Die partielle Identität von AO-Instanzen wird von datenorientierten Kopplungsarchitekturen generell unterstützt, die Gleichheit von AO-Typen zusätzlich von solchen datenorientierten Kopplungsarchitekturen, die eine Abstimmung von Datenobjekttyp-Spezifikationen vorsehen.

3.2.1 Datenorientierte Kopplungsarchitektur mit redundanter Datenhaltung

Im Folgenden wird eine typische datenorientierte Kopplungsarchitektur beschrieben. Unterstützt werden mehrere Aufgaben unterschiedlicher Unternehmen mit teilweise identischen AO-Instanzen. Es wird davon ausgegangen, dass nur eine der betrachteten Aufgaben die gemeinsamen AO-Instanzen modifiziert. Diese Aufgabe wird von dem als Quell-AwS bezeichneten AwS realisiert.

Nicht betrachtet wird der Fall, dass mehrere Aufgaben die gemeinsamen AO-Instanzen modifizieren. Dieser Fall führt auf der Ebene der Aufgabenrealisierungen zu konfliktären Schreibzugriffen (vgl. z.B. [Rahm94, 163ff]). Diese müssen so behandelt werden, dass eine 1-Kopien-Serialisierbarkeit erreicht wird (vgl. [BHG87]).

Gemeinsame AO-Instanzen implizieren eine Gleichheit der zugehörigen AO-Typen. Diese wird durch die vorliegende Kopplungsarchitektur nicht behandelt.

Wesentliches Merkmal dieser Architektur ist die redundante Speicherung der gemeinsamen AO-Instanzen mit dem Ziel einer hohen Betriebsunabhängigkeit. Aus der redundanten Speicherung folgt allerdings das Problem der Gewährleistung der Datenkonsistenz (vgl. z.B. [Krem99, 107ff]). Dieses wird durch entsprechende Mechanismen der Kopplungsarchitektur behandelt.

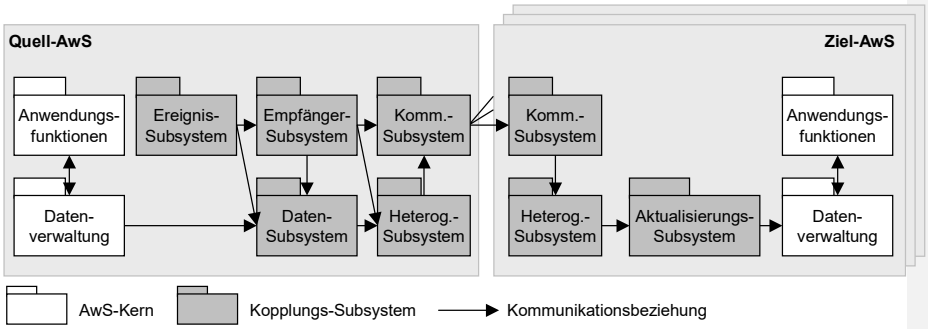

Bild 3 Datenorientierte Kopplungsarchitektur mit redundanter Datenhaltung

Die in Bild 3~~Bild 3~~ dargestellte Kopplungsarchitektur unterscheidet sich hinsichtlich ihres Aufbaus in nur wenigen Punkten von der in Abschnitt 3.1 beschriebenen ereignisorientierten Kopplungsarchitektur. Aus diesem Grund beschränken sich die folgenden Ausführungen auf die Darstellung der Unterschiede. Ein wesentlicher Unterschied besteht darin, dass das Kopp-

hat formatiert: Schriftart: Nicht Fett

lungs-Subsystem in den Ziel-AwS nicht mit dem Subsystem für Anwendungsfunktionen, sondern mit dem Datenverwaltungs-Subsystem verbunden ist. Die empfangenen Daten werden dort direkt abgelegt. Eine darüber hinausgehende Verarbeitung wird nicht angestoßen.

Die hier beschriebene Kopplungsarchitektur folgt dem Push-Prinzip, wobei das **Ereignis-Subsystem** des Quell-AwS bestimmt, wann eine Propagation zu erfolgen hat. Dabei können folgende Varianten unterschieden werden:

- Propagation zu definierten Zeitpunkten (in Bild 3~~Bild 3~~ dargestellte Variante),
- Propagation beim Eintreten von fachlichen Ereignissen im AwS-Kern.

Das **Daten-Subsystem** ermittelt auch bei dieser Architektur die zu übermittelnden Daten. In einem Propagationslauf werden entweder alle Daten oder nur geänderte Daten übertragen. Im letzteren Fall werden zusätzliche Mechanismen zu deren Bestimmung benötigt.

Das **Aktualisierungs-Subsystem** übernimmt in den Ziel-AwS die Aktualisierung der Daten unter Verwendung der von den Datenverwaltungs-Subsystemen angebotenen Schnittstellen. Dabei muss insbesondere gewährleistet werden, dass die Daten in der Reihenfolge geändert werden, in der sie auch im Quell-AwS geändert wurden. Die Bildung von Datenbanktransaktionen im Quell-AwS ist auch in den Ziel-AwS zu berücksichtigen.

3.2.2 Datenorientierte Kopplungsarchitektur mit nicht-redundanter Datenhaltung

Nun wird eine zweite datenorientierte Kopplungsarchitektur vorgestellt, die sich von der ersten durch eine zentralisierte, nicht-redundante Haltung der gemeinsamen Daten unterscheidet. Die oben getroffene Einschränkung, dass gemeinsame Daten nur durch eine Aufgabe manipuliert werden, wird nun aufgehoben.

Die Verwaltung der zentral gehaltenen Daten erfolgt durch ein eigenes Datenverwaltungs-Subsystem, das eine (standardisierte) Datenschnittstelle bereitstellt und konfliktäre Schreibzugriffe behandelt (Bild 4~~Bild 4~~). Die **Kommunikations-Subsysteme** übertragen bei dieser Kopplungsarchitektur die Anfragen von den dezentralen AwS-Kernen an die zentrale Datenverwaltung und die Ergebnisse der Anfrage in der Gegenrichtung.

Die **Heterogenitäts-Subsysteme** dienen auch hier der Überwindung der fachlichen Heterogenität.

hat formatiert: Schriftart: Nicht Fett

hat formatiert: Schriftart: Nicht Fett

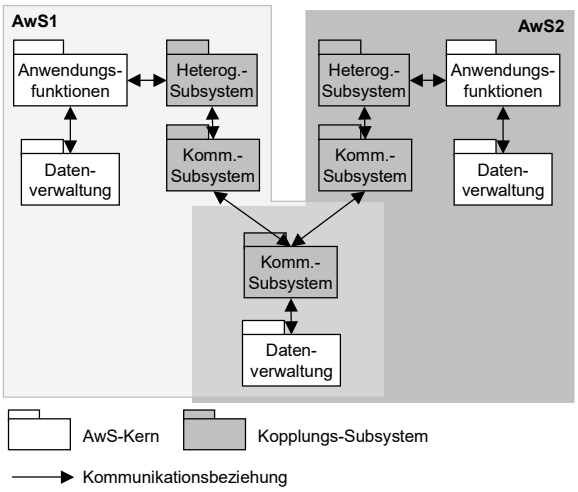

Bild 4 Datenorientierte Kopplungsarchitektur mit nicht-redundanter Datenhaltung

Wie in Bild 4~~Bild 4~~ dargestellt, liegt bei dieser Kopplungsarchitektur aus funktionaler Sicht eine Überlappung der AwS vor; das gemeinsame Teilsystem ist sowohl Teil von AwS1 als auch von AwS2. Aus softwaretechnischer Sicht kann das gemeinsame Teilsystem entweder als eigenständige Komponente oder zusammen mit den weiteren Subsystemen von AwS1 oder AwS2 in einer Komponente realisiert werden.

hat formatiert: Schriftart: Nicht Fett

3.3 Funktionsorientierte Kopplungsarchitekturen

Funktionsorientierte Kopplungsarchitekturen ermöglichen die gemeinsame Nutzung von Funktionen und ggf. zugehöriger Daten durch mehrere AwS. Diese Klasse von Kopplungsarchitekturen muss folglich sowohl die Realisierung der Funktionen als auch die Realisierung der durch die Funktionen bearbeiteten Daten behandeln. Es muss u.a. festgelegt werden,

- ob Programmmodule, welche die gemeinsame Funktion realisieren, redundant oder nicht-redundant gehalten werden,
- ob die gemeinsame Funktion in separaten Prozessen je AwS oder in einem gemeinsamen Prozess ausgeführt wird,
- wie die gemeinsame Funktion aktiviert wird.

In Bezug auf die Daten wird auch hier die bereits in Abschnitt 3.2 angesprochene Unterscheidung zwischen Typ- und Instanz-Ebene getroffen. Grundsätzlich stellen sich hier die gleichen Probleme, die im Rahmen der datenorientierten Kopplungsarchitekturen behandelt werden. Das Problem der Kopplung auf Typ-Ebene ist bei den funktionsorientierten Kopplungsarchitekturen generell gegeben. Es wird zumindest teilweise bereits im Rahmen der Fragestellun-

gen zur Haltung von Programmmodulen behandelt, da diese ja auch Typ-Spezifikationen beinhalten. Hinsichtlich der Instanz-Ebene der Daten sind unterschiedliche Szenarien denkbar. Sind die durch die Funktion bearbeiteten Daten für alle nutzenden AwS gleich, so stellen sich auf der Instanz-Ebene wiederum die im Rahmen der Diskussion der datenorientierten Kopplungsarchitekturen bereits angesprochenen Probleme.

Mit Hilfe von funktionsorientierten Kopplungsarchitekturen können grundsätzlich die folgenden Aufgabenbeziehungen behandelt werden:

- Partielle Gleichheit von Lösungsverfahren,
- Partielle Gleichheit von AO-Typen,
- Partielle Identität von AO-Instanzen.

Dabei wird nur die partielle Gleichheit von Lösungsverfahren generell von funktionsorientierten Kopplungsarchitekturen unterstützt; die Unterstützung der weiteren Beziehungsarten hängt von der konkreten Ausprägung der Kopplungsarchitektur ab.

3.3.1 Funktionsorientierte Kopplungsarchitektur mit redundanter Modulhaltung, separaten Prozessen und redundanter Datenhaltung

Die im Folgenden beschriebene Form einer typischen funktionsorientierten Kopplungsarchitektur geht davon aus, dass für eine Menge von Aufgaben ein teilweise gleiches Lösungsverfahren verwendet werden soll. Die Aufgaben werden von AwS verschiedener Unternehmen durchgeführt. Die AO-Instanzen, auf denen das gemeinsame Lösungsverfahren operiert, sind für alle Aufgaben identisch. Sowohl die Gleichheit der Lösungsverfahren, als auch die Identität der AO-Instanzen sollen durch die Kopplungsarchitektur unterstützt werden. Es wird davon ausgegangen, dass die Pflege des Programmmoduls, welches das gemeinsame Lösungsverfahren realisiert, und die Pflege der zugehörigen Daten nur einem der betrachteten Unternehmen obliegt. Das AwS dieses Unternehmens wird als Quell-AwS bezeichnet.

Bei dieser in Bild 5~~Bild 5~~ dargestellten Kopplungsarchitektur verfügt jedes der betrachteten AwS über eine eigene Kopie des Programmmoduls und führt diesen in einem eigenen Prozess auf einer eigenen Kopie der gemeinsamen Daten aus. Durch dieses Konzept wird eine relativ hohe Betriebsunabhängigkeit erreicht. Aufgrund der redundanten Haltung des Programmmoduls stellt sich hier allerdings das Problem der Propagation von Änderungen am Modul. Das Problem der gemeinsam genutzten Daten wird mit der in Abschnitt 3.2.1 behandelten datenorientierten Kopplungsarchitektur mit redundanter Datenhaltung gelöst. In Bild 5~~Bild 5~~ ist diese Teil-Kopplungsarchitektur in Form des datenorientierten Kopplungs-Subsystems angedeutet.

hat formatiert: Schriftart: Nicht Fett

hat formatiert: Schriftart: Nicht Fett

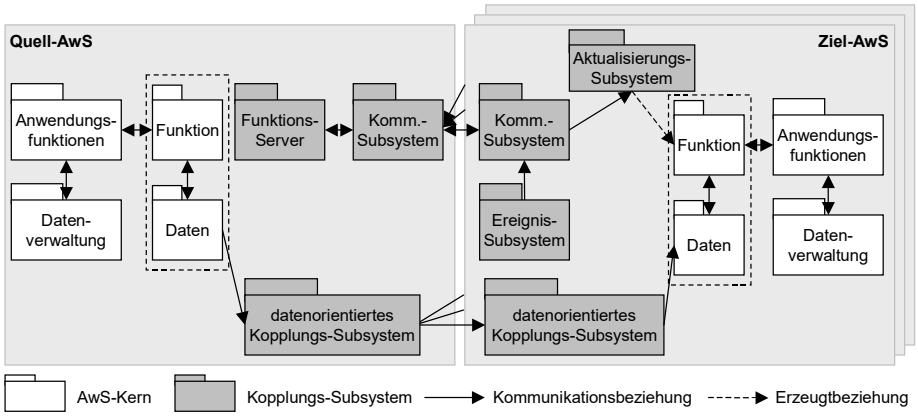

Bild 5 Funktionsorientierte Kopplungsarchitektur mit redundanter Modulhaltung, separaten Prozessen und redundanter Datenhaltung

Hinsichtlich der Propagation von Änderungen in der Realisierung des Lösungsverfahrens wird in Bild 5~~Bild 5~~ von einem Pull-Prinzip ausgegangen. Das Ziel-AwS fragt in bestimmten Zeitabständen beim Quell-AwS an, ob neuere Versionen des Programmmoduls vorliegen. Das **Ereignis-Subsystem** bestimmt die Zeitpunkte, zu denen eine Anfrage erfolgen soll und stößt diese an. In der Anfrage wird das angeforderte Modul und die aktuell genutzte Version spezifiziert. Die Aktualisierung des Moduls muss mit dem Betrieb des AwS koordiniert werden.

hat formatiert: Schriftart: Nicht Fett

Der **Funktionsserver** ist das zentrale Subsystem dieser Kopplungsarchitektur auf Seite des Quell-AwS. Er verwaltet die zu replizierenden Module. Im Fall einer Anfrage durch ein Ziel-AwS vergleicht der Funktionsserver die von diesem AwS genutzte Version mit der aktuell verfügbaren Version. Stellt er fest, dass die genutzte Version veraltet ist, so übermittelt er unter Verwendung des Kommunikations-Subsystems die aktuelle Version an das Ziel-AwS.

Auf Seite des Ziel-AwS nimmt das **Aktualisierungs-Subsystem** die neue Version des Moduls entgegen und ersetzt damit das bisher verwendete Modul.

3.3.2 Funktionsorientierte Kopplungsarchitektur mit nicht-redundanter Modulhaltung, gemeinsamem Prozess und nicht-redundanter Datenhaltung

Die in Bild 6~~Bild 6~~ dargestellte Kopplungsarchitektur behandelt die gleiche Problemstellung wie die vorherige Kopplungsarchitektur. Das Problem der gemeinsamen Nutzung einer Funktion wird hier aber dadurch behandelt, dass alle gekoppelten AwS eine gemeinsame Kopie des Programmmoduls und der Daten sowie einen gemeinsamen Prozess nutzen. Zentrales Subsystem der Architektur ist ein **Funktionsserver**, der Funktionen verwaltet und nach außen anbietet. Wesentliche Aufgaben dieses Funktionsservers sind

hat formatiert: Schriftart: Nicht Fett

- die Weitergabe von Aufrufen an die jeweiligen Funktionen,

- die Koordination paralleler Zugriffe
- und die Zugriffskontrolle.

Als Basismaschine für die Realisierung eines solchen Funktionsservers kann z.B. ein TP-Monitor (vgl. zu TP-Monitoren z.B. [GrRe94]) eingesetzt werden. Auch Laufzeitumgebungen von Komponententechnologien wie z.B. CORBA, COM+ oder EJB können hierfür verwendet werden.

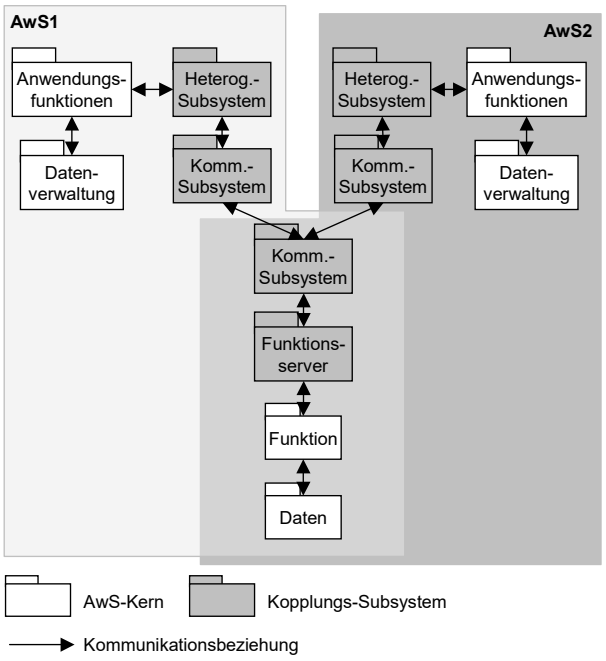

Bild 6 Funktionsorientierte Kopplungsarchitektur mit nicht-redundanter Modulhaltung, gemeinsamem Prozess und nicht-redundanter Datenhaltung

Als weitere Subsysteme übermitteln die **Kommunikations-Subsysteme** Anfragen und die Ergebnisse der Anfragen zwischen den dezentralen AwS-Kernen und dem zentralen gemeinsam genutzten AwS-Kern. Bei Bedarf bewältigen **Heterogenitäts-Subsysteme** die fachliche Heterogenität bzgl. der Anfragen und der Ergebnisse.

4. Kopplungsmechanismen in SAP R/3

Zur Realisierung der beschriebenen Kopplungsarchitekturen werden Kopplungsmechanismen verwendet, die Dienste und Kommunikationsprotokolle für die Realisierung der Kopplungsarchitekturen bereitstellen. Die Kopplungsmechanismen nutzen ihrerseits Dienste von Betriebssystemen, Kommunikationsplattformen, Web Server etc. Sie beinhalten systemtechni-

sche und fachliche Aspekte und geben so einen engen Rahmen für die Entwicklung einer Kopplungsarchitektur vor. Im Folgenden werden ausgewählte Kopplungsmechanismen der Standardsoftware R/3 von SAP vorgestellt und hinsichtlich ihrer Eignung zur Realisierung der beschriebenen Kopplungsarchitekturen diskutiert.

4.1 BAPI (Business Application Programming Interface)

Der BAPI-Kopplungsmechanismus stellt allgemeine Dienste für die Kommunikation von AwS bereit. Er bietet eine objektorientierte Schnittstelle, über die ein Zugriff auf die Funktionen einer R/3-Komponente und auf die durch die R/3-Komponente verwalteten Daten möglich ist. Der Zugriff kann hierbei durch weitere R/3-Komponenten erfolgen, aber auch durch Nicht-R/3-Systeme. 1996 führte die SAP als Basiskonzept für die Gesamtarchitektur von R/3 die **Business Framework Architecture** (BFA) ein (vgl. zur BFA [SAP01c, 6ff]). Sie gibt einen Rahmen für die interne Strukturierung von R/3 und für die Anbindung von AwS an R/3 vor. Die BFA teilt R/3 in **Business Components** auf. Diese sollen getrennt installierbar und weiterentwickelbar sein. Eine Business Component besteht aus einer Menge von **Business Objects**, die fachliche Entitäten wie z.B. Kundenaufträge repräsentieren. Sie kapseln die zur Beschreibung einer Entität benötigten Daten und die auf diesen Daten operierenden Funktionen. Die Funktionen eines Business Object werden hierbei als Methoden bezeichnet. Die Schnittstelle einer Business Component wird durch die in ihr enthaltenen Business Objects definiert (vgl. Bild 7~~Bild 7~~). Methoden eines Business Object, die von außerhalb der zugehörigen Business Component aufgerufen werden können, werden als BAPI bezeichnet.

hat formatiert: Schriftart: Nicht Fett, Deutsch (Deutschland)

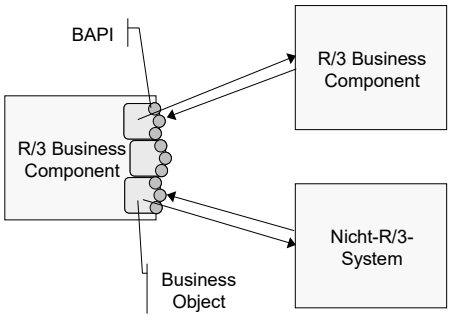

Bild 7 Elemente der Business Framework Architecture

R/3 wird bereits mit einer ganzen Reihe von Business Objects ausgeliefert. Diese können durch die Kunden modifiziert und erweitert werden. Außerdem können mit Hilfe der R/3-Entwicklungsumgebung neue Business Objects implementiert werden. Die Definition eines Business Object in R/3 wird als **Objekttyp** bezeichnet (vgl. zu Objekttyp [SAP01c, 10]). Die BAPI werden in Form von Funktionsbausteinen realisiert. Hierbei handelt es sich um spezielle Programmmodule, die mit der SAP eigenen Programmiersprache ABAP/4 realisiert wer-

den. Der Zugriff auf die BAPI einer R/3-Komponente kann auf zwei unterschiedliche Arten erfolgen (vgl. [SAP01c, 26ff]):

- funktionsorientiert oder
- objektorientiert.

Bei der funktionsorientierten Variante wird der zugehörige Funktionsbaustein via RFC direkt aufgerufen. Der RFC ist die R/3-Variante des Remote Procedure Calls. Beim objektorientierten Zugriff wird nicht der Funktionsbaustein via RFC aufgerufen sondern das Business Object Repository (BOR), das die Objekttypen verwaltet (vgl. zum BOR [SAP01c,17]). Dieses leitet den Aufruf entsprechend weiter. Im aufrufenden System wird hierbei typischerweise ein Proxy-Objekt verwendet, das die Methodenaufrufe entgegennimmt und dann den RFC-Aufruf durchführt (vgl. zu Proxy-Objekt z.B. [GHJV96, 227ff]).

Da der BAPI-Kopplungsmechanismus allgemein gehalten ist, kann er grundsätzlich zur Realisierung aller in Kapitel 3 beschriebenen Arten von Kopplungsarchitekturen eingesetzt werden. Hierbei sind allerdings teilweise erhebliche Erweiterungen erforderlich. Soll auf Basis des BAPI-Kopplungsmechanismus eine **ereignisorientierte Kopplungsarchitektur** gemäß der in Abschnitt 3.1 beschriebenen Ausprägung realisiert werden, so sind insbesondere die Funktionen von Ereignis-, Empfänger- und Heterogenitäts-Subsystem zu ergänzen.

Die Realisierung einer **datenorientierten Kopplungsarchitektur mit redundanter Datenhaltung** wird durch den BAPI-Kopplungsmechanismus nur sehr unzureichend unterstützt. Besser geeignet ist hierfür der von SAP zusätzlich angebotene **ALE-Kopplungsmechanismus**, der eine Erweiterung des BAPI-Kopplungsmechanismus darstellt und in Abschnitt 4.2 genauer behandelt wird. Eine **datenorientierte Kopplungsarchitektur mit nicht-redundanter Datenhaltung** hingegen lässt sich mit Hilfe des BAPI-Kopplungsmechanismus relativ einfach realisieren. Die gemeinsam genutzten Daten werden hierbei innerhalb des R/3-Systems verwaltet. Damit ein externes AwS auf diese Daten zugreifen kann, muss ein entsprechendes BAPI realisiert werden. Defizite bestehen allerdings auch hier in der fehlenden Unterstützung bei der Bewältigung der fachlichen Heterogenität.

Der BAPI-Kopplungsmechanismus kann auch zur Realisierung einer funktionsorientierten Kopplungsarchitektur eingesetzt werden. Allerdings erweist sich die Realisierung einer **funktionsorientierten Kopplungsarchitektur mit redundanter Modulhaltung, separaten Prozessen und redundanter Datenhaltung** unter alleiniger Verwendung des BAPI-Kopplungsmechanismus als relativ aufwändig. So müsste der in Abschnitt 3.3.1 beschriebene Mechanismus zur Propagation von Programmmodulen ergänzt werden, da er nicht direkt unterstützt wird. Ebenfalls unzureichend unterstützt wird die bei redundanter Datenhaltung notwendige Propagation von Datenänderungen. Weitestgehend unproblematisch ist hingegen die Realisierung einer **funktionsorientierten Kopplungsarchitektur mit nicht-redundanter Modulhaltung, gemeinsamem Prozess und nicht-redundanter Datenhaltung**. Die ge-

meinsam genutzte Funktion und der zugehörige Datenobjekttyp werden hierbei innerhalb des R/3-Systems in Form eines Objekttyps realisiert. Der Zugriff auf den Objekttyp erfolgt über entsprechende BAPI. Problematisch ist hier wiederum nur die fehlende Unterstützung bei der Bewältigung der fachlichen Heterogenität.

4.2 ALE (Application Link Enabling)

Mit dem ALE-Kopplungsmechanismus bietet R/3 seit Release 3.0 einen Mechanismus zur Realisierung einer **datenorientierten Kopplungsarchitektur mit redundanter Datenhaltung** an (vgl. zu ALE [SAP01a; Nagp99]). Der ALE-Kopplungsmechanismus unterstützt die redundante Haltung von Konfigurations-, Stamm- und Bewegungsdaten und schafft somit die Voraussetzungen zur Aufteilung einer R/3-Installation in mehrere Teil-Installationen, die auf getrennten Rechnern betrieben werden können. Neben der Synchronisation von Datenbeständen innerhalb einer R/3-Installation wird auch eine Synchronisation mit externen AwS unterstützt. Der ALE-Kopplungsmechanismus geht generell davon aus, dass Daten von genau einem AwS gepflegt werden. Das in Abschnitt 3.2.1 angesprochene Problem der Manipulation von Daten durch mehrere AwS wird somit nicht behandelt.

Der ALE-Mechanismus ist in zahlreichen Varianten verfügbar. Insbesondere ist eine Unterscheidung nach dem verwendeten Basismechanismus möglich. Ursprünglich handelte es sich hierbei um eine Erweiterung des IDoc-Kopplungsmechanismus (vgl. zu IDoc [Nagp99, 547ff]). Mittlerweile kann neben dem IDoc-Kopplungsmechanismus auch der im vorherigen Abschnitt beschriebene BAPI-Kopplungsmechanismus als Basis eingesetzt werden (vgl. [SAP01b, 26f; SAP01a]). Die folgende Diskussion beschränkt sich auf die Betrachtung einer Variante des Mechanismus, die sich mit der Synchronisation von Stammdaten unter Verwendung der BAPIs mit Hilfe des SMD-Tool (Shared Master Data) beschäftigt (vgl. [SAP01a]).

Bei der betrachteten Variante bestimmt das Quell-AwS, wann eine Propagation zu erfolgen hat. Im Gegensatz zu dem in Bild 3~~Bild 3~~ dargestellten Ansatz erfolgt hier die Propagation in Abhängigkeit vom Eintreten von Ereignissen im AwS-Kern. Das Ereignis-Subsystem wird hierbei durch den **Änderungsbelegmechanismus** und den **Änderungszeigermechanismus** abgedeckt. Der Änderungsbelegmechanismus wird nicht nur im Umfeld von ALE verwendet. Mit ihm können allgemein Datenbankänderungen in Form von Änderungsbelegen protokolliert werden. Der Änderungszeigermechanismus setzt auf diesen Belegen auf und entscheidet, ob die protokollierten Änderungen relevant sind für die Propagation. Ist dies der Fall, so werden entsprechende Merker in Form von Änderungszeigern angelegt. Diese können anschließend verarbeitet werden. Da mit Hilfe der Änderungszeiger genau festgehalten wird, welche Daten geändert wurden, ist eine Beschränkung auf eine Propagation der tatsächlich veränderten Daten möglich.

hat formatiert: Schriftart: Nicht Fett

Der ALE-Kopplungsmechanismus beinhaltet auch ein Subsystem, das sich mit der Ermittlung von Empfängern beschäftigt. Im Mittelpunkt dieses Subsystems steht das **ALE Verteilungsmodell**. Dieses bietet sog. Filterobjekttypen an, mit denen Regeln definiert werden können, die angeben, an welche Ziel-AwS die Daten zu propagieren sind. Diese Regeln können sich auf den Inhalt der zu propagierenden Daten beziehen. Die Bewältigung der fachlichen Heterogenität wird durch den ALE-Kopplungsmechanismus in Form der **Feldumsetzung** unterstützt. Die Aufgaben des in Bild 3~~Bild 3~~ dargestellten Aktualisierungs-Subsystems werden zum Teil durch die **ALE Objektserialisierung** abgedeckt. Mit der Objektserialisierung kann gewährleistet werden, dass alle Nachrichten zu einem Objekt in den Ziel-AwS in der gleichen Reihenfolge verarbeitet werden, in der sie im Quell-AwS erzeugt wurden.

hat formatiert: Schriftart: Nicht Fett

4.3 SAP Business Workflow / WebFlow

R/3 enthält in Form des SAP Business Workflow bereits seit Release 3.0 ein eigenes Workflow-Management-System (WFMS), mit dem betriebswirtschaftliche Abläufe in R/3 flexibel abgebildet werden können. Die Abläufe werden hierbei in Form von **Workflow-Definitionen** erfasst, die eine Menge von sogenannten Einzelschrittaufgaben umfassen (vgl. [BMS99, 18]). Bei der Ausführung einer Workflow-Definition in Form eines Workflow greift der SAP Business Workflow zur Durchführung der Einzelschrittaufgaben auf die bereits angesprochenen **Objekttypen** zurück.

Der SAP Business Workflow war zunächst auf Workflows innerhalb eines R/3-Systems beschränkt. Später wurde er mit WebFlow um Mechanismen erweitert, die eine Unterstützung von Workflows über die Grenzen eines R/3-Systems hinaus ermöglichen. WebFlow unterstützt unter anderem folgende Formen der Interaktion zwischen einem R/3-System und einem externen AwS, das kein R/3-System sein muss:

- Starten eines Workflow in einem R/3-System von einem externen AwS aus und anschließende Übertragung der Ergebnisse des Workflow an das externe AwS.
- Starten eines Workflow in einem externen AwS aus einem Workflow heraus, der in einem R/3-System läuft, und anschließende Entgegennahme der Ergebnisse des Workflow.

Bei der Realisierung von WebFlow orientierte sich die SAP an dem von der Workflow Management Coalition (WFMC) entwickelten **Wf-XML Standard** (vgl. [WFMC00]) zur Erreichung einer Interoperabilität von Workflow Engines. Mit dem Begriff Workflow Engine wird hierbei das Subsystem eines WFMS bezeichnet, das die eigentliche Ausführungsumgebung für die Workflows bereitstellt. R/3 unterstützt im Release 4.6C die Beta-Fassung des Standards vom 11. Januar 2000 (vgl. [SAP01a]). Die Standards der WFMC unterscheiden zwei Formen der Interaktion (Models of Interoperability). Bei **geschachtelten Prozessen** (Nested

processes) wartet ein Workflow[3], der den Start eines anderen Workflow verursacht hat, auf dessen Ergebnisse. Bei **verketten Prozessen** (Chained processes) ist dies nicht der Fall. Nachdem der angeforderte Workflow gestartet wurde, läuft der anfordernde Workflow weiter oder wird beendet (vgl. [WFMC96, 13]).

Zur Realisierung dieser Interaktionsformen müssen die Workflow Engines, die die Workflows erzeugen und ausführen, an ihren Schnittstellen verschiedene Operationen unterstützen, die eine entsprechende Kommunikation ermöglichen. In [WFMC00] werden hierfür insgesamt vier Operationen definiert sowie ihre Realisierung unter Verwendung von XML und HTTP beschrieben. Mit der Operation **CreateProcessInstance** kann die Erzeugung und das Ausführen eines Workflow angefordert werden. Mit **GetProcessInstanceData** können Informationen über einen Workflow angefordert werden. Mit **ChangeProcessInstanceState** kann der Zustand eines Workflow geändert werden. So kann z.B. mit dieser Operation ein Workflow vorübergehend angehalten werden. Die letzte Operation **ProcessIntanceStateChanged** erlaubt es einer Workflow Engine, die einen Workflow ausführt, anderen Systemen Änderungen des Zustands dieses Workflow mitzuteilen. Insbesondere kann mit dieser Operation eine Workflow Engine, die die Erzeugung des Workflow angefordert hatte, informiert werden, wenn der Workflow beendet wurde. Die Ergebnisse des Workflow können dabei auch übertragen werden.

Der SAP Business Workflow kann als Kopplungsmechanismus verstanden werden, der sich mit dem Austausch von Ereignissen und zugehörigen Daten beschäftigt. In Kombination mit WebFlow kann der SAP Business Workflow folglich zur Realisierung einer überbetrieblichen AwS-Kopplung in Form einer **ereignisorientierten Kopplungsarchitektur** genutzt werden. Ausgehend von der in Abschnitt 3.1 beschriebenen ereignisorientierten Kopplungsarchitektur handelt es sich beim SAP Business Workflow vor allem um eine Erweiterung der Empfänger-Subsysteme. Die dort beschriebenen Empfänger-Subsysteme sind relativ einfach gehalten. Sie sind zustandslos und treffen somit ihre Entscheidungen bzgl. der Weiterleitung der Ereignisse unabhängig von vorher ausgetauschten Ereignissen. Jeder Austausch eines Ereignisses wird einzeln betrachtet. Der SAP Business Workflow erweitert diese einfachen Empfänger-Subsysteme um die typischen Funktionen eines WFMS zur Erstellung und Ausführung von Workflow-Definitionen. Die Ausführung beinhaltet hierbei die Weiterleitung von Ereignissen und Daten unter Berücksichtigung des aktuellen Zustands des zugehörigen Workflow. Durch diese Erweiterungen wird nun die Unterstützung komplexer überbetrieblicher Prozesse möglich, in deren Rahmen häufig nicht nur einzelne Ereignisse sondern komplexe Sequenzen zusammengehöriger Ereignisse ausgetauscht werden.

Das Ereignis-Subsystem kann durch die in R/3 enthaltene **Ereignisverwaltung** abgedeckt werden. Diese ermöglicht die Überwachung der Zustände von Objekten und die Meldung von

[3] Der in R/3 verwendetet Begriff Workflow entspricht in den Standards der WFMC dem Begriff Prozess-Instanz.

Zustandsänderungen an verschiedene Ereignisverbraucher (vgl. [BMS99, 145ff]). Als Ereignisverbraucher können hierbei Workflows oder Einzelschrittaufgaben auftreten. Ein Workflow kann beim Eintreten eines Ereignisses gestartet werden, eine Einzelschrittaufgabe kann durch das Eintreffen eines Ereignisses beendet werden. Ein Subsystem zur Bewältigung der fachlichen Heterogenität ist im SAP Business Workflow allerdings nicht vorgesehen.

5. Zusammenfassung und Ausblick

Kopplungsarchitekturen sind Realisierungskonzepte für die Integration von AwS. Im Mittelpunkt des vorliegenden Beitrags stehen eine Analyse und Systematisierung von Kopplungsarchitekturen für die überbetriebliche Integration von AwS sowie eine Diskussion ihrer Realisierung mithilfe von Kopplungsmechanismen, wie sie von SAP R/3 angeboten werden.

Die Notwendigkeit zur Kopplung von AwS ergibt sich aus den Beziehungen zwischen den von den AwS unterstützten Aufgaben. Im vorliegenden Beitrag wurden die Aufgabenbeziehungen Reihenfolgebeziehung, partielle Identität von AO-Instanzen, partielle Gleichheit von AO-Typen und partielle Gleichheit von Lösungsverfahren unterschieden. Die der Unterstützung dieser Beziehungen dienenden Kopplungsarchitekturen wurden in die drei Klassen ereignis-, daten- und funktionsorientierte Kopplungsarchitekturen unterteilt.

Die hier vorgestellten Ausprägungen von Kopplungsarchitekturen orientieren sich im Wesentlichen am klassischen Paradigma von Daten und darauf operierenden Funktionen sowie einer Kopplung von Funktionen über gemeinsame Daten oder den Austausch von Nachrichten. Dieses Paradigma bestimmt nach wie vor viele Produkte und Gestaltungsvorschläge in der Praxis. Die beschriebenen Kopplungsarchitekturen sind vor allem auf die Integrationskonzepte [FeSi01, 215ff] datenflussorientierte Funktionsintegration und Datenintegration ausgerichtet, unterstützen aber auch Integrationsziele, wie sie speziell durch das Integrationskonzept Objektintegration verfolgt werden.

In weiteren Arbeiten soll untersucht werden, wie sich eine konsequente Ausrichtung am Paradigma der Objektorientierung auf die Gestaltung von Kopplungsarchitekturen und auf die damit erreichbare Unterstützung der einzelnen Integrationskonzepte auswirkt. Die notwendigen Kopplungsmechanismen hierfür sind in Form von objektorientierten Basistechnologien zunehmend verfügbar.

Literatur

[Beck91] *Becker, J.:* CIM-Integrationsmodell – Die EDV-gestützte Verbindung betrieblicher Bereiche. Springer, Berlin 1991.

[BHG87] *Bernstein, P. A.; Hadzilacos, V.; Goodman, N.:* Concurrency Control and Recovery in Database Systems. Addison-Wesley, Reading 1987.

[BMS99] *Berthold, A.; Mende. U.; Schuster, H.:* SAP Business Workflow – Konzept, Anwendung, Entwicklung. Addison-Wesley, München 1999.

[FeSi01] *Ferstl, O. K.; Sinz, E. J.:* Grundlagen der Wirtschaftsinformatik. 4. Auflage, Oldenbourg, München 2001.

[GHJV96] *Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.:* Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software. Addison-Wesley, Bonn 1996.

[GrRe94] *Gray, J.; Reuter, A.:* Transaction processing - concepts and techniques. 2. Auflage, Kaufmann, San Mateo 1994.

[Krem99] *Kremer, R.:* Replikatives Informationsmanagement in verteilten Groupware-Umgebungen. Shaker Verlag, Aachen 1999.

[Lint01] *Linthicum, D. S.:* B2B Application Integration – e-Business-Enable Your Enterprise. Addison-Wesley, Boston 2001.

[Mert00] *Mertens, P.:* Integrierte Informationsverarbeitung 1 – Administrations- und Dispositionssysteme in der Industrie. 12. Auflage, Gabler, Wiesbaden 2000.

[Nagp99] *Nagpal, A.:* ALE, EDI, & IDoc Technologies for SAP. Prima, Rocklin California 1999.

[Rahm94] *Rahm, E.:* Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenbankverarbeitung. Addison-Wesley, Bonn 1994.

[SAP01a] *SAP AG:* R/3 Online Help Release 4.6C. 2001.

[SAP01b] *SAP AG:* BAPI-Programmierleitfaden Release 46C. https://www012.sap-ag.de/~sapidb/011000358700004269971999d.doc, Abruf am 2002-04-07. (nur für geschlossene Benutzergruppe zugänglich)

[SAP01c] *SAP AG:* BAPI-Benutzerhandbuch. https://www012.sap-ag.de/~sapidb/011000358700001724561999d.doc, Abruf am 2002-04-07. (nur für geschlossene Benutzergruppe zugänglich)

[Sche90] *Scheer, A.-W.:* CIM Computer Integrated Manufacturing – Der computergesteuerte Industriebetrieb. 4. Auflage, Springer, Berlin 1990.

[WFMC00] *Workflow Management Coalition:* Workflow Management Coalition Workflow Standard – Interoperability Wf-XML Binding – Document Number WFMC-TC-1023 Version 1.0. 2000.

[WFMC96] *Workflow Management Coalition:* Workflow Management Coalition Workflow Standard – Interoperability Abstract Specification – Document Number WFMC-TC-1012 Version 1.0. 1996.