Universität Ulm
Fakultät für Informatik

Einsatz von Workflow Engines zur effizienten Anpassung standardisierter Branchensoftware an unterschiedliche Kundenanforderungen

Masterarbeit

Martin Steinle

Betreuer:

Prof. Dr. Peter Dadam, Hilmar Acker, Abt. Datenbanken und Informationssysteme

Martin Staib, Marc Graner, Fritz und Macziol GmbH, Ulm

Korrektoren:

Prof. Dr. Peter Dadam, Abt. Datenbanken und Informationssysteme

Dr. Manfred Reichert, TU Twente (NL), Abt. Information Systems

Abgabedatum: 25.10.2005

Erklärung

Hiermit erkläre ich, Martin Steinle, dass ich diese Masterarbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die Grundsätze und Empfehlungen „Verantwortung in der Wissenschaft“ der Universität Ulm beachtet habe.

Ulm, den 25.10.2005

Martin Steinle

Danksagung

An erster Stelle möchte ich meinen Betreuern Prof. Dr. Peter Dadam und Hilmar Acker danken, die mich in vielen Diskussionen weitergebracht und stets tatkräftig unterstützt haben. Dank gilt auch Ulrich Kreher, der mir bei allen Fragen zum ADEPT-Prototypen weitergeholfen hat. Ebenso möchte ich Manfred Reichert dafür danken, dass er sich bereit erklärt hat, diese Arbeit zu korrigieren.

Mein besonderer Dank gilt Martin Staib und Marc Graner für die Betreuung meiner Arbeit seitens der Firma Fritz und Macziol. Ihnen und der gesamten Abteilung Industrial Applications danke ich dafür, dass sie stets ein offenes Ohr für meine Fragen hatten und immer für ein positives Arbeitsumfeld sorgten. Ein spezielles Dankeschön geht an Nico Przybylek für das Korrekturlesen dieser Arbeit und die vielen hilfreichen Anmerkungen.

Ulm, den 25.10.2005

Martin Steinle

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Ausgangssituation

Ausgangssituation dieser Masterarbeit ist das Versandsystem VAS (Versandautomation und Absicherung) der Firma Fritz & Macziol, das zur Automatisierung und Absicherung aller Versand- und Anliefervorgänge in Werken der Schüttgutindustrie eingesetzt wird. Es verwaltet alle Daten von Aufträgen, Versand-Lieferungen und Anlieferungen sowie Stammdaten von Waren, Kunden, Spediteuren und Lkw, die ins Werk kommen. Hauptaufgabe von VAS ist, Versand- und Anliefervorgänge zu automatisieren, sodass möglichst wenige (im Idealfall keine) Werksmitarbeiter benötigt werden. Dazu existieren Schnittstellen zu den Anlagen des Werks, wie Beladeanlagen, Fahrzeugwaagen oder Identifikationssystemen. In vielen Werken sind Versand und Anlieferungen rund um die Uhr möglich, weshalb auch VAS rund um die Uhr verfügbar sein muss. Dies wird unter anderem dadurch erreicht, dass zwei redundante zentrale Server existieren.

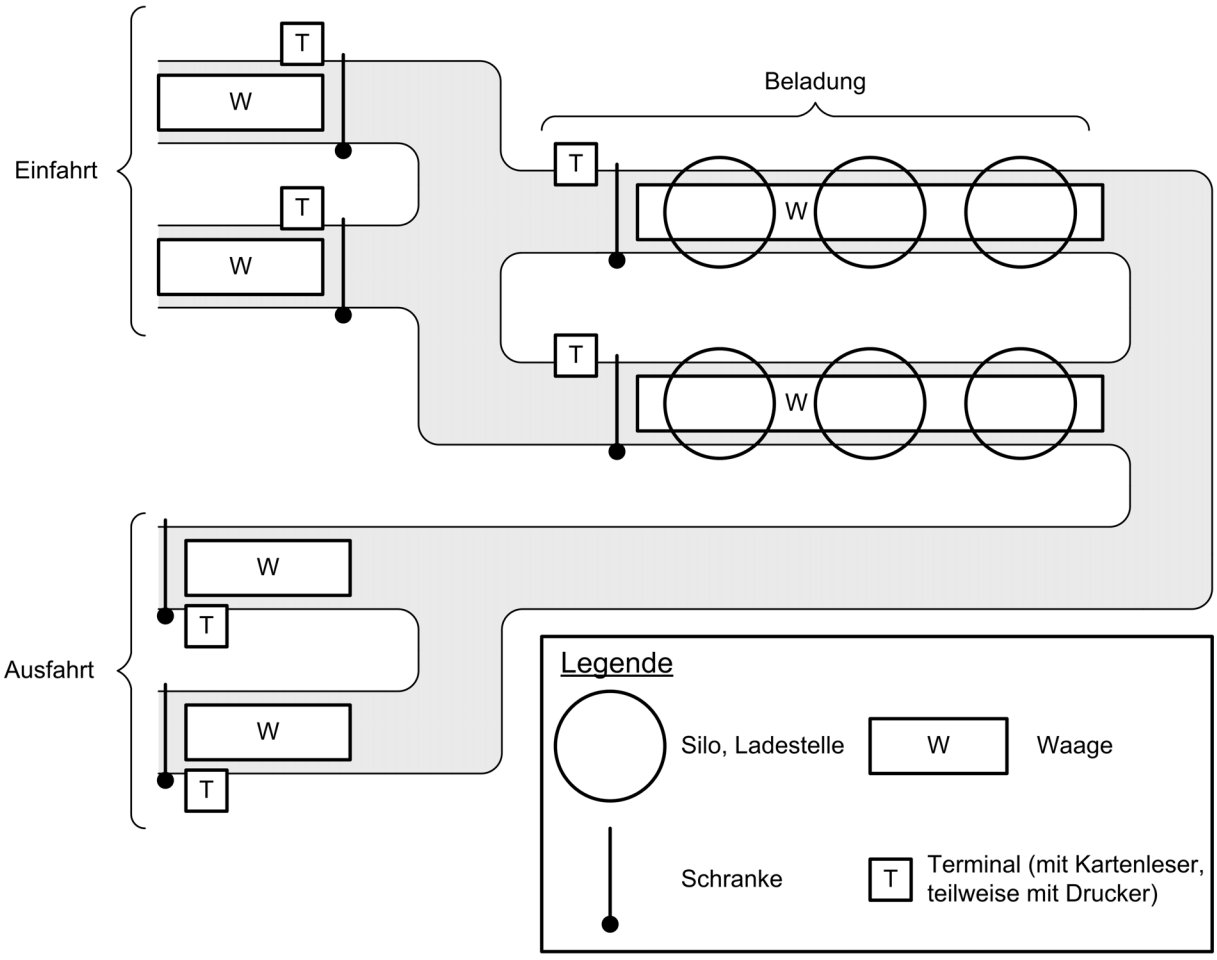

Abbildung 1-1: Schematischer Werksaufbau (stark vereinfacht)

Waren, die versendet werden, sind häufig lose (z.B. Zement) und werden in Tankfahrzeugen transportiert. Aber auch Waren, die in Säcken oder auf Paletten verpackt sind, können mit Hilfe von VAS verwaltet werden. Der Aufbau eines Werks ist in Abbildung 1-1 beispielhaft und stark vereinfacht dargestellt.

Der Ablauf des Werksbesuchs eines Lkw ist ungefähr folgendermaßen. Der Lkw fährt zunächst auf die Waage an einer Einfahrtsstelle und identifiziert sich am dortigen Terminal z.B. durch Eingabe von Benutzername und Passwort oder mit einer Chipkarte. Dann gibt er an, welche Waren er anliefert bzw. welche Waren er abholen möchte. Das Gewicht des Lkw wird registriert, es werden Ladepapiere für ihn ausgedruckt und er erhält die Information, zu welcher Ladestraße er fahren muss. Ist dies alles erfolgreich abgeschlossen, wird die Schranke geöffnet und er kann ins Werk fahren. An der Ladestraße angekommen, muss er sich wieder identifizieren. Dadurch kann VAS auf die Daten zugreifen, die bei der Einfahrt angegeben wurden und weiß somit, mit welcher Ware und in welcher Menge der Lkw beladen werden soll. Er fährt unter das Silo, das ihm von VAS angezeigt wird. Dort wird der Lkw automatisch (bzw. unter Mithilfe des Fahrers) mit der richtigen Menge beladen, woraufhin er entweder zur nächsten Ladestraße (falls er noch weitere Waren mitnehmen will) oder zu einer Ausfahrtstelle fahren kann. Wird statt loser Ware verpackte verladen, werden Lkw von einem Werksmitarbeiter beladen. Dieser hat entweder ein Terminal, an dem er herausfinden kann, welche Waren in welcher Menge auf einen Lkw geladen werden sollen, oder er erfährt dies von den Ladepapieren des Lkw. An der Ausfahrt muss sich der Lkw wieder identifizieren und wird zur Kontrolle nochmals gewogen. Durch den Vergleich mit dem Einfahrtsgewicht wird erkannt, wie viel er zugeladen hat. Ist die Menge korrekt, wird ein Lieferschein gedruckt und die Schranke geöffnet, sodass der Lkw das Werk verlassen kann. Hat der Lkw zu wenig oder zu viel geladen, muss er zurück ins Werk und entsprechend nachladen bzw. abladen.

Sämtliche Stationen im Werk können auch durch Werksmitarbeiter durchgeführt werden. Dazu werden die Terminals an den verschiedenen Stellen im Werk nicht verwendet. Ein- und Ausfahrt eines Lkw werden von Büroarbeitsplätzen in einem zentralen Versandbüro aus gesteuert, von wo VAS ebenfalls Schranken und Waagen ansteuern kann.

Zwischen den einzelnen Unternehmen, in denen VAS eingesetzt ist, bestehen große Unterschiede. So werden in den Werken mancher Unternehmen nur Teile des Versands durch VAS gesteuert. Andere Teile (z.B. Anlieferungen oder Versand bestimmter Warentypen) werden ohne Unterstützung durch VAS abgewickelt. In manchen Unternehmen beschränkt sich die Unterstützung von VAS auf die Einfahrts- und Ausfahrtsabwicklung von Fahrzeugen. Weiter unterscheiden sich die Unternehmen durch die Zusammenarbeit von VAS mit vorhandenen ERP-Systemen. In manchen Unternehmen erhält VAS alle Stamm- und Vorgangsdaten vom ERP-System, an das es abgewickelte Vorgänge zurückmeldet. In anderen Unternehmen ist es möglich, direkt in VAS Stammdaten zu verändern und Vorgänge anzulegen. Zu den einzelnen Unterschieden zwischen den Unternehmen siehe auch Kapitel 2.

1.2 Ziel der Arbeit

Durch die großen Unterschiede zwischen Unternehmen muss VAS an jedes Unternehmen angepasst werden. Zwar sind einzelne, kleine Module so generisch gehalten, dass sie in der Regel unverändert wieder verwendet werden können, doch vor allem die mitunter sehr komplexen Abläufe müssen meistens komplett neu implementiert werden. Dies verursacht sehr hohen Aufwand, da in der aktuellen Version jeder Prozessschritt seinen Nachfolger direkt bedingt. Dadurch ist die Definition der Abläufe über den kompletten Programmcode von VAS verteilt, und nur mit VAS vertraute Entwickler können neue Abläufe implementieren oder bereits bestehende ändern.

Auch nach der Inbetriebnahme sind von Zeit zu Zeit Anpassungen von VAS notwendig, da immer wieder neue Funktionen gewünscht oder Abläufe umgestaltet werden. Auch bei Ausfällen bestimmter Werksanlagen kann eine Anpassung von VAS dazu beitragen, den Betrieb aufrecht zu erhalten. Auch diese Anpassungen können in der Regel nur von VAS-Entwicklern durchgeführt werden, und vor allem muss oft auch für kleine Änderungen VAS neu gestartet werden, sodass dieses bei Änderungen immer eine Zeit lang nicht verfügbar ist.

Ziel dieser Arbeit ist, die genannten Probleme zu lösen bzw. zu entschärfen. Dazu sollen drei Teilziele erreicht werden:

- Verringerung des Anpassungsaufwands für die Inbetriebnahme neuer Werke, um VAS schneller und kostengünstiger an neue Werke anpassen zu können.
- Verringerung der Anforderungen an Personen, die Anpassungen vornehmen, sodass nicht nur VAS-Entwickler, sondern im Idealfall auch Kunden möglichst viele Änderungen vornehmen können.
- Ermöglichung von Veränderungen zur Laufzeit des Systems, sodass Änderungen auch ohne Systemneustart wirksam werden.

1.3 Lösungsansatz: Einsatz von Workflow-Engines

In dieser Arbeit soll evaluiert werden, inwiefern der Einsatz von Workflow-Engines zum Erreichen der im letzten Absatz genannten Ziele beitragen kann. Dabei soll herausgefunden werden, welche Verbesserungen der Einsatz von Workflow-Engines liefern kann und welche Nachteile eventuell dafür in Kauf genommen werden müssen.

Eine Workflow-Engine ist zentraler Bestandteil eines Workflow-Management-Systems. Ein Workflow-Management-System dient dazu, vorgegebene Arbeitsabläufe mit mehreren Teilnehmern zu verwalten [Jabl95]. Aufgabe der Workflow-Engine ist die Abwicklung solcher Arbeitsabläufe, *Prozesse* bzw. *Prozessinstanzen* (engl. Workflow) genannt. Eine weitere Komponente eines Workflow-Management-Systems ist ein Werkzeug zur Prozessmodellierung, mit dem *Prozessvorlagen* (auch

Prozessschemata oder *Prozessdefinitionen* genannt) definiert (oft graphisch modelliert) und der Workflow-Engine zur Ausführung übergeben werden können. Üblicherweise besteht ein Workflow-Management-System außerdem aus einem Werkzeug zur Überwachung und Administration von Prozessen sowie aus *Workflow-Client-Anwendungen*, die die Schnittstelle zwischen Workflow-Engine und Benutzern darstellen.

Prozessvorlagen werden auf der Basis eines fest definierten Prozess-Metamodells erstellt. Dieses legt die zulässigen Prozesse und möglichen Operationen darauf fest. Eine Prozessvorlage definiert alle Schritte, *Aktivitäten* genannt, die zur Ausführung eines Prozesses notwendig sind, sowie deren Abfolge. Zusätzlich wird für jede Aktivität definiert, durch welche Mitarbeiter sie ausgeführt werden kann und welche Daten sie als Eingabeparameter zur Ausführung erhält bzw. nach Ausführung als Ausgabeparameter zurückgibt. Ferner wird festgelegt, wie Ein- und Ausgabeparameter einer Aktivität auf Prozessvariablen abgebildet werden. Ein Beispiel stellt Abbildung 1-2 dar. Darin und im Folgenden wird die Notation des ADEPT-Basismodells [Reic00] verwendet, allerdings werden auch nicht formale Bedingungen an die Kanten alternativer Verzweigungen notiert.

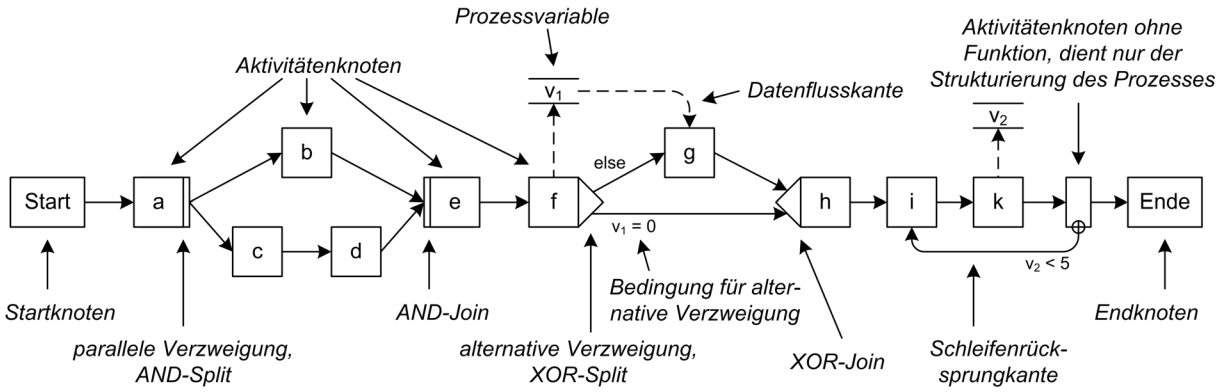

Abbildung 1-2: Beispiel eines Prozesses und seiner Elemente. Dargestellt sind Kontroll- und Datenfluss, die Zuordnung von Aktivitäten zu Anwendungsfunktionen und Bearbeitern wird in der Regel nicht dargestellt.

Es gibt zwei Arten von Aktivitäten: *Automatische Aktivitäten* werden von der Workflow-Engine selbst gestartet und ohne Mitwirkung eines Benutzers ausgeführt, *manuelle Aktivitäten* werden von einem Benutzer ausgeführt, meistens unter Zuhilfenahme eines Anwendungsprogramms. Entsprechend wird für automatische Aktivitäten und manche manuelle Aktivitäten ein Anwendungsprogramm festgelegt, das bei der Abarbeitung der Aktivität ausgeführt wird. Unterschiedliche Anwendungsprogramme müssen auf unterschiedliche Weise angesprochen werden. Mögliche Technologien sind entfernte Prozeduraufrufe, die Kommunikation mittels Nachrichten oder die Verwendung von Web Services, um nur einige zu nennen. Auf die Anbindung unterschiedlicher Anwendungsprogramme wird im Rahmen dieser Arbeit nicht eingegangen, ein Mechanismus zur Anbindung beliebiger Anwendungsprogramme wird in [LeRo00] skizziert. Einen Spezialfall automatischer Aktivitäten stellen Subprozesse dar: Dabei

wird für eine Aktivität eine komplette Prozessvorlage angegeben. Wird die Aktivität ausgeführt, wird eine Prozessinstanz der Prozessvorlage erzeugt und von der Workflow-Engine abgearbeitet. Eine Ausprägung einer Aktivität, die zur Laufzeit ausgeführt wird, wird *Aktivitäteninstanz* genannt. Sie ist Teil einer Prozessinstanz, so wie eine Aktivität Teil einer Prozessvorlage ist.

Die Bearbeiter manueller Aktivitäten stehen vor der Ausführung eines Prozesses oft nicht genau fest, z.B. kann als Bearbeiter für eine Aktivität ein beliebiger Mitarbeiter einer Abteilung vorgesehen sein. Deshalb wird zusätzlich zu den Prozessvorlagen ein *Organisationsmodell* definiert. Dieses beschreibt die organisatorische Struktur eines Unternehmens. Dazu wird die Struktur eines Unternehmens mit allen organisatorischen Einheiten (Abteilungen, Unterabteilungen, Stellen, …) und Rollen (Eine Rolle definiert die Fähigkeiten eines Mitarbeiters, z.B. „Programmierer“, „Administrator“) abgebildet und alle Mitarbeiter werden den Organisationseinheiten und Rollen zugeordnet[1]. Ebenso werden allen manuellen Aktivitäten Kriterien zugeordnet, anhand deren die Bearbeiter ermittelt werden, die diese Aktivität ausführen können. Grundlage hierfür sind die Definitionen im Organisationsmodell, z.B. kann einer Aktivität als Bearbeiter die Abteilung „Support“ und die Rolle „Programmierer“ zugeordnet werden, was bedeutet, dass alle Programmierer der Abteilung „Support“ diese Aktivität ausführen können.

Aufgabe einer Workflow-Client-Anwendung ist, einem Benutzer seine „Aufgaben“ auf einer *Arbeitsliste* anzuzeigen. Dies sind alle Aktivitäteninstanzen aller aktiven Prozessinstanzen, die von der Workflow-Engine als ausführbar markiert sind und für die der Benutzer als Bearbeiter in Frage kommt. Der Benutzer kann auf seiner Arbeitsliste Einträge, d.h. Aktivitäteninstanzen, zur Bearbeitung auswählen. Diese werden dann von den Arbeitslisten aller anderen Benutzer entfernt, und der Benutzer wird fest als Bearbeiter der Aktivitäteninstanz eingetragen. Anschließend kann der Benutzer die Aktivitäteninstanz starten, wodurch das für die Aktivität festgelegte Anwendungsprogramm ausgeführt wird. Je nach Einsatzgebiet kann eine Workflow-Client-Anwendung die Auswahl zu bearbeitender Schritte auch automatisch vornehmen, z.B. indem sie beim Beenden einer Aktivitäteninstanz den ersten Eintrag der Arbeitsliste auswählt und startet.

Schlägt die Ausführung einer Aktivität fehl, muss je nach Aktivität und Prozess unterschiedlich darauf reagiert werden. Je nach Anwendungs-Kontext, kann das Fehlschlagen einer Aktivität einfach ignoriert und wie eine erfolgreiche Beendigung der Aktivität behandelt werden. In anderen Fällen müssen die Aktivität oder mehrere Aktivitäten zurückgesetzt und neu gestartet werden, oder es muss sogar die komplette Prozessinstanz zurückgesetzt werden. Das Verhalten im Fehlerfall wird in der Prozessvorlage festgelegt.

---

[1] Dies ist nur ein Beispiel für ein Organisationsmodell. Unterschiedliche Workflow-Engines unterstützen oft sehr verschiedene Organisationsmodelle.

Der Lebenszyklus einer Aktivitäteninstanz ist durch das Zustandsdiagramm in Abbildung 1-3 festgelegt. Direkt nach der Erzeugung einer Prozessinstanz sind alle ihre Aktivitäteninstanzen im Zustand „nicht aktiviert“, nur die Instanz der Start-Aktivität ist im Zustand „aktiviert“. Wird eine manuelle Aktivitäteninstanz von einem Bearbeiter ausgewählt, wechselt sie in den Zustand „ausgewählt“. Wird sie vom Bearbeiter gestartet, wechselt sie in den Zustand „laufend“. Automatische Aktivitäteninstanzen werden von der Workflow-Engine gestartet und wechseln daher direkt vom Zustand „aktiviert“ in den Zustand „laufend“. Bricht eine Aktivitäteninstanz ab (z.B. aufgrund eines Fehlers), wechselt sie in den Zustand „abgebrochen“. Wird sie erfolgreich beendet, wechselt sie in den Zustand „beendet“. Wird eine Aktivitäteninstanz gar nicht ausgeführt (z.B., weil sie in einem Teilzweig liegt, der nicht ausgeführt wird), wechselt sie vom Zustand „nicht aktiviert“ in den Zustand „nicht ausgeführt“. Abgebrochene, erfolgreich beendete und nicht ausgeführte Aktivitäteninstanzen können zurückgesetzt werden, wonach sie wieder im Zustand „nicht aktiviert“ sind.

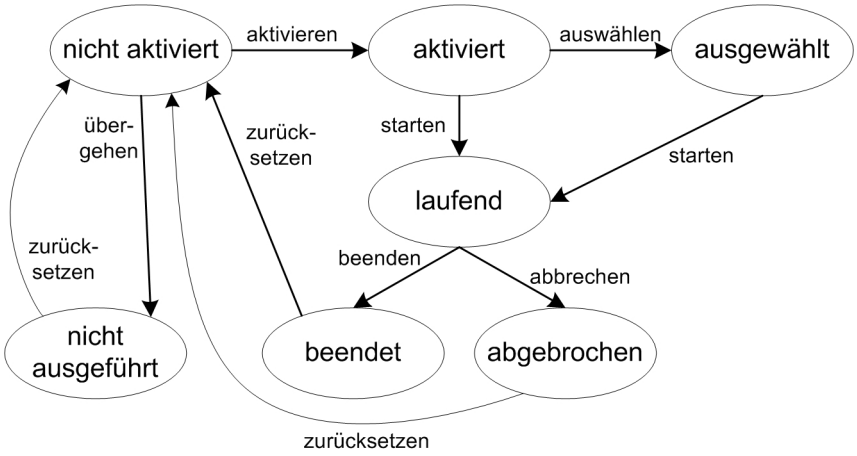

Abbildung 1-3: Zustandsdiagramm einer Aktivitäteninstanz.

1.4 Übersicht über diese Arbeit

Diese Arbeit gliedert sich wie folgt: In Kapitel 2 wird VAS analysiert. Dazu wird ermittelt, welche Abläufe vorkommen. Diese werden anhand von drei Unternehmen, in denen VAS eingesetzt wird, vorgestellt.

In Kapitel 3 wird darauf eingegangen, wie Workflow-Engines in VAS eingesetzt werden können. Hierzu soll untersucht werden, inwiefern dies mit verfügbaren Workflow-Engines erreicht werden kann und inwiefern VAS dafür angepasst werden muss.

In Kapitel 4 werden die hauptsächlich zur Modellierzeit relevanten Aspekte betrachtet. Darin werden unterschiedliche Prozessbeschreibungssprachen analysiert und es wird darauf eingegangen, welche Abhängigkeiten von Anwendungsfunktionen existieren. Weiter wird behandelt, wie die Abhän-

gigkeiten formal beschrieben werden können, um die Arbeit von Prozessmodellierern durch automatische Korrektheitsprüfungen zu erleichtern.

Kapitel 5 untersucht, inwiefern sich Gemeinsamkeiten zwischen den Prozessen in unterschiedlichen Unternehmen identifizieren lassen und ob diese ausgenutzt werden können, um die Wartbarkeit des Systems zu erhöhen. Abschließend werden in Kapitel 6 einige verwandte Arbeiten betrachtet.

2 Analyse von VAS

In diesem Kapitel wird analysiert, welche Teile von VAS für den Einsatz einer Workflow-Engine geeignet sind und für welche Teile dies weniger sinnvoll ist. Im Ergebnis werden die Abläufe in VAS, wie es bei drei Musterunternehmen eingesetzt wird, vorgestellt. Abschließend wird untersucht, inwiefern sich das Anwendungsszenario von VAS von üblichen Anwendungsszenarien für Workflow-Engines unterscheidet.

2.1 Anwendungsfälle für eine Workflow-Engine

In VAS kommen zwei Arten von Vorgängen vor: Zum einen Versand- und Anliefervorgänge, bei denen Lkw-Fahrer und teilweise auch Werksmitarbeiter beteiligt sind, zum anderen alle Vorgänge, die der Verwaltung der Daten im System dienen und von Werksmitarbeitern ausgeführt werden, z.B. die Pflege von Stamm- und Bewegungsdaten und das Erstellen von Auswertungen und Statistiken. Letztere bestehen aus sehr kurzen Prozessen mit nur wenigen Schritten. Ein Beispiel ist die Stammdatenverwaltung: Dazu wird z.B. ein Kundendatensatz aufgerufen, einzelne Details werden abgeändert und anschließend wird der Datensatz gespeichert oder die Bearbeitung abgebrochen. Eventuell ist noch ein Schritt vorgeschaltet, in dem der Benutzer einen Datensatz anhand bestimmter Kriterien suchen kann. Bei jedem Schritt kann der Benutzer den Vorgang abbrechen. Dies ist in Abbildung 2-1 dargestellt.

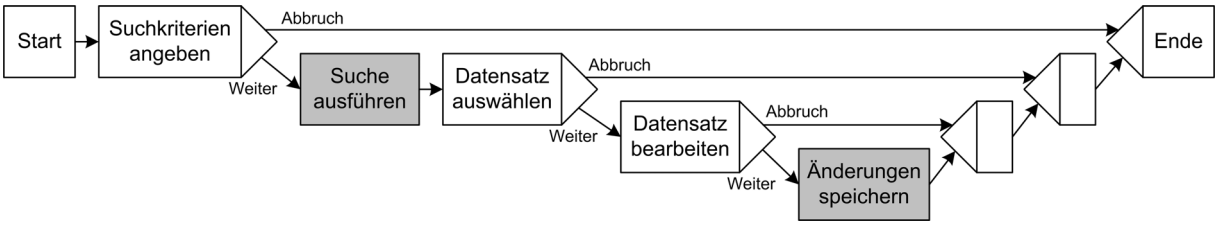

Abbildung 2-1: Ablauf der Änderung eines Stammdatensatzes. Die grau hinterlegten Aktivitäten stellen vollständig automatische Aktivitäten dar, die anderen stellen Bildschirmmasken dar, die vom Benutzer bedient werden. An den Kanten ist jeweils die Benutzerentscheidung notiert, die dazu führt, dass der entsprechende Pfad im Ablauf gewählt wird.

Die Steuerung solcher Vorgänge durch eine Workflow-Engine ist wegen des strengen Ablaufschemas, dem sie folgen, relativ einfach zu realisieren. Zudem sind sie sehr schnell abgeschlossen (innerhalb weniger Minuten) und eine Bearbeiterzuordnung ist nicht notwendig. Andererseits lohnt der Einsatz einer Workflow-Engine zur Steuerung solcher Vorgänge kaum, weil sie aus sehr wenigen Teilschritten bestehen. Da auch Änderungen dieses Ablaufs (die mit einem Workflow-Management-System leichter zu realisieren wären) kaum zu erwarten sind, bietet eine Workflow-Engine für diese Tätigkeiten keine Vorteile. Stattdessen ist mit höherem Aufwand für die Realisierung und schlechterer Ausführungszeit

(eine Workflow-Engine benötigt zusätzliche Zeit) zu rechnen. Diese Vorgänge werden deshalb im Folgenden nicht betrachtet.

Anders verhält es sich bei Versand- und Anlieferungsprozessen: Diese sind nicht nur deutlich umfangreicher (im Sinne der Anzahl von Teilschritten), sondern werden auch häufiger geändert und unterscheiden sich stärker zwischen verschiedenen Unternehmen. Für diese soll nun untersucht werden, inwiefern sie von einer Workflow-Engine gesteuert werden können und welche Konsequenzen das hat.

2.2 Abläufe in unterschiedlichen Unternehmen

Als Grundlage für diese Arbeit wurde VAS in der aktuellen Version analysiert, wie es in drei Unternehmen, jeweils in mehreren Werken, eingesetzt wird. Da die Versandabläufe in den drei Unternehmen sehr unterschiedlich sind, werden hier alle vorgestellt. Es werden nur die Prozesse für Versand und Anlieferungen per Lkw betrachtet, da zum Versand mit anderen Transportmitteln wie Eisenbahn und Schiff nur sehr wenige Informationen zur Verfügung stehen.

Die Abläufe in den Unternehmen lassen sich in zwei Ebenen untergliedern:

1. Der Gesamtablauf, von der Einfahrt bis zur Ausfahrt, beschreibt den „Weg“ eines Lkw im Werk. Die Schritte dieses Ablaufs sind „Einfahrt“, „Ausfahrt“ und die einzelnen Be- und Entladeschritte. Jeder Schritt kann auf einem bestimmten Terminal oder einer Gruppe von Terminals ausgeführt werden. Jeder Prozess ist als Ganzes einem Lkw bzw. Fahrer zugeordnet, dessen Werksbesuch er steuert.

2. Prozesse an einem Terminal. Diese definieren den gesamten Ablauf, der an einem Terminal stattfindet. Sie beginnen in der Regel mit der Identifikation eines Fahrers bzw. Lkw und enden, wenn der Lkw das Terminal verlässt. Sie entsprechen ungefähr den Schritten der Prozesse der ersten Ebene, d.h. sie steuern die Abläufe von Einfahrt, Ausfahrt und der einzelnen Be- und Entladeschritte.

2.2.1 Abläufe in Unternehmen eins

Die Abläufe in diesem Unternehmen sind durch einen relativ geringen Automatisierungsgrad gekennzeichnet. So werden Fahrzeuge durch Werksmitarbeiter ohne Unterstützung von VAS beladen. Zudem werden keine Fahrzeugstammdaten gehalten, d.h., Fahrzeuge können nicht überprüft werden (z.B. auf ungewöhnliches Gewicht). Die Werke des Unternehmens verfügen über zwei Waagen mit Selbstbedienungsterminals, an denen sowohl die Einfahrts- als auch die Ausfahrtsabwicklung durchgeführt werden kann. In der aktuellen Ausbaustufe werden Anlieferungen nicht von VAS behandelt, weshalb hier auch nur Versandprozesse betrachtet werden. Für diese gibt zwei Arten von Aufträgen: Großaufträge, bei denen ein Kunde des Unternehmens innerhalb eines bestimmten Zeitraums eine bestimmte

Menge eines Artikels zu festgelegten Konditionen abholen kann. Bei Einzelaufträgen wird für einen einmaligen Versandvorgang genau festgelegt, von welchem Artikel welche Menge abgeholt werden kann. Ein Einzelauftrag kann aus mehreren Positionen bestehen, d.h. mehreren Artikeln, die einzeln abgeholt werden.

Zudem gibt es zwei Versandarten: Versand loser Ware und Versand verpackter Ware. Beim Versand loser Ware werden Ein- und Ausfahrt von Lkw durch den Lkw-Fahrer an einem Selbstbedienungsterminal abgewickelt. Bei der Einfahrt wird eine Chipkarte ausgegeben, die im Werk zur Identifikation dient, und auf der die Daten der Lieferung gespeichert werden. Diese Daten liest ein Werksmitarbeiter an der Beladestelle aus, um zu erfahren, mit welchen Waren der Lkw beladen werden soll. Bei der Ausfahrt wird die Chipkarte wieder eingezogen. Beim Versand verpackter Ware werden alle Prozessschritte von Werksmitarbeitern abgearbeitet, der Lkw wird nicht gewogen (die zugeladene Menge ist durch die Zahl der Einzelpackungen genau bestimmt) und es werden keine Chipkarten ausgegeben. Der Ablauf beim Versand verpackter Ware ist sehr einfach: Bei der Einfahrt sucht ein Werksmitarbeiter die gewünschte Lieferung, gibt die Daten des Lkw (z.B. das Kfz-Kennzeichen) ein aus und druckt einen Ladeschein. Mit dem Ladeschein fährt der Lkw weiter zur Beladestelle, wo er beladen wird. Bei der Ausfahrt druckt der zuständige Werksmitarbeiter einen Lieferschein aus und gibt ihn dem Lkw-Fahrer mit. Dieser Ablauf ist in Abbildung 2-2 vereinfacht dargestellt. Die Vorgänge, die bei Einfahrt und Ausfahrt des Lkw durchgeführt werden, ähneln bezüglich Komplexität und Struktur den Vorgängen zur Stammdatenbearbeitung. Wie diese auch, bestehen sie aus wenigen Schritten, in denen Datensätze ausgewählt und bearbeitet werden und sie können nach jedem Schritt abgebrochen werden (in der Abbildung nicht dargestellt), wonach sie wiederholt werden müssen.

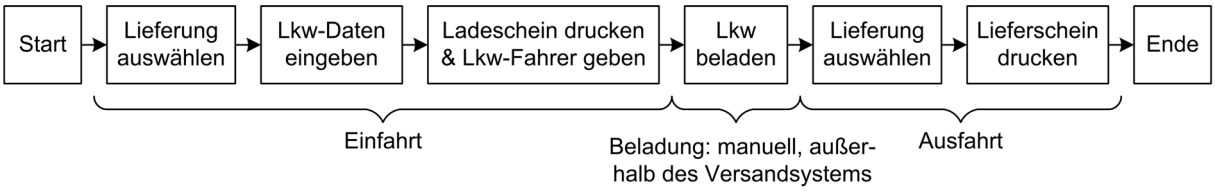

Abbildung 2-2: Ablauf Versand verpackter Ware, Unternehmen eins (vereinfacht).

Ein- und Ausfahrtsabwicklung beim Versand loser Ware sind stärker prozessorientiert implementiert. So können z.B. bei der Einfahrtsabwicklung beim Versand loser Ware auch die Daten einer ausgewählten Lieferung verändert werden (z.B. eine andere Menge für die zu versendende Ware angegeben werden), oder es kann eine neue Lieferung im System angelegt werden anstatt eine existierende auszuwählen, was aber eigentlich nicht vorgesehen ist. Dies liegt daran, dass einem Werksmitarbeiter mehr Freiheit bei der Bedienung von VAS eingeräumt werden kann als einem Lkw-Fahrer. Der Ablauf am Einfahrtsterminal für den Versand loser Ware ist in Abbildung 2-3 dargestellt.

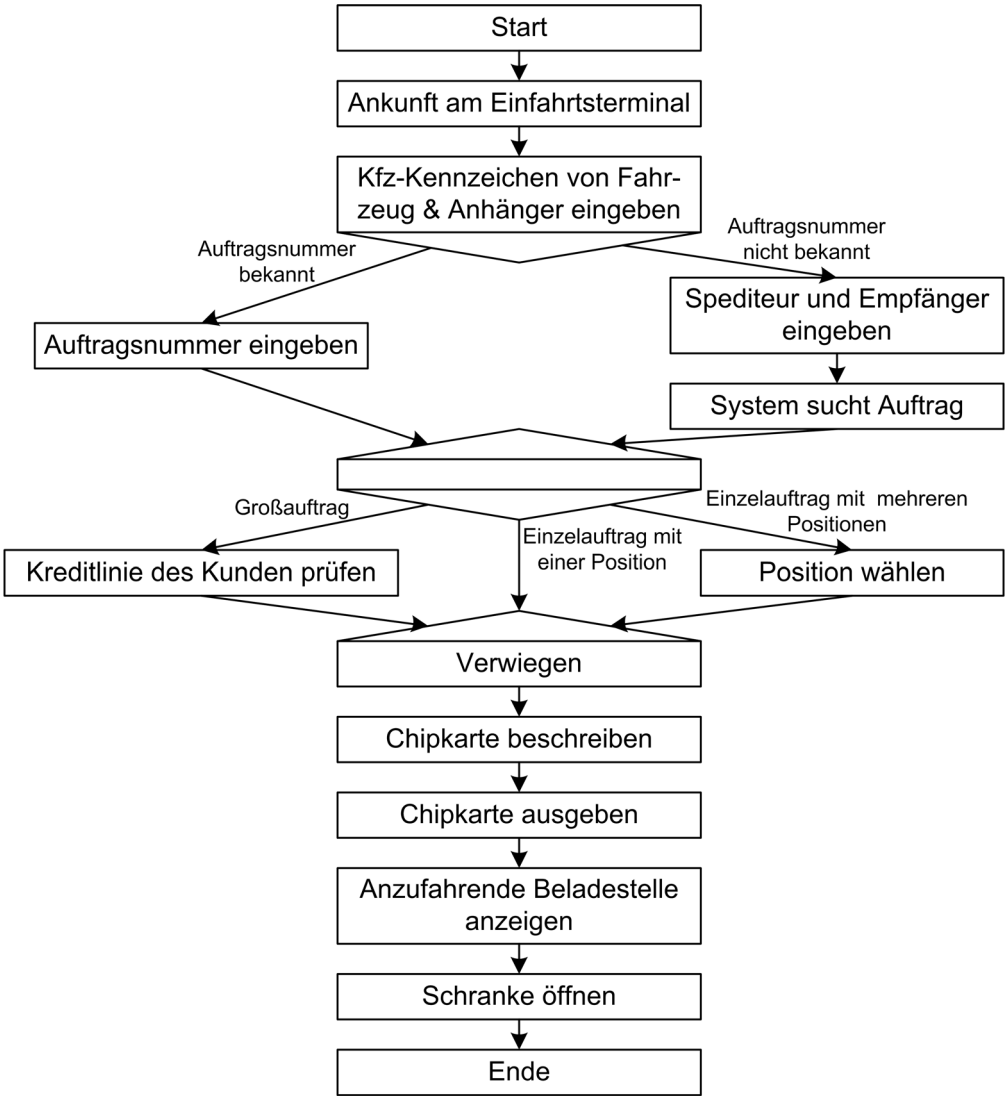

Abbildung 2-3: Einfahrtsprozess beim Versand loser Ware in Unternehmen eins.

Ein Lkw-Fahrer, der im Werk ankommt, fährt auf die Waage an einem Einfahrtsterminal. An diesem gibt er die Kfz-Kennzeichen seines Fahrzeugs und des Anhängers ein sowie entweder eine Auftragsnummer oder, falls er die nicht kennt, Spediteur und Empfänger, anhand deren das System den entsprechenden Auftrag sucht. Handelt es sich um einen Großauftrag, wird die Kreditlinie des Kunden per Abfrage an das ERP-System geprüft. Bei Einzelaufträgen mit mehreren Positionen muss der Lkw-Fahrer eine auswählen. Anschließend wird der Lkw gewogen, und alle Daten der Lieferung (Material, Menge, Kfz-Kennzeichen des Lkw, Ladestelle) werden auf einer Chipkarte gespeichert. Diese Chipkarte nimmt der Lkw-Fahrer mit ins Werk. Nach dem Ausgeben der Chipkarte wird dem Lkw-Fahrer angezeigt, zu welcher Beladestelle er fahren muss und die Schranke wird geöffnet. An der Beladestelle liest ein Werksmitarbeiter die Chipkarte des Lkw-Fahrers aus, belädt den Lkw und speichert auf der Chipkarte, dass die Lieferung freigegeben ist. Wieder an der Ein-/Ausfahrtsstelle, wird der in Abbildung 2-4 dargestellte Prozess ausgeführt.

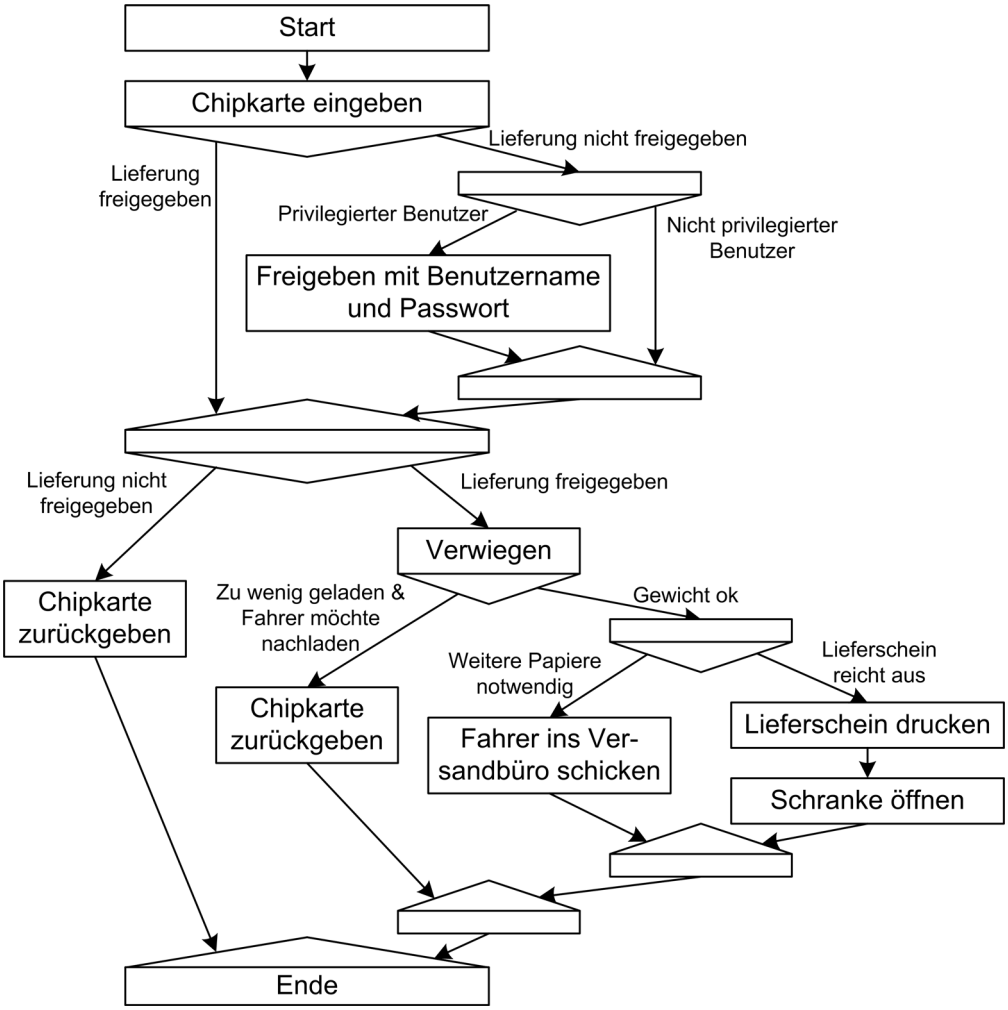

Abbildung 2-4: Ablauf am Ausfahrtsterminal beim Versand loser Ware, Unternehmen eins.

An der Ausfahrt angekommen, gibt der Fahrer seine Chipkarte ein. Ist seine Lieferung nicht freigegeben, kann er sie selbst durch Eingabe von Benutzername und Passwort freigeben, falls er die entsprechenden Rechte besitzt, andernfalls muss er zurück zur Beladestelle. Ist die Lieferung freigegeben, wird der Lkw verwogen, und aus dem Vergleich mit dem Einfahrtsgewicht die Zulademenge errechnet. Ist diese niedriger als geordert und der Fahrer möchte nachladen lassen, erhält er seine Chipkarte zurück und kann noch mal zur Beladestelle fahren. Andernfalls wird ein Lieferschein gedruckt, die Schranke geöffnet, und der Lkw verlässt das Werk. Eventuell zusätzlich notwendige Papiere, wie z.B. Dokumente für den Versand ins Ausland, erhält der Fahrer im Versandbüro.

2.2.2 Abläufe in Unternehmen zwei

Die Werke in Unternehmen zwei haben unterschiedliche Abläufe, da in einem der Werke die Beladung automatisiert ist, in den anderen Werken aber nicht. Bisher wird nur der Versand loser Ware von VAS unterstützt, Anlieferungen und Vorgänge mit verpackten Waren sind erst in der nächsten Aus-

baustufe geplant. Deshalb gibt es noch keine ausreichenden Informationen über diese Vorgänge, die im Folgenden nicht weiter betrachtet werden sollen. Ein interessanter Aspekt ist, dass die drei Teile des Prozesses, Einfahrt, Ausfahrt und Beladung, jeder für sich entweder durch den Lkw-Fahrer allein oder durch einen Werksmitarbeiter durchgeführt werden können (in den Werken ohne automatisierte Beladung kann die Beladung nur durch Werksmitarbeiter durchgeführt werden). Dabei sind beliebige Kombinationen von durch den Lkw-Fahrer allein und durch einen Werksmitarbeiter durchgeführten Prozessteilen möglich. Eine Besonderheit des Unternehmens ist, dass nicht Fahrzeuge identifiziert werden, sondern Fahrer. Die Daten von Fahrzeugen werden zwar ebenfalls verwaltet, im Gegensatz zu den beiden anderen Unternehmen, bei denen Fahrer vom System im Prinzip ignoriert werden, ist der Fahrer aber wichtiger als das Fahrzeug. In manchen Werken des Unternehmens werden für Ein- und Ausfahrt die gleichen Terminals verwendet, in anderen Werken stehen getrennte Terminals für die Einfahrts- und Ausfahrtsabwicklung zur Verfügung.

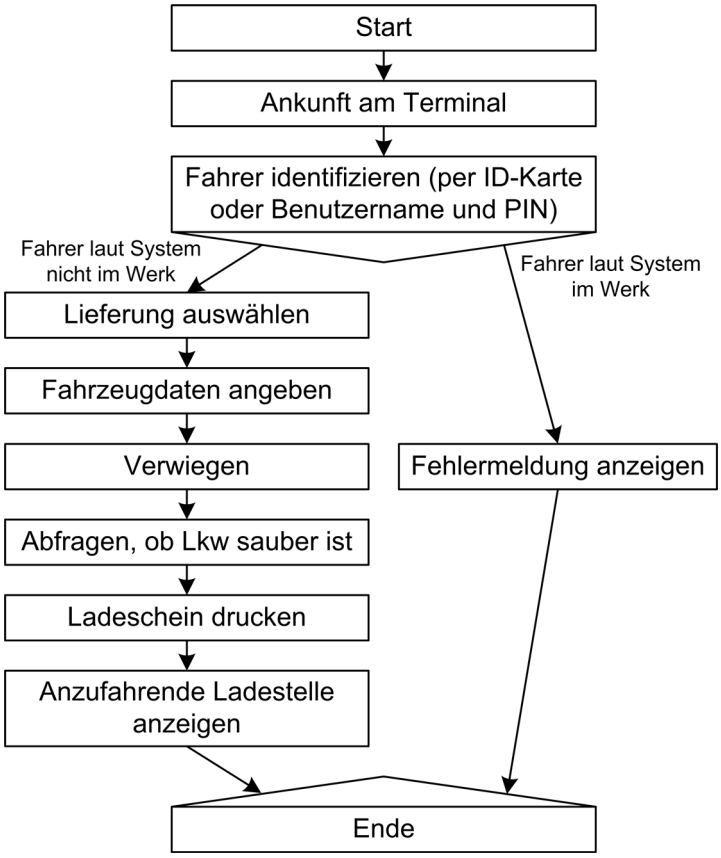

Abbildung 2-5: Vereinfachter Ablauf an einem (reinen) Einfahrtsterminal, Unternehmen zwei.

Abbildung 2-5 zeigt den Ablauf an einem reinen Einfahrtsterminal. Nach der Identifikation per ID-Karte oder Benutzername und PIN wird überprüft, ob der Fahrer laut den Daten von VAS bereits im Werk ist. In diesem Fall wird ein Fehler gemeldet und der Prozess abgebrochen. Andernfalls wählt der Fahrer durch Eingabe seiner ihm bekannten Liefernummer seine Lieferung aus. Fahrzeugdaten sind in

der Regel in den Daten der Lieferung oder des Fahrers hinterlegt, können von diesem aber geändert werden bzw. müssen eingegeben werden, falls sie noch nicht im System hinterlegt sind. Der Lkw wird gewogen und der Fahrer erhält nachdem er die Sauberkeit seines Fahrzeugs bestätigt hat, seinen Ladeschein. Abschließend wird dem Fahrer angezeigt, zu welcher Ladestelle er fahren muss.

Wird die Einfahrtsbehandlung eines Fahrzeugs von einem Werksmitarbeiter durchgeführt, ist der Ablauf sehr ähnlich, allerdings hat der Werksmitarbeiter mehr Freiheitsgrade in der Bedienung. Er wählt zunächst eine Lieferung aus, und gibt dann in beliebiger Reihenfolge Fahrzeug- und Fahrerdaten ein. Die letzten vier Schritte sind identisch zur Einfahrtsabwicklung am Selbstbedienungsterminal.

An der Beladestelle gibt es zwei Möglichkeiten: manuelle und automatische Beladung. Bei der manuellen Beladung belädt entweder der Lkw-Fahrer selbst oder ein Werksmitarbeiter den Lkw. Einzige Aufgabe von VAS ist, den Belader daran zu erinnern, eine Probe zu nehmen, falls dies erforderlich ist. Die Automatische Beladung ist etwas komplizierter: Sie beginnt damit, dass sich der Fahrer mit einer ID-Karte identifiziert. VAS führt daraufhin einige Überprüfungen durch (ist der Lkw an der richtigen Ladestraße, ist die Ladestraße frei, …). Sind die Überprüfungen erfolgreich, wird die Schranke geöffnet und der Lkw kann auf die Ladestraße fahren, wobei ihm angezeigt wird, unter welches Silo er fahren soll. Ist eine Überprüfung nicht erfolgreich, wird eine entsprechende Fehlermeldung angezeigt und der Vorgang abgebrochen. Die Beladung wird nicht von VAS, sondern der Anlagensteuerung durchgeführt. Diese erhält die dazu benötigten Daten von VAS. Die Anlagensteuerung meldet, wenn die Beladung beendet und wie viel zugeladen wurde. An dieser Stelle tritt eine Besonderheit auf. Beim Beladen sammelt sich das Material im Tank kegelförmig unter der Einfüllöffnung. Dabei kann es passieren, dass der Kegel bis zur Einfüllöffnung emporwächst, bevor die geforderte Menge eingefüllt wurde. In diesem Fall muss die Beladung abgebrochen werden (auch dies wird VAS mitgeteilt), der Lkw muss eine gewisse Strecke fahren, sodass sich das Material im Tank verteilt. Anschließend kann die Beladung fortgesetzt werden. Dazu wird ein neuer Beladevorgang gestartet.

Abbildung 2-6 stellt den Ablauf am Ausfahrtsterminal vereinfacht dar. Analog zur Einfahrt, muss sich der Fahrer per ID-Karte oder Benutzername und PIN identifizieren. Es wird wieder überprüft, ob der Fahrer laut den Daten von VAS im Werk ist und eine Fehlermeldung angezeigt, falls dies nicht der Fall ist. Andernfalls werden dem Fahrer noch einmal die Daten seiner Lieferung angezeigt. Der Lkw wird gewogen und es wird überprüft, ob er die richtige Menge zugeladen hat. Abschließend werden der Lieferschein und eventuell notwendige Zertifikate ausgedruckt. Dieser Ablauf ist wieder vereinfacht dargestellt. In der Abbildung fehlt, dass Identifikation, Verwiegung und Gewichtsüberprüfung fehlschlagen können, was zu einem Abbruch des Ablaufs führt.

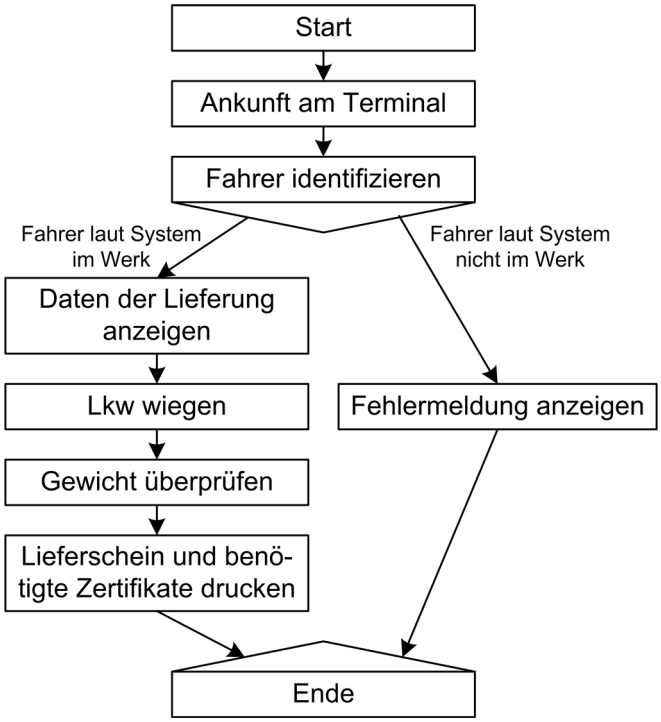

Abbildung 2-6: Ablauf an einem Ausfahrtsterminal, Unternehmen zwei (vereinfacht).

An kombinierten Ein-/Ausfahrtsterminals müssen Einfahrts- und auch Ausfahrtsabwicklung möglich sein. Da VAS nicht sicher entscheiden kann, ob ein Fahrer an einem Ein-/Ausfahrtsterminal ins Werk fahren oder das Werk verlassen möchte, muss der Fahrer dies selbst angeben. Der Ablauf an einem kombinierten Ein-/Ausfahrtsterminal ist in Abbildung 2-7 dargestellt.

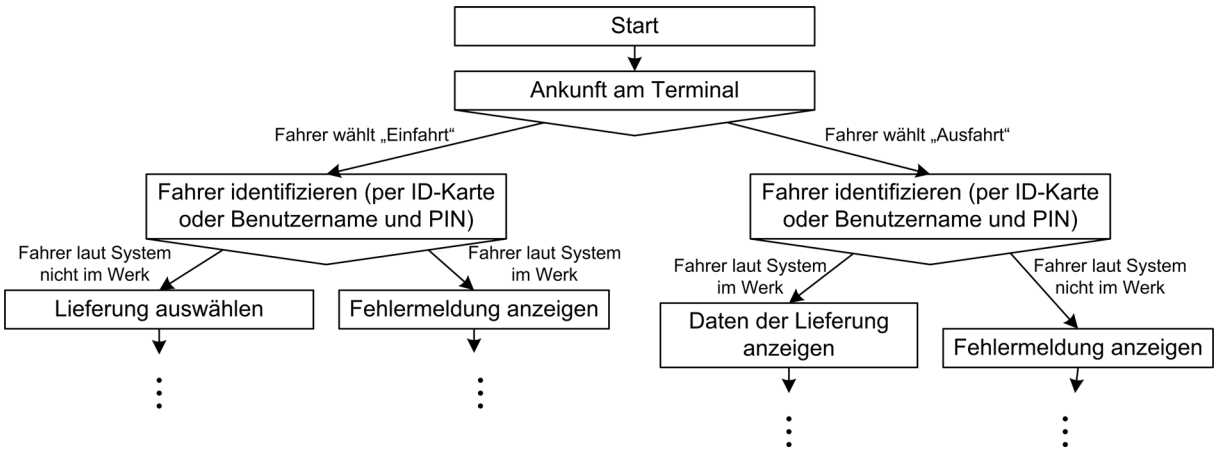

Abbildung 2-7: Ablauf an kombinierten Ein- /Ausfahrtsterminals in Unternehmen zwei. Der Prozess wird wie in Abbildung 2-5 bzw. Abbildung 2-6 dargestellt fortgesetzt.

Eine Spezialität von Unternehmen zwei ist die Behandlung von Rücklieferungen. Kunden wird die Möglichkeit eingeräumt, innerhalb eines gewissen Zeitraums ab Abholung abgeholte Ware wieder

zurückzubringen. Diese wird dann von der Lieferung, die bei der Abholung erstellt wurde, abgebucht. Erst wenn der Zeitraum für Rücklieferungen verstrichen ist oder eine Rücklieferung erfolgt ist, wird die entsprechende Lieferung abgeschlossen und an das ERP-System gemeldet.

2.2.3 Abläufe in Unternehmen drei

Unternehmen drei zeichnet sich durch einen hohen Automatisierungsgrad beim Versand von loser Ware aus. Sowohl Anlieferungen als auch Versandvorgänge sind vollständig automatisiert, d.h. sie werden ohne Unterstützung durch Werksmitarbeiter ausgeführt. Bei Bedarf können Ein- und Ausfahrtsabwicklung auch durch Werksmitarbeiter durchgeführt werden. Die manuelle Beladung ist im Notbetrieb möglich, wird aber außerhalb des Systems durchgeführt. Eine weitere Besonderheit ist, dass statt nur an der Ausfahrt, auch an der Ladestraße Lkw gewogen und Lieferscheine gedruckt werden können. Außerdem ist es möglich, dass ein Lkw gleich mehrere Produkte lädt oder bei einem einzigen Werksbesuch Waren anliefert und neue lädt. In Unternehmen drei werden zur Identifikation der Fahrzeuge Transponder eingesetzt, die über eine gewisse Entfernung ausgelesen werden können. An jedem Lkw, der für Versand oder Anlieferungen in einem Werk des Unternehmens eingesetzt wird, ist ein solcher Transponder angebracht. Kommt ein Lkw an ein Einfahrts-, Ausfahrts- oder Beladeterminal, wird er automatisch identifiziert.

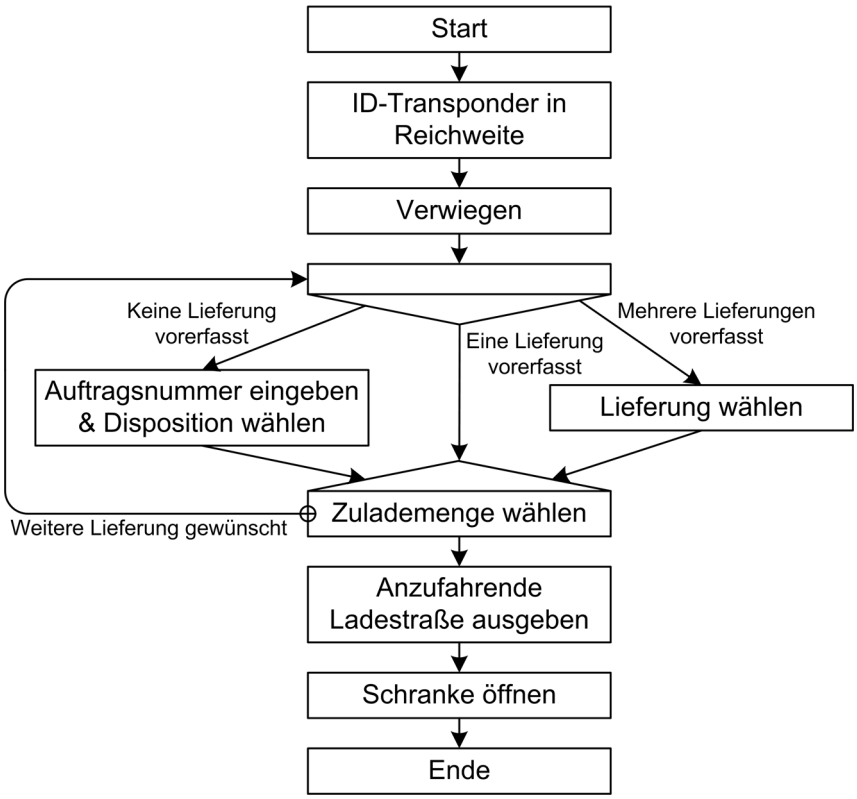

Abbildung 2-8: Ablauf am Einfahrtsterminal bei Versand / Anlieferung loser Ware, Unternehmen drei.

Der Versand verpackter Ware wird durch Werksmitarbeiter durchgeführt. Diese öffnen die Schranken an Ein- und Ausfahrt. Lkw, die verpackte Ware laden, werden nicht gewogen. Stattdessen gibt der Werksmitarbeiter, der den Lkw belädt, im System die Menge der Ware an.

Der Ablauf am Einfahrtsterminal ist in Abbildung 2-8 dargestellt. Der Prozess startet damit, dass ein Lkw auf die Einfahrtswaage fährt, wodurch in Reichweite des Einfahrtsterminals kommt und automatisch identifiziert wird. Nachdem der Fahrer am Terminal eine Verwiegung ausgelöst hat, sucht VAS nach vorerfassten Lieferungen für das Fahrzeug. Existiert eine, wird diese automatisch ausgewählt, existieren mehrere, muss der Fahrer eine daraus wählen. Existiert keine, kann der Fahrer seine Auftragsnummer eingeben. Aus dem so ausgewählten Auftrag kann dann eine Position gewählt werden, aus der eine Lieferung erzeugt wird. Die ausgewählte Lieferung kann sowohl eine Anlieferung als auch eine Auslieferung sein. In jedem Fall muss der Fahrer die Menge wählen, die er mitnehmen bzw. abliefern will. Diese ist durch das Maximalgewicht des Lkw und die in der Lieferung festgelegte Menge nach oben begrenzt. Der Fahrer kann mehrere Lieferungen wählen, allerdings begrenzt durch die Anzahl der Kammern des Tanks seines Lkw. Abschließend wird angegeben, zu welcher Ladestraße der Fahrer fahren muss, und die Schranke wird geöffnet.

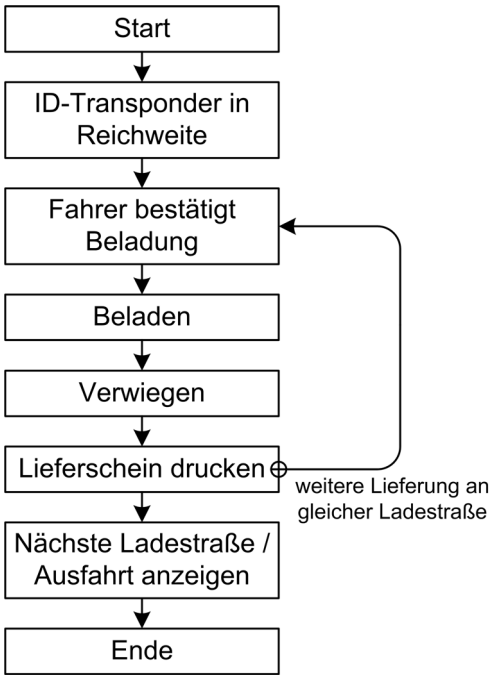

Abbildung 2-9: Ablauf bei der Beladung mit loser Ware, Unternehmen drei.

An der Ladestraße ist der Ablauf wie in Abbildung 2-9 dargestellt. Der Lkw fährt direkt auf die Ladestraße, wodurch er dem Beladeterminal so nahe kommt, dass er automatisch identifiziert wird. Nun muss er die Beladung am Terminal bestätigen, woraufhin VAS die Anlagensteuerung startet, die das Fahrzeug belädt. Anschließend wird durchgeführt, was bei anderen Unternehmen bei der Ausfahrt

geschieht: Der Lkw wird gewogen und ein Lieferschein gedruckt. Dadurch kann vermieden werden, dass der Lkw zwischen zwei Beladevorgängen zum Ausfahrtsterminal fahren muss, und trotzdem das Gewicht einer einzelnen Lieferung registriert werden kann. Das ermittelte Gewicht dient der gerade abgeschlossenen Lieferung als „Ausfahrtsgewicht“ und der nächsten Lieferung, falls eine solche existiert, als „Einfahrtsgewicht“. Wurde an der Einfahrt eine weitere Lieferung ausgewählt, die an der gleichen Ladestraße ausgeführt werden kann, wird der Lkw sofort weiter beladen, andernfalls wird er zur nächsten Ladestraße oder zur Ausfahrt geleitet.

Da „Ausfahrtsverwieung“ und Lieferscheindruck bereits an den Ladestraßen durchgeführt wurden, ist der Ablauf an der Ausfahrt sehr einfach. Kommt ein Lkw zur Ausfahrt, wird er automatisch identifiziert. Es besteht die Möglichkeit, einen Lieferschein nachdrucken zu lassen, anschließend wird die Schranke geöffnet und der Lkw verlässt das Werk.

2.2.4 Gesamtablauf in den drei Unternehmen

Der Prozess erster Ebene (d.h. der Gesamtprozess) in den drei Unternehmen ist auf den ersten Blick identisch und besteht aus den drei Schritten Einfahrt, Be- und Entladen sowie Ausfahrt. Abgesehen davon, dass sich die drei Schritte in ihren Details unterscheiden (z.B. ist die Einfahrtsabwicklung in allen drei Werken unterschiedlich), unterscheiden sich auch die Prozesse erster Ebene selbst zwischen unterschiedlichen Unternehmen.

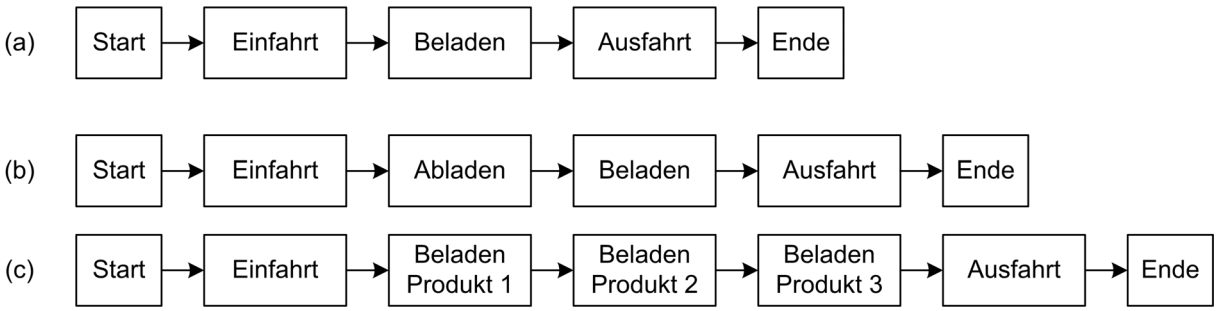

Abbildung 2-10: Prozess erster Ebene der Musterunternehmen. In Unternehmen eins und zwei kommt nur der Prozess (a) vor, in Unternehmen drei variieren die Prozesse erster Ebene, Beispiele sind die abgebildeten.

In den Unternehmen eins und zwei kommt nur der in Abbildung 2-10 (a) dargestellte Prozess vor. Dagegen kann bei Unternehmen drei kein allgemeines Prozessschema festgelegt werden, da jede Prozessinstanz ihr eigenes Schema hat. Bei der Einfahrt, wenn der Lkw-Fahrer auswählt, welche Produkte er anliefert und welche er lädt, wird der Gesamtprozess definiert. Es besteht sogar die Möglichkeit, dass der Gesamtprozess alleine aus Ein- und Ausfahrt besteht.

2.3 Besonderheiten in VAS

Die Abläufe in VAS unterscheiden sich in einigen Punkten von den Abläufen, die üblicherweise von Workflow-Engines gesteuert werden. Eine Besonderheit der Versandprozesse in Unternehmen drei ist, dass ihr genauer Ablauf erst nach dem Start des Prozesses festgelegt wird, indem bei der Einfahrtsabwicklung eines Lkw ausgewählt wird, ob Waren angeliefert werden und welche Waren der Lkw mitnehmen will. In üblichen Workflow-Szenarien sind Prozesse bereits vor ihrem Start fest definiert, und es wird höchstens in Ausnahmesituationen von der Prozessvorlage abgewichen. Die Versandprozesse in Unternehmen drei werden jedoch erst nach dem Start vollständig definiert. In gewisser Weise werden die Prozesse in allen Unternehmen erst während der Einfahrtsabwicklung vollständig festgelegt, da zuvor noch nicht klar ist, an welcher Ladestraße ein Lkw beladen wird. Dies ist aber, wie wir noch sehen werden, ein essentieller Bestandteil der Prozessdefinitionen.

Bei Versandprozessen, zumindest bei voll automatisierten, fällt weiter auf, dass alle Schritte von ein und demselben Bearbeiter (dem Lkw-Fahrer) durchgeführt werden. Anders als in einer Büroumgebung, bei der jeder Mitarbeiter einen festen Arbeitsplatz hat bzw. ohne allzu große Umstände einen beliebigen Arbeitsplatz nutzen kann, ist bei VAS sehr wichtig, dass jeder Prozessschritt an der richtigen Stelle im Werk, d.h. auch am richtigen Terminal durchgeführt wird. So kann ein bestimmtes Produkt nur aus einem Silo auf einen Lkw geladen werden, das das Produkt auch enthält. Für Einfahrt und Ausfahrt werden in der Regel Terminals benötigt, die mit einer geeichten Waage und einem Drucker für Lieferscheine oder Ladepapiere ausgerüstet sind. Aus Sicht der Workflow-Engine nehmen daher Terminals (als Repräsentanten von verschiedenen „Orten“ im Werk, wie Belade- und Abladestellen, Ein- und Ausfahrtsstellen) die Rolle ein, die sonst Bearbeiter haben. Es ist sogar in gewisser Hinsicht ein Organisationsmodell für die Terminals notwendig: Beispielweise kann die Ausfahrtsbehandlung eines Lkw an einem beliebigen von mehreren Ausfahrts-Terminals durchgeführt werden, was einer Zuordnung der Rolle „Ausfahrtsterminal“ zu einer Gruppe von Terminals entspricht. Trotz der Zuordnung von Terminals anstatt von Bearbeitern zu Prozessschritten, muss dennoch jedem Prozessschritt ein Bearbeiter zugeordnet werden, nämlich der Lkw bzw. Fahrer des Lkw, der Waren anliefert oder mitnimmt. Selbst wenn Teile des Prozesses nicht vom Lkw-Fahrer, sondern von Werksmitarbeitern durchgeführt werden, ist dennoch weniger relevant, welcher Werksmitarbeiter eine Aktivität ausführt, als die Stelle im Werk, an der die Aktivität ausgeführt wird und der betreffende Lkw. Dadurch kann sichergestellt werden, dass nur technisch mögliche Aktionen durchgeführt werden (z.B. Beladen nur an der Stelle im Werk, an der die benötigten Waren auch vorhanden sind) und dass immer der richtige Lkw behandelt wird (beim Beladen ist es wichtig, den richtigen Lkw zu beladen, ebenso wie bei der Ausfahrt wichtig ist, dass ein Lkw „seinen“ Lieferschein erhält). Soll darüber hinaus sichergestellt werden, dass nur autorisierte Werksmitarbeiter bestimmte Tätigkeiten durchführen, muss für eine Ak-

tivität zusätzlich zu einer Terminalzuordnung und der Zuordnung zu einem Lkw auch eine Mitarbeiterzuordnung existieren, was z.B. durch eine Rechteverwaltung realisiert werden kann.

3 Einsatz von Workflow-Engines in VAS

In diesem Kapitel wird betrachtet, wie Workflow-Engines in VAS eingesetzt werden können. Abschnitt 3.1 geht zunächst auf die Steuerung der Prozesse zweiter Ebene ein, bevor in Abschnitt 3.2 eine Lösung vorgestellt wird, in der sowohl Prozesse erster als auch Prozesse zweiter Ebene von einer Workflow-Engine gesteuert werden. Abschnitt 3.3 analysiert Schwächen und Stärken der beiden Ansätze und Abschnitt geht abschließend darauf ein, was an VAS selbst geändert werden muss, um den Einsatz von Workflow-Engines zu ermöglichen. In Abschnitt 3.4 wird kurz auf Punkte der Anpassung von VAS an die zuvor vorgestellten Lösungen eingegangen.

3.1 Steuerung der Prozesse zweiter Ebene durch Workflow-Engines

Die Prozesse erster Ebene sind teils sehr einfach (z.B. Unternehmen eins, wo die Prozesse erster Ebene aus den drei Schritten Einfahrt, Beladung und Ausfahrt bestehen), teils von Instanz zu Instanz sehr unterschiedlich (z.B. Unternehmen drei, mit im Extremfall mehreren Be- und Entladeschritten in einem Prozess). Deshalb soll zunächst nur betrachtet werden, wie die Prozesse zweiter Ebene von einer Workflow-Engine gesteuert werden können. Die Prozesse erster Ebene werden nicht explizit gesteuert, stattdessen wird ihr Zustand in der Datenbank gespeichert und während der Abwicklung der Prozesse zweiter Ebene aktualisiert. Dazu benötigen die Terminals folgende Funktionalität:

- An jedem Terminal wird überprüft, ob der Lkw-Fahrer, der es gerade bedient, laut den Daten von VAS im Werk ist. Wird an einem Einfahrtsterminal erkannt, dass der Lkw-Fahrer im Werk ist oder an einem Belade- oder Ausfahrtsterminal, dass der Lkw-Fahrer nicht im Werk ist, muss eine Fehlermeldung erzeugt werden und der Prozess kann nicht fortgesetzt werden. Damit die Überprüfung funktioniert, muss bei der Einfahrtsbehandlung der Status eines Lkw auf „im Werk“ gesetzt werden, umgekehrt muss bei der Ausfahrtsbehandlung der Status des Lkw auf „nicht im Werk“ gesetzt werden. Dies dient z.B. dazu, Lkw-Fahrern, die ohne Einfahrtsabwicklung ins Werk gekommen sind (in manchen Werken ist dies möglich), eine passende Fehlermeldung anzeigen zu können.

- Da sich Lkw frei im Werk bewegen können, kann es vorkommen, dass ein Lkw-Fahrer ein falsches Terminal verwendet. Es kann z.B. sein, dass er das Terminal an einer Ladestraße verwendet, an der die von ihm benötigten Waren nicht verfügbar sind. Deshalb muss an Beladeterminals und an Ausfahrtsterminals überprüft werden, ob ein Lkw an diesem Terminal zu diesem Zeitpunkt bedient werden kann. An Beladeterminals wird dazu überprüft, ob eine offene Lieferung für den Lkw besteht, die an dieser Ladestraße abgewickelt werden kann. An Ausfahrtsterminals wird überprüft, ob noch offene Lieferungen für den Lkw bestehen, die die-

ser erst abwickeln muss, bevor er das Werk verlassen darf. Dementsprechend muss bei der Beladung eines Lkw die entsprechende Lieferung als abgewickelt markiert werden.

- Am Ende jedes Prozesses zweiter Ebene sollte der Lkw-Fahrer darüber informiert werden, zu welchem Terminal er als nächstes fahren soll. Dies kann ebenfalls auf Basis der offenen Lieferungen für den Lkw ermittelt werden.

Da Prozesse zweiter Ebene an einem Terminal von einem Bearbeiter ausgeführt werden, wird für die Aktivitäten der Prozesse keine Bearbeiterzuordnung benötigt. Abbildung 3-1 stellt den Ein- und Ausfahrtsprozess am Beispiel von Unternehmen zwei dar. Die grau hinterlegten Prozessschritte dienen der Aktualisierung oder Abfrage des Lkw-Zustands, d.h. der Verwaltung des Prozesses erster Ebene für den Lkw.

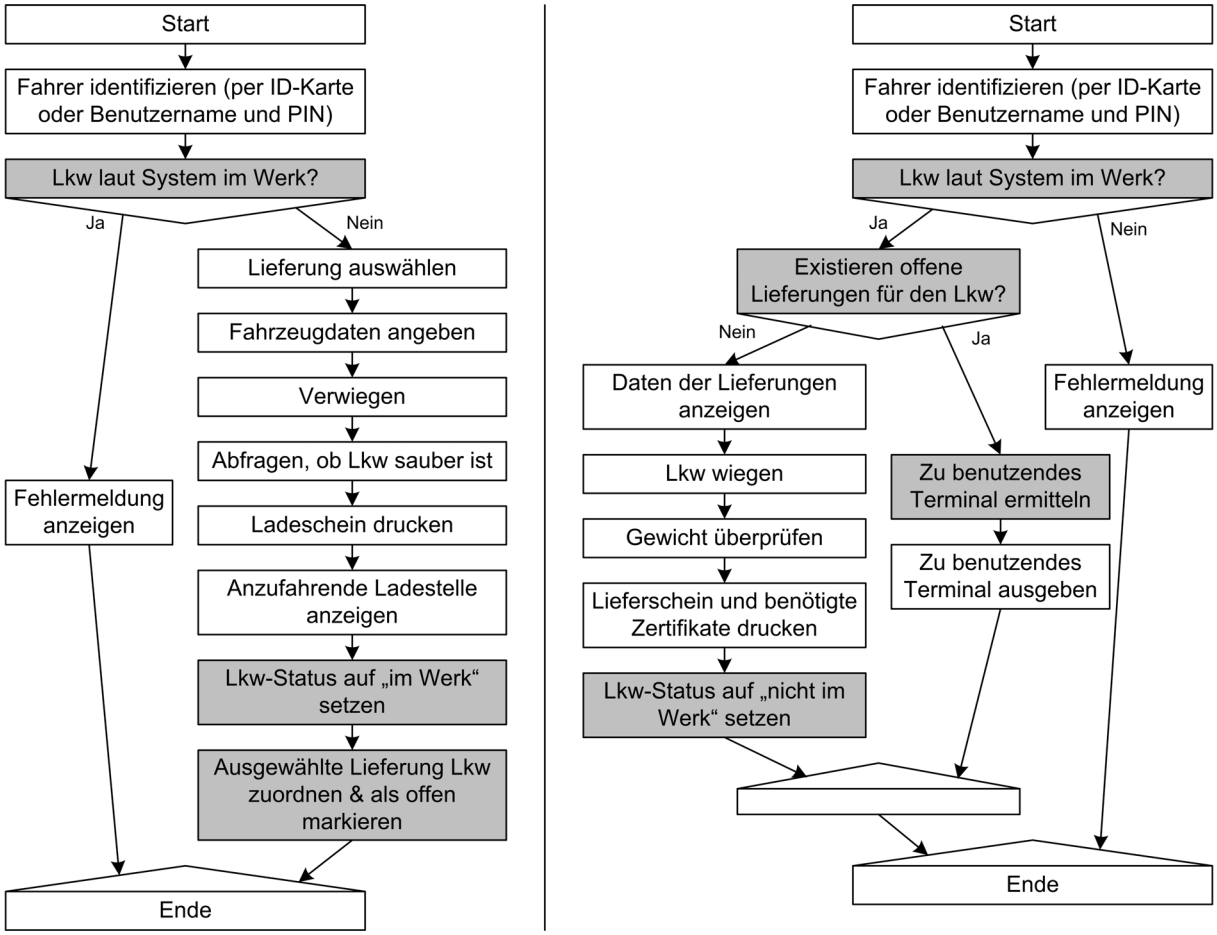

Abbildung 3-1: Ablauf im Einfahrts- (links) und Ausfahrtsterminal (rechts) am Beispiel von Unternehmen zwei. Schritte, die der Überprüfung oder Aktualisierung des Lkw-Zustands dienen, sind grau hinterlegt.

Terminals müssen darauf reagieren können, dass während der Bearbeitung eines Prozesses zweiter Ebene der Lkw-Fahrer das Terminal verlassen kann. Das Terminal muss dies registrieren und den

Prozess abbrechen. Relativ einfach gestaltet sich dies, wenn sich der Fahrer durch eine Karte identifiziert: Während die Karte noch vorhanden ist, kann davon ausgegangen werden, dass der Lkw-Fahrer anwesend ist, und es kann sich auch kein anderer anmelden, da dazu zuerst die andere Karte entfernt werden müsste. Eventuell kann die Karte wie bei einem Geldautomaten bei Prozessstart eingezogen und erst beim Ende des Prozesses wieder ausgegeben werden. Ist dies nicht möglich, muss aber eine Lösung für den Fall vorhanden sein, dass ein Lkw-Fahrer seine Karte vor Beendigung des Prozesses entfernt. Dann muss der Prozess abgebrochen und alle seine Auswirkungen müssen zurückgenommen werden. Dabei ist wichtig, dass der Lkw-Zustand auch nach Abbruch eines Prozesses richtig gespeichert ist, z.B. darf ein Lkw nicht als „im Werk“ gespeichert sein, wenn die Einfahrtsabwicklung des Lkw abgebrochen wurde.

Wird ein Benutzer durch Eingabe von Benutzername und Passwort identifiziert, kann nicht festgestellt werden, wann er ein Terminal verlässt. Deshalb muss eine Zeitbeschränkung bestehen, sodass nach einer gewissen Zeit ohne Benutzerinteraktion davon ausgegangen wird, dass der Lkw-Fahrer nicht mehr anwesend ist, woraufhin seine Prozessinstanz abgebrochen wird, um einem anderen Lkw-Fahrer die Benutzung des Terminals zu ermöglichen. Außerdem muss dafür gesorgt werden, dass ein Benutzer automatisch abgemeldet wird, wenn er sich an einem anderen Terminal anmeldet.

Ähnlich schwierig verhält es sich bei der Identifikation mittels entfernt auslesbarer Transponder. Dabei kann zwar wie bei der Verwendung von Karten erkannt werden, wann ein Lkw-Fahrer ein Terminal verlässt, aber es muss damit gerechnet werden, dass ein anderer Transponder zu nahe an eine Antenne kommt und damit den aktuell angemeldeten Lkw-Fahrer ersetzt. Dann muss sofort der aktuelle Prozess abgebrochen und mit der Behandlung des neuen Lkw-Fahrers begonnen werden.

Damit ein Terminal nicht durch einen Nutzer blockiert werden kann, der z.B. seine Identifikations-Karte vergisst, muss er nach einer bestimmten Zeit ohne Interaktion in jedem Fall automatisch abgemeldet werden.

Wird ein Prozess abgebrochen, muss dafür gesorgt werden, dass das System nicht in einem inkonsistenten Zustand bleibt und dass kein Geschäftsvorfall falsch abgewickelt wird. Beispielsweise könnte passieren, dass ein Lkw ohne Lieferschein das Werk verlässt oder ein Lieferschein mit falschen Daten gedruckt wird. Das lässt sich in vielen Fällen relativ einfach realisieren, indem die Daten, die während des Prozesses verändert werden, erst ganz am Ende in die Datenbank geschrieben werden, sodass bei einem Prozessabbruch keine veränderten Daten gespeichert werden. Um garantieren zu können, dass der Abbruch eines Prozesses zu keinem inkonsistenten Systemzustand führt, wird ein Transaktionskonzept benötigt, das auch Kompensationsaktivitäten einschließt. Allerdings unterstützen nur wenige Workflow-Engines und Prozessbeschreibungssprachen ausreichend mächtige Transaktionskonzepte.

Die Software eines Terminals besteht aus der Workflow-Engine, welche die lokalen Prozesse steuert, den Anwendungsfunktionen, die aus den Prozessen heraus verwendet werden und einer Pro-

zessvorlage. Darüber hinaus wird eine Komponente benötigt, die beim Start eines Terminals eine Prozessinstanz zur lokal vorhandenen Prozessvorlage erzeugt und startet, deren Ende erkennt und dann eine neue Prozessinstanz erzeugt und startet. Außerdem muss erkannt werden, wenn ein Benutzer das Terminal vorzeitig verlässt. Dazu müssen Kartenleser oder die Lesegeräte für Transponder überwacht werden, außerdem muss erkannt werden, wenn ein Benutzer eine Zeit lang keine Aktion mehr ausführt. In diesen Fällen muss der lokale Prozess abgebrochen werden. Die Architektur der Software eines Terminals ist in Abbildung 3-2 einschließlich der Komponenten der Umgebung schematisch dargestellt. Die Komponente „Prozess-Verwaltung“ dient dem Starten, Erkennen des Endes und Abbrechen von Prozessen. Die Komponente „Überwachung“ dient dazu, das vorzeitige Verlassen des Terminals durch einen Benutzer zu erkennen.

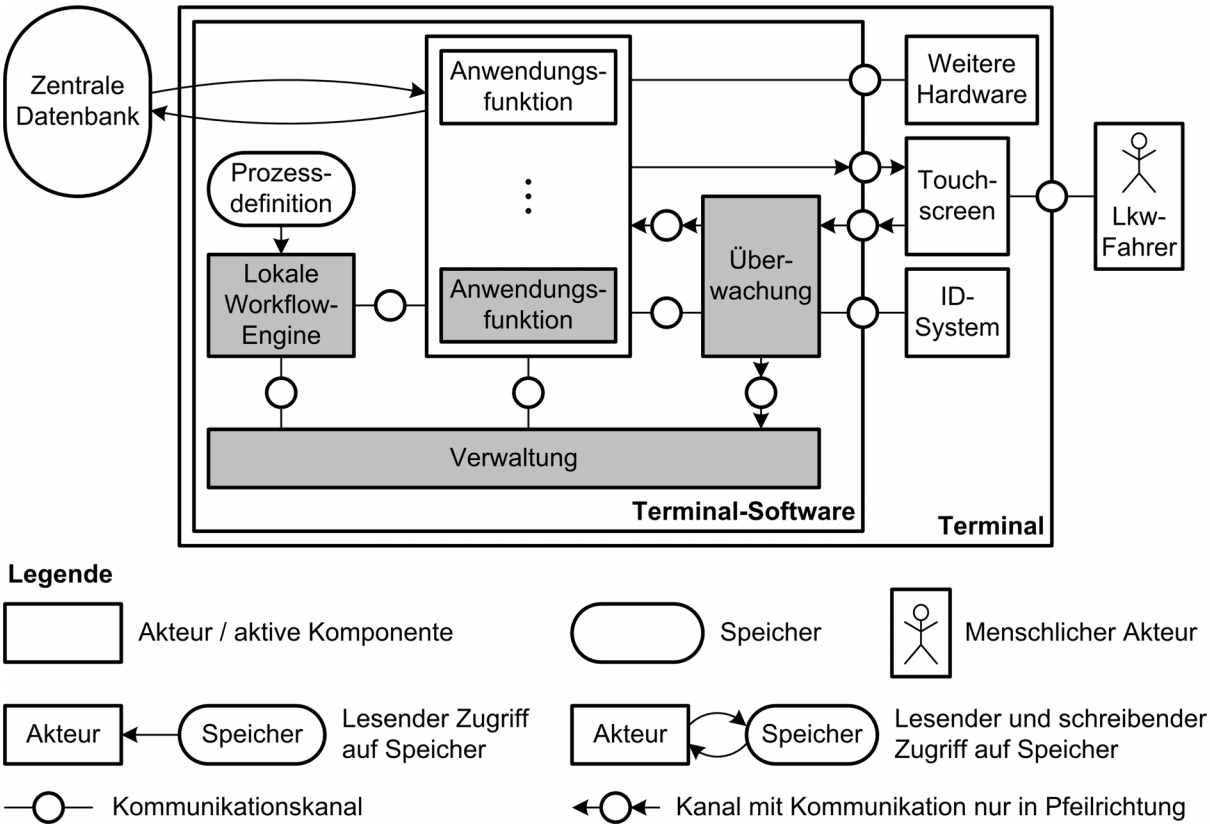

Abbildung 3-2: Architektur der Terminal-Software. Software-Komponenten, die in allen Terminals aller Unternehmen gleich sind, sind grau hinterlegt. Manche Anwendungsfunktionen sind in allen Unternehmen gleich, andere unterscheiden sich.[2]

Zu erkennen ist, dass die Verwaltungssoftware in allen Terminals aller Unternehmen identisch ist. Damit sind die Unterschiede zwischen Unternehmen auf die Anwendungsfunktionen und die Prozess-

[2]Zur Notation wurde FMC (Fundamental Modeling Concepts) verwendet, siehe [FMC].

definitionen beschränkt. Wie in Abschnitt 5.1.3 beschrieben wird, können auch viele Anwendungsfunktionen ohne Anpassung in allen Unternehmen eingesetzt werden, lediglich z.B. GUI-Komponenten unterscheiden sich zwischen verschiedenen Unternehmen.

Um die Wartbarkeit von VAS zu verbessern, bietet es sich an, die Prozessvorlagen der Prozesse zweiter Ebene an einer zentralen Stelle zu verwalten. Dadurch müssen Änderungen an Prozessvorlagen nur an der zentralen Stelle durchgeführt werden, anstatt die Prozessvorlagen auf allen betroffenen Terminals anpassen zu müssen. Der Zugriff auf die Prozessdefinitionen kann so vorgenommen werden, dass die Verwaltungskomponente der Terminals vor dem Start eines Prozesses zunächst überprüft, ob eine neue Version der benötigten Prozessdefinition existiert und diese bei Bedarf lädt. Dazu müssen die Terminals allerdings wissen, welche Prozessdefinition sie laden sollen, damit z.B. Einfahrtsterminals nicht die Prozessdefinition für Ausfahrtsterminals erhalten. Diese Information muss in den Konfigurationsdaten eines Terminals oder der zentralen Prozessverwaltung hinterlegt werden. Wird eine Prozessdefinition an zentraler Stelle verändert, laufen zunächst alle Terminals unverändert mit der alten Version des Prozesses weiter. Sobald auf einem Terminal, das die veränderte Prozessdefinition verwendet, ein Prozess beendet ist, wird das Terminal die neue Prozessdefinition laden und mit dieser weiterarbeiten. So können unkompliziert und relativ schnell die Prozesse für alle Terminals angepasst werden.

Wenn auch die Prozesse erster Ebene von einer Workflow-Engine gesteuert werden, d.h., wenn eine zentrale Workflow-Engine vorhanden ist (siehe Abschnitt 3.2), kann eventuell deren Prozessvorlagenverwaltung als zentraler Speicher für die Vorlagen der Prozesse zweiter Ebene verwendet werden. Allerdings ist dies nur dann möglich, wenn die Prozesse erster und zweiter Ebene in der gleichen Prozessbeschreibungssprache formuliert sind, da eine Workflow-Engine in der Regel nur Prozessvorlagen in der Sprache verwalten kann, die sie selbst verwendet.

3.2 Steuerung der Prozesse erster und zweiter Ebene durch eine Workflow-Engine

In diesem Abschnitt soll gezeigt werden, wie eine Architektur von VAS aussehen könnte, bei der sowohl die Prozesse erster Ebene, d.h. die Gesamt-Prozesse für alle Lkw von der Einfahrts- bis zur Ausfahrtsabwicklung, als auch die Prozesse zweiter Ebene von einer Workflow-Engine gesteuert werden.

Zu den lokalen Workflow-Engines der Terminals kommt eine zentrale Workflow-Engine hinzu, die die Prozesse erster Ebene steuert. Jeder Prozess erster Ebene ist einem Lkw bzw. Lkw-Fahrer zugeordnet. Jede Aktivität daraus (z. B. Einfahrt, Ausfahrt, Beladen mit Produkt x) ist einem Terminal bzw. einer Gruppe von Terminals (z.B. allen Ausfahrtsterminals) zugeordnet, an dem bzw. an denen sie ausgeführt werden kann. Aus Sicht der zentralen Workflow-Engine ist ein Terminal ein Bearbeiter. Die Zuordnung von Aktivitäten zu Terminals kann deshalb im Organisationsmodell der Workflow-

Engine abgebildet werden. Um Aktivitäten einer Gruppe von Terminals zuordnen zu können, werden Rollen für Terminals definiert, z.B. die Rolle „Einfahrtsterminal“ und die Rolle „Ausfahrtsterminal“. Jedem Terminal werden eine oder mehrere Rollen zugeordnet. Einer Aktivität eines Prozesses erster Ebene kann anstatt eines Terminals auch eine Rolle zugeordnet werden. Damit wird definiert, dass dieser Prozess von einem beliebigen Terminal, dem die Rolle zugeordnet ist, ausgeführt werden kann[3]. Die Erfassung von Terminals im Organisationsmodell kann auch bei der Behandlung des Ausfalls eines Terminals helfen. Dies wird im Abschnitt 3.2.1 erläutert. Da die Prozesse erster Ebene nur dazu dienen, Lkw zu den richtigen Terminals zu leiten, werden ihren Aktivitäten keine Anwendungsfunktionen zugeordnet. Sie werden als manuelle Aktivitäten modelliert. Allerdings müssen die Aktivitäten Daten zurückliefern, anhand deren an alternativen Verzweigungen und bei Schleifen entschieden wird, welcher Weg genommen werden soll (vergleiche Abbildung 3-3).

Die Zuordnung von ganzen Prozessen erster Ebene zu Lkw bzw. deren Fahrern wird nicht von allen Workflow-Engines unterstützt. Manche bieten die Möglichkeit, beim Erzeugen einer Prozessinstanz Metadaten wie eine Beschreibung zu dieser zu speichern. Dies kann dazu genutzt werden, eine Identifikationsnummer des Lkw bei der Prozessinstanz zu speichern. BPEL4WS (siehe Abschnitt 4.1.2) bietet ein Konzept namens correlation set, das es ermöglicht, Prozessinstanzen anhand beliebiger, vordefinierter Daten zu identifizieren. Unterstützt eine Workflow-Engine keinen der beiden Mechanismen, muss außerhalb der Workflow-Engine an einer zentralen Stelle im System die Zuordnung von Lkw-Identifikationsnummern zu Prozessinstanzen verwaltet werden.

Die Prozesse erster Ebene unterscheiden sich in allen Unternehmen von Instanz zu Instanz. Dies liegt vor allem daran, dass die Zuordnung der Schritte für Be- und Entladen zu Terminals davon abhängen, welches Produkt angeliefert oder abgeholt wird. Bei Unternehmen drei kommt noch hinzu, dass mehrere Produkte auf einmal angeliefert oder abgeholt werden können, sodass bei der Einfahrt erst festgelegt wird, wie viele Prozessschritte (einer pro Belade- oder Abladevorgang) benötigt werden.

Abbildung 3-3 zeigt die Vorlagen der Prozesse erster Ebene in den drei Musterunternehmen. Dabei wird auch berücksichtigt, dass bei Unternehmen eins und zwei bei der Ausfahrt das Gewicht des Lkw kontrolliert wird und im Bedarfsfall die Lademenge korrigiert werden kann und dass bei Unternehmen zwei in manchen Fällen ein einzelner Beladevorgang in mehreren Schritten ausgeführt wird. Die Vorlagen können nicht vollständig definiert werden. So fehlt in allen Prozessen die Terminalzuordnung bei Be- und Entladeschritten, da diese erst bei der Einfahrt mit der Auswahl der zu entladenden bzw. zu beladenden Produkte festgelegt wird. In Unternehmen drei können auch die Be- und Ent-

[3] Die Zuordnung von Aktivitäten zu (möglichen) Bearbeitern (hier: Terminals) kann im Allgemeinen deutlich flexibler definiert werden, als hier dargestellt. Dazu sei z.B. auf [LeRo00, Jabl95] verwiesen. Hier wird nicht genauer darauf eingegangen.

ladeschritte nicht angegeben werden (in der Abbildung sind sie durch einen Platzhalter repräsentiert), da bei der Einfahrt ausgewählt wird, ob Waren angeliefert und welche und wie viele Produkte abgeholt werden.

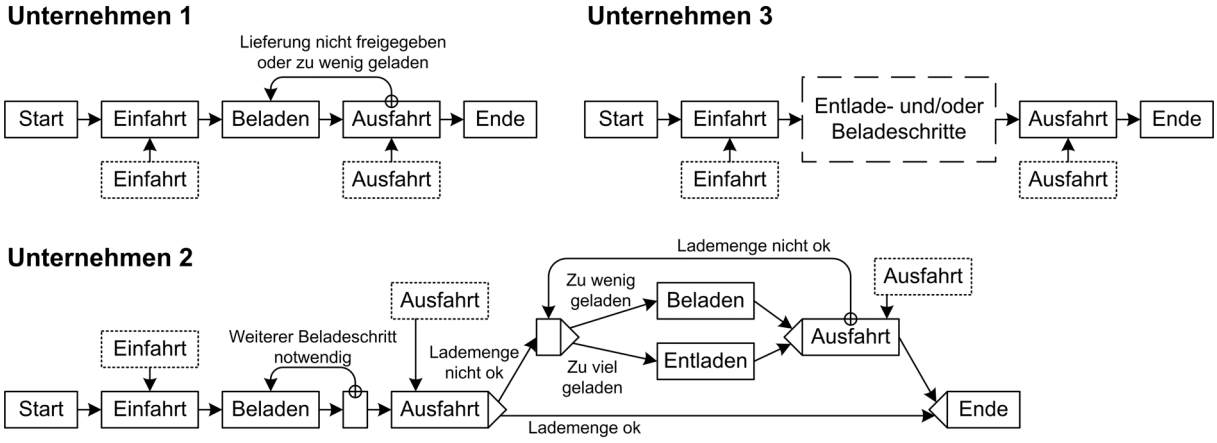

Abbildung 3-3: Vorlagen der Prozesse erster Ebene in den drei Musterunternehmen, einschließlich der zugeordneten Terminals bzw. Rollen.

Die noch fehlenden Angaben der Schemata der Prozessinstanzen erster Ebene werden durch den Lkw-Fahrer festgelegt, der z.B. auswählt, welche Produkte er laden möchte. Allerdings definiert der Lkw-Fahrer nicht das neue Prozessschema, er wählt lediglich aus, welche Produkte er anliefert oder abholt. Aufgabe des Systems ist, daraus das Prozessschema zu vervollständigen. Bei Unternehmen eins und zwei ist dies relativ einfach: Es muss lediglich die Terminalzuordnung für Be- und Entladeschritte abhängig von den ausgewählten Produkten definiert werden. Die Definition der Prozesse erster Ebene in Unternehmen drei ist weniger einfach, da darauf geachtet werden muss, dass Entladeaktivitäten vor Beladeaktivitäten ausgeführt werden. Diese Information muss auf irgendeine Weise bei den Aktivitäten hinterlegt werden. Eine Möglichkeit stellt dar, die Aktivitäten mit Prioritäten zu versehen und nach diesen zu sortieren, sodass alle Aktivitäten mit hoher Priorität vor den Aktivitäten mit niedrigerer Priorität angeordnet werden. Prinzipiell handelt es sich um Abhängigkeiten zwischen den einzelnen Aktivitäten, die mit ähnlichen Beschreibungsmitteln wie den in [Krot03] oder [AAF+02] vorgestellten beschrieben werden können. Prinzipiell kann die automatische Anordnung von Prozessschritten weiter verfeinert werden. Damit könnten zum Beispiel die Wege, die ein Lkw im Werk zurücklegen muss, optimiert werden. Der dadurch erreichte Nutzen stünde aber angesichts der kurzen Wege in einem Werk in keinem Verhältnis zum benötigten Aufwand, zumal sich Lkw-Fahrer durch einen fest vorgegebenen Weg auch in ihrer Handlungsfreiheit eingeschränkt fühlen könnten.

Die Prozesse erster Ebene werden erst bei der Einfahrt eines Lkw vollständig festgelegt. Deshalb ist es nicht sinnvoll, die Aktivität „Einfahrt“ im Prozess erster Ebene einzufügen. Dies bewirkt nur, dass nach der Erzeugung und dem Start einer Prozessinstanz erster Ebene während der Einfahrtsab-

wicklung zusätzlich die erste Aktivität (die Aktivität „Einfahrt“) des Prozesses gestartet und wieder beendet werden muss. Außerdem müsste das Schema der Prozessinstanz geändert werden, während sie bereits läuft, da dieses ja erst während der Einfahrtsabwicklung vollständig festgelegt wird. Wird diese Aktivität stattdessen im Schema des Prozesses erster Ebene weggelassen, reicht es aus, während der Einfahrtsabwicklung einen Prozess erster Ebene anzulegen und zu starten. In Abbildung 3-4 sind für Unternehmen zwei und drei Beispiele von Schemata von Prozessinstanzen dargestellt. Darin wird deutlich, dass in Unternehmen zwei die Terminalzuordnung zu Be- und Entladeschritten und in Unternehmen zusätzlich die Abfolge von Be- und Entladeschritten für jede Prozessinstanz neu definiert werden.

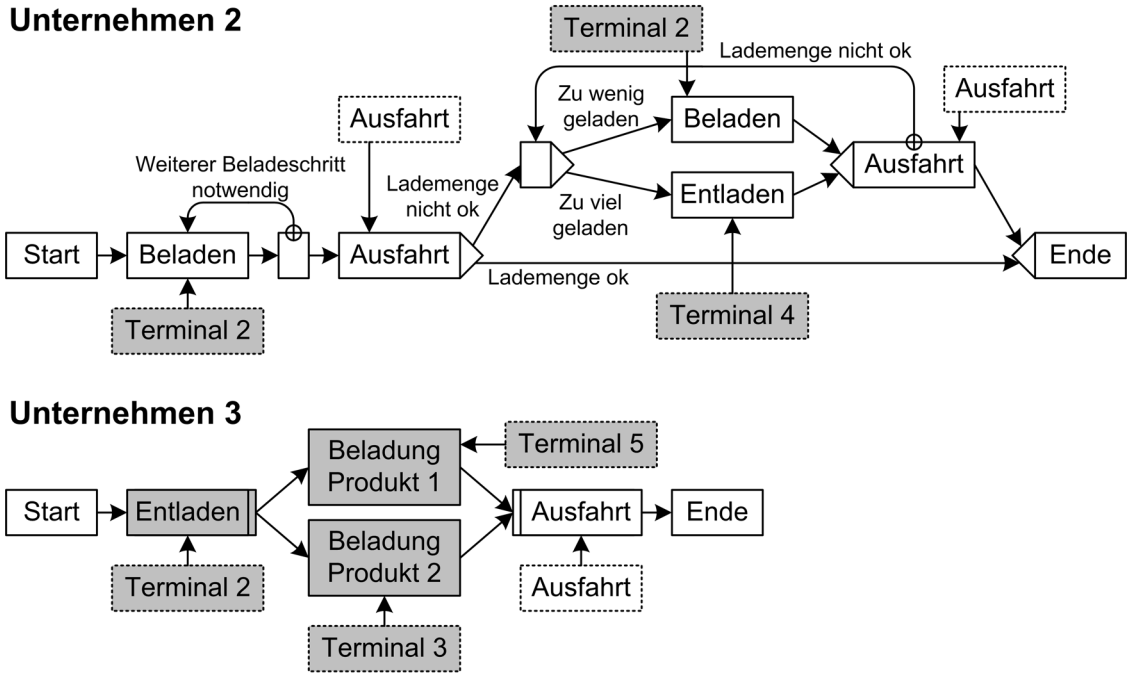

Abbildung 3-4: Beispiele für Schemata von Prozess*instanzen* erster Ebene in Unternehmen zwei und drei. Die im Vergleich zur Prozessvorlage zusätzlich definierten Eigenschaften sind grau hinterlegt.

In Unternehmen eins werden Fahrzeuge ohne Unterstützung durch VAS beladen. Das bedeutet, dass der zentralen Workflow-Engine nicht mitgeteilt werden kann, wann der Beladeschritt gestartet und wann er beendet wird. Darum kann dieser Schritt nicht von der zentralen Workflow-Engine gesteuert werden. Da es auch nicht sinnvoll ist, eine Aktivität für die Einfahrtsbehandlung in den Prozessen erster Ebene vorzusehen, bleibt für die Prozesse erster Ebene in Unternehmen eins nur noch die Aktivität „Ausfahrt“ übrig. Diese von einer Workflow-Engine steuern zu lassen, ist nicht sinnvoll. Darum sollte bei Unternehmen eins die in Abschnitt 3.1 vorgestellte Lösung zum Einsatz kommen.

Werden die Prozesse erster Ebene von einer Workflow-Engine gesteuert, hat dies auch einen geringen Einfluss auf die Prozesse zweiter Ebene. In Abbildung 3-5 sind die veränderten Prozesse zwei-

ter Ebene am Beispiel von Einfahrts- und Ausfahrtsterminal in Unternehmen zwei dargestellt. Alle Aktivitäten, die der Aktualisierung oder Abfrage des Lkw-Zustands dienen, sind durch Aktivitäten ersetzt, die Prozesse erster Ebene aktualisieren oder abfragen. Der Grund dafür ist, dass in den Prozessen erster Ebene die Zustände von Lkw verwaltet werden. Im Prozess an Einfahrtsterminals wird bei der Erzeugung von Instanzen der Prozesse erster Ebene zunächst deren Schema aus der Vorlage und den Angaben des Lkw-Fahrers generiert, bevor eine Instanz dieses neuen Schemas erzeugt und gestartet wird.

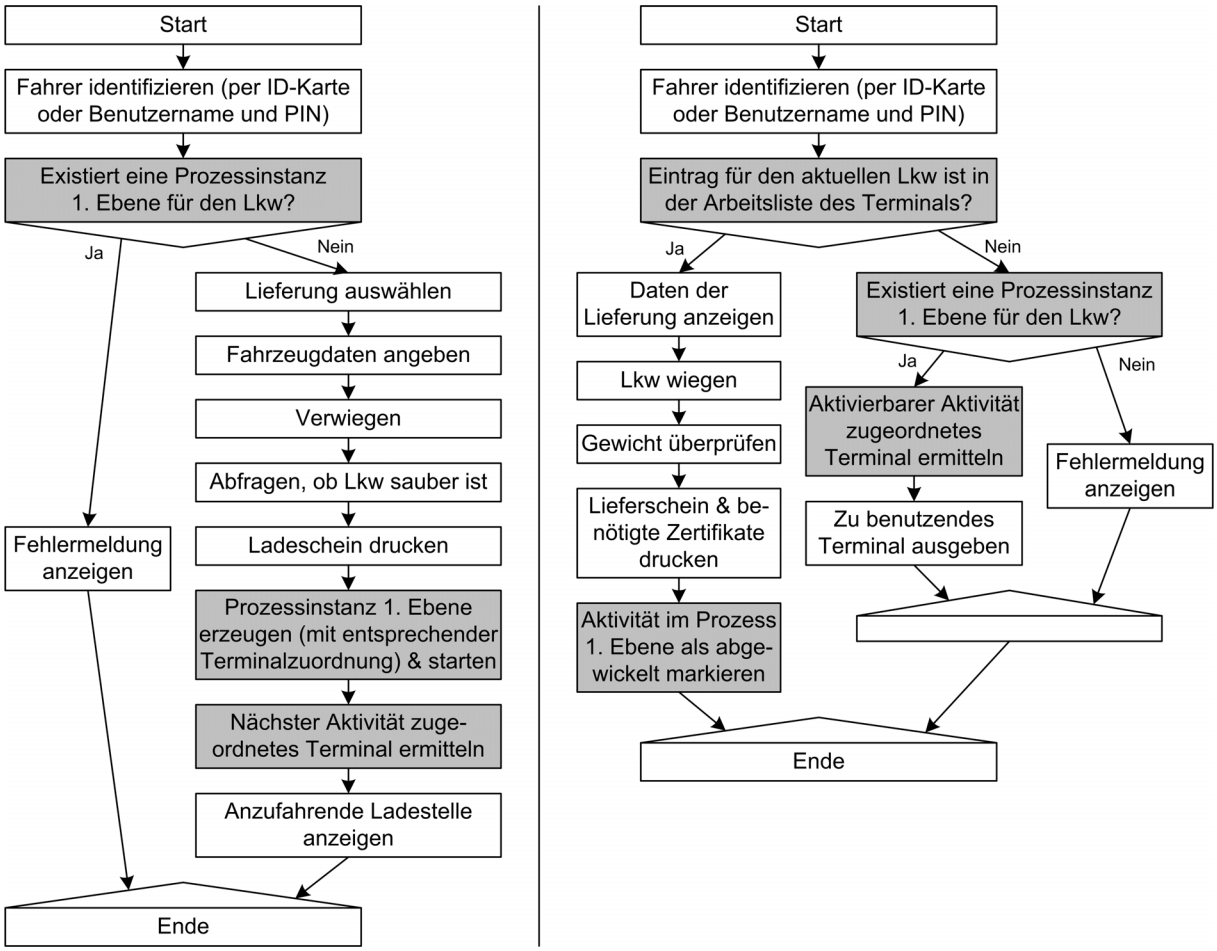

Abbildung 3-5: Prozesse zweiter Ebene (für Einfahrtsterminals, links, und Ausfahrtsterminals, rechts) beim Einsatz einer zentralen Workflow-Engine. Die Aktivitäten, die der Verwaltung der Prozessinstanz erster Ebene dienen, sind grau hinterlegt.

Bei der Abwicklung der Prozesse zweiter Ebene ist es wichtig, dass die entsprechenden Aktivitäten des zugehörigen Prozesses erster Ebene gestartet und am Ende des Prozesses zweiter Ebene beendet werden. Dadurch kann die zentrale Workflow-Engine den Prozess erster Ebene weiterschalten und damit die Arbeitslisten der Terminals aktualisieren, sodass diese auch ermitteln können, welches Terminal ein Lkw-Fahrer als nächstes verwenden muss. Dies wird durch den Einsatz einer zentralen

Workflow-Engine und speziell die Zuordnung von Terminals zu den Aktivitäten der Prozesse erster Ebene stark vereinfacht. Die zentrale Workflow-Engine verwaltet für jedes Terminal eine Arbeitsliste, in der alle Aktivitäten der Prozesse erster Ebene eingetragen sind, die gerade aktivierbar und an diesem Terminal ausführbar sind. Dadurch müssen zur Ermittlung des nächsten zu verwendenden Terminals nicht mehr alle offenen Lieferungen untersucht werden. Stattdessen gibt es zwei Möglichkeiten, das Terminal zu ermitteln, das ein Lkw-Fahrer als nächstes verwenden muss: Es können die Arbeitslisten aller Terminals nach Einträgen durchsucht werden, die zu einer dem Lkw zugeordneten Prozessinstanz gehören. Alternativ kann ein Terminal die dem Lkw zugeordnete Prozessinstanz analysieren und die Terminalzuordnung der aktivierbaren Aktivitäten dieser Prozessinstanz ermitteln. Beide Methoden funktionieren aber erst, wenn der aktuell bearbeitete Prozessschritt erster Ebene als beendet markiert wurde, sodass die zentrale Workflow-Engine den Prozess erster Ebene weiterschalten konnte.

Werden die Prozesse erster Ebene als Zustand in der Datenbank gehalten, wird ihr Zustand von einem abgebrochenen Prozess zweiter Ebene nicht verändert, wenn sichergestellt ist, dass Daten nur persistent gespeichert werden, wenn der Prozess erfolgreich beendet wird. Werden die Prozesse erster Ebenen von einer zentralen Workflow-Engine gesteuert, ist ihr Zustand nicht in der Datenbank gespeichert, weshalb dieser auch nicht automatisch gegen Veränderungen durch nicht erfolgreich abgeschlossene Prozesse zweiter Ebene geschützt ist. Dieser Schutz kann erreicht werden, indem Änderungen an den Prozessen erster Ebene erst am Ende der Prozesse zweiter Ebene ausgeführt werden, sodass ein Abbruch eines Prozesses zweiter Ebene immer vor der Änderung eines Prozesses erster Ebene passieren muss. Dies ist natürlich keine ideale Lösung, besser wäre, wenn die eingesetzte Workflow-Engine ein Transaktionskonzept unterstützt, mit dem Änderungen einer abgebrochenen Prozessinstanz zurückgenommen werden können.

Das Programm eines Terminals muss geringfügig verändert werden, wenn Prozesse erster Ebene zentral von einer Workflow-Engine gesteuert werden. Zum einen muss die Kommunikation mit der zentralen Workflow-Engine ermöglicht werden, zum anderen müssen die Anwendungsfunktionen, die der Verwaltung des Lkw-Zustands dienen, durch solche ersetzt werden, die die Prozesse erster Ebene anlegen, verändern und abfragen.

3.2.1 Reaktion auf den Ausfall von Hardware-Komponenten

Fällt eine Hardware-Komponente für längere Zeit aus (so lange, dass nicht einfach gewartet werden kann, bis die Komponente wieder funktioniert), muss darauf reagiert werden, um den Betrieb in einem Werk weiter garantieren zu können. Unter Umständen müssen auch die verwendeten Prozesse angepasst werden. Dies betrifft allerdings in der Regel keine laufenden Prozessinstanzen. Da die Ausführungszeit der Prozesse relativ kurz ist (in der Regel unter einer Stunde), lohnt es sich oft nicht, diese anzupassen. Im Folgenden sollen verschiedene Szenarien von Hardware-Ausfällen durchgespielt wer-

den. Zum besseren Verständnis der Ausführungen sei an dieser Stelle auf den schematischen Aufbau eines Werks, dargestellt in Abbildung 1-1, hingewiesen.

Hardware-Ausfälle können in verschiedene Kategorien eingeteilt werden.

- Eines von mehreren Ausfahrtsterminals fällt aus. Darauf kann reagiert werden, indem dieses Terminal bei der zentralen Workflow-Engine abgemeldet wird, wodurch diese keine Aktivitäten in die Arbeitsliste dieses Terminals legt und somit kein Lkw-Fahrer zu dem Terminal geschickt wird. Dies kann sogar automatisch geschehen, indem die zentrale Workflow-Engine selbständig ein Terminal abmeldet, mit dem eine gewisse Zeit lang nicht kommuniziert werden kann. Diese Lösung greift nicht nur bei neu gestarteten, sondern auch bei bereits laufenden Prozessinstanzen.

- Eines von mehreren Einfahrtsterminals fällt aus. Da Lkw selbständig, ohne Steuerung oder Unterstützung durch VAS, Einfahrtsterminals anfahren, kann VAS Lkw in diesem Fall nicht umleiten. Dies kann bzw. muss dadurch erreicht werden, dass die Einfahrt, deren Terminal ausgefallen ist, durch Schilder, Absperrband oder ähnliches abgesperrt wird.

- Ein Beladeterminal, ein Silo oder eine ganze Ladestraße fällt aus. Belade-prozesse werden Terminals bzw. Ladestraßen auf Basis des Produkts, das verladen werden soll, und in der Datenbank gespeicherten Zuordnungen von Produkten zu Silos und Silos zu Ladestraßen zugeordnet. Fällt ein Silo oder eine Ladestraße aus, ist diese Zuordnung hinfällig und muss geändert werden. Kann ein Produkt auch aus einem anderen, intakten Silo geladen werden, muss die Zuordnung entsprechend geändert werden, andernfalls wird sie entfernt. Dadurch wird erreicht, dass Lkw-Fahrer nicht mehr zum defekten Terminal bzw. der defekten Ladestraße geschickt werden. Will ein Lkw-Fahrer ein Produkt abholen, das durch den Ausfall nicht verfügbar ist, muss ihm ein Fehler gemeldet werden. In diesem Fall wird aber das Werk in der Regel versuchen, Kunden über die aktuellen Probleme zu informieren, sodass gar kein Lkw ins Werk kommt, der nicht verfügbare Produkte abholen möchte. Lkw, die bereits im Werk sind, müssen anders behandelt werden. Ist ein Beladeterminal defekt, wird es von der zentralen Workflow-Engine abgemeldet. Beim Versuch, eine dem Terminal zugeordnete Aktivität in einem Prozess erster Ebene zu starten, erkennt die zentrale Workflow-Engine, dass kein geeignetes Terminal für die Bearbeitung der Aktivität zur Verfügung steht. Damit kann eine Fehlermeldung an den Fahrer erzeugt werden. Der Ausfall eines einzelnen Silos kann mit der vorgestellten Lösung nicht behandelt werden. Dazu müssten nicht nur die Terminals überwacht werden, sondern auch die Silos, und den Aktivitäten der Prozesse erster Ebene müssten Silos zugeordnet werden anstatt von Terminals.

- Eine Hardware-Komponente fällt aus, sodass ein Terminal nur eingeschränkt einsetzbar ist. Ein Beispiel dafür könnte der Ausfall einer Schranke sein. In so einem Fall ist die Einfahrt, Ausfahrt oder Beladestraße weiter benutzbar, allerdings muss der lokal verwendete Prozess angepasst werden (sodass z.B. die Schranke nicht mehr angesteuert wird). Hat jedes Terminal einen eigenen Speicher für seine Prozessdefinition, kann einfach die Prozessdefinition des betroffenen Terminals angepasst werden. Werden aber die Prozessdefinition für alle Terminals an einer zentralen Stelle verwaltet, gelten Änderungen für alle Terminals gleichen Typs. Deshalb muss eine Lösung gefunden werden, die es einzelnen Terminals erlaubt, von der Standard-Prozessdefinition abzuweichen. Eine Möglichkeit besteht darin, im zentralen Speicher für Prozessdefinitionen bei jeder Prozessdefinition zu hinterlegen, ob sie als Standard-Version dient oder als spezielle Version für ein bestimmtes Terminal. Wenn ein Terminal eine Prozessdefinition aus dem zentralen Speicher lädt, muss es die spezielle Version erhalten, wenn eine solche für es existiert. Existiert keine spezielle Version für das Terminal, darf es die Standard-Version der Prozessdefinition erhalten.

- Ein Terminal fällt aus, die durch das Terminal gesteuerte Hardware funktioniert aber weiterhin. Ein Beispiel für dieses Szenario ist der Ausfall eines Ausfahrtsterminals. In diesem Fall muss die Ausfahrtsabwicklung von Werksmitarbeitern durchgeführt werden, die von ihrem Arbeitsplatz mit Unterstützung durch VAS die Hardware an der Ausfahrt wie Schranke und Waage bedienen. Dieser muss dann auch den entsprechenden Prozessschritt im Prozess erster Ebene als abgeschlossen markieren, sodass der Prozess weitergeschaltet wird.

- Ein Terminal fällt aus und muss durch ein Terminal anderen Typs „vertreten“ werden. Beispiele für dieses Szenario sind der Ausfall eines Ein- oder Ausfahrtsterminals in einem Werk, in dem jeweils nur ein Ein- bzw. Ausfahrtsterminal verfügbar ist. In diesem Fall muss die Funktionalität dieses Terminals durch ein anderes übernommen werden. Beispielsweise können Ein- und Ausfahrtsterminals als Ersatz füreinander eingesetzt werden. Dazu muss dem Ersatzterminal die gleiche Rolle zugewiesen werden wie dem ausgefallenen, sodass die zentrale Workflow-Engine diesem auch die entsprechenden Aktivitäten in die Arbeitsliste legt. Da das Ersatzterminal anderen Typs ist als das ausgefallene, muss eventuell die Prozessdefinition für dieses Terminal angepasst werden. Wird z.B. ein Einfahrtsterminal als Ersatz für ein Ausfahrtsterminal verwendet, muss dessen Prozessdefinition durch die Definition eines Prozesses für kombinierte Ein-/Ausfahrtsterminals ersetzt werden.

Durch den Einsatz von Workflow-Technologie, wie er in den vorherigen Abschnitten dieses Kapitels dargestellt wurde, kann angemessen und relativ schnell auf viele der skizzierten Hardware-Ausfälle reagiert werden: Fällt eines von mehreren Einfahrts- oder Ausfahrtsterminals aus, wird es einfach von der zentralen Workflow-Engine abgemeldet. Dadurch können sogar schon laufende Pro-

zessinstanzen problemlos weitergeführt werden. Muss ein Einfahrtsterminal ein Ausfahrtsterminal ersetzen (oder umgekehrt), wird das defekte Terminal von der zentralen Workflow-Engine abgemeldet und das Ersatzterminal bei der zentralen Workflow-Engine als Vertreter dieses Terminals registriert oder es wird dem vertretenden Terminal eine weitere Rolle zugeordnet. Eventuell muss zusätzlich die Prozessdefinition für das vertretende Terminal angepasst werden. Ist ein Terminal nur eingeschränkt nutzbar, weil unkritische Teile der verwendeten Hardware ausgefallen sind, kann die dann benötigte Prozessanpassung für das Terminal durch die Erzeugung einer neuen, nur für dieses Terminal freigegebenen Version des Prozessschemas gelöst werden.

Dabei besteht eine Einschränkung, was die Änderung von Prozessschemata betrifft. Ein Beispiel: Für das Schema des Einfahrtsprozesses wird für Terminal x eine veränderte Version erstellt, weil bei diesem Terminal ein Defekt besteht. Noch bevor der Defekt behoben wurde, soll das Schema des Einfahrtsprozesses geändert werden, um Änderungen der Unternehmenspolitik zu reflektieren. Diese Änderungen sollen für alle Terminals gelten, gleichzeitig sollen aber die durch den Defekt an Terminal x bedingten Änderungen erhalten bleiben. Eine Lösung könnte sein, das alte, an den Defekt angepasste Prozessschema für Terminal x beizubehalten, bis der Defekt behoben wurde, und dann auch für dieses die neue Prozessversion zu verwenden. In der Regel sollen aber die Prozesse an allen Terminals der neuen Unternehmenspolitik entsprechen, und an Terminal x sollen zusätzlich die durch den Defekt begründeten Änderungen gelten. Gefragt ist also die Kombination der allgemeinen mit den Terminal-spezifischen Änderungen (siehe Abbildung 3-6).

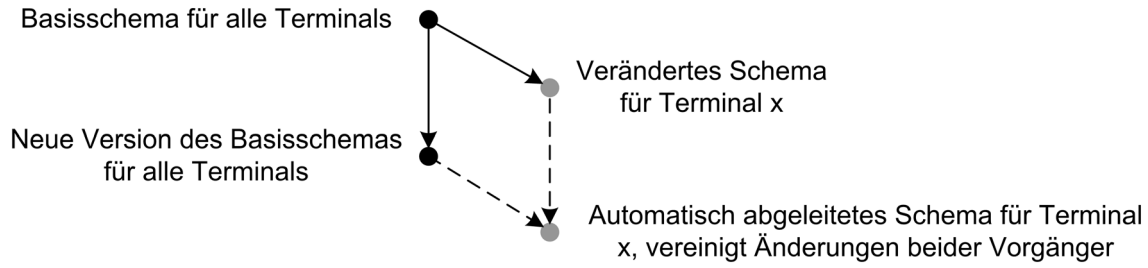

Abbildung 3-6: Ableitung eines Schemas für ein einzelnes Terminal und allgemein für alle Terminals (durchgezogene Pfeile) sowie automatische Kombination der unterschiedlichen Veränderungen zu einer neuen Version des Schemas (gestrichelte Pfeile)

Im ADEPT-Projekt wurden Konzepte entwickelt, um Veränderungen an einem Prozessschema auf Prozessinstanzen zu propagieren, und zwar auch auf Prozessinstanzen, deren Schema bereits verändert wurde [RRD04]. Diese Technologie kann verwendet werden, um die Änderungen des allgemeinen Schemas auf das bereits veränderte Schema für Terminal x automatisch anzuwenden. Dazu werden das alte, unveränderte Prozessschema, das neue Prozessschema und das für Terminal x angepasste alte Prozessschema benötigt. Um deren Verfügbarkeit gewährleisten zu können, bietet sich eine zentrale

Verwaltung der Prozessinstanzen zweiter Ebene in Verbindung mit einer Versionsverwaltung für diese an. Die Versionsverwaltung muss sowohl die einzelnen Versionen als auch die Information verwalten, von welcher Version eine andere abgeleitet wurde.

Sehr kurzfristigen Störungen in Anlagenteilen kann durch Ad-hoc-Änderungen der laufenden Prozessinstanzen der betroffenen Terminals begegnet werden. Damit können nicht ausführbare Aktivitäten im Schema einer oder mehrerer einzelner Prozessinstanzen zweiter Ebene durch andere ersetzt werden und verhindern nicht die Abwicklung der gesamten Prozessinstanz. Solche Änderungen unterstützen Workflow-Engines üblicherweise nicht, eine Ausnahme dazu bildet allerdings ADEPT [ReDa98].

3.3 Bewertung der Ansätze

Dieser Abschnitt legt die Vor- und Nachteile der beiden vorgestellten Ansätze zum Einsatz von Workflow-Engines in VAS dar und zeigt Alternativ-Ansätze in manchen Detailpunkten. Die vorgestellten Ansätze müssen sich an den in Abschnitt 1.2 aufgestellten Zielen (schnellere Anpassung von VAS an neue Werke, Anpassung von Abläufen zur Laufzeit und durch wenig qualifiziertes Personal) messen lassen.

Durch die Auftrennung der Terminal-Programme in explizit modellierte Prozesse und kleine Anwendungsfunktionen sollte sich VAS an neue Werke deutlich beschleunigt anpassen lassen. Sehr viele Anwendungsfunktionen können aus anderen Werken wieder verwendet werden und die Definition von Prozessen mit Hilfe eines (im Idealfall grafischen) Editors ist weniger aufwendig als sie zu programmieren. Dies trifft auf beide vorgestellten Ansätze zu, unabhängig davon, ob die Prozesse erster Ebene von einer Workflow-Engine gesteuert werden oder nur die Lkw-Zustände in einer Datenbank gehalten werden. Ebenso ist es bei beiden Ansätzen möglich, Prozessdefinitionen ohne Neustart von VAS auszutauschen. Die Anforderungen an die Personen, die Abläufe neu definieren oder bestehende verändern, sind durch den Einsatz von Workflow-Engines ebenfalls verringert. Diese müssen keine Programmiersprache beherrschen, sondern nur eine deutlich leichter zu erlernende Prozessbeschreibungssprache. Zudem können Prozesse in den meisten Fällen grafisch modelliert werden. Sind Abhängigkeiten zwischen Anwendungsfunktionen beschrieben und kann das zur Prozessmodellierung eingesetzte Werkzeug diese Beschreibungen analysieren, müssen die Anwender dieses Werkzeugs noch weniger Wissen über VAS haben. In diesen Punkten unterscheiden sich die beiden Ansätze nicht.

Ein Problem des Einsatzes von Workflow-Engines, speziell bei der Steuerung von Prozessen zweiter Ebene, könnte die Zeit sein, die eine Workflow-Engine zum Weiterschalten im Prozess braucht. Diese Zeit ist naturgemäß deutlich größer als die Zeit, die das Weiterschalten durch ein fest

programmiertes Programm bräuchte[4]. Bei den Prozessen zweiter Ebene, die vor allem den Ablauf von Bildschirmmasken steuern, ist es sehr wichtig, dass die Masken ausreichend schnell weitergeschaltet werden, um die Benutzbarkeit der Terminals nicht zu beeinträchtigen. Liegen einige automatische Schritte ohne Benutzerinteraktion zwischen dem Aufruf zweier Bildschirmmasken, muss die Workflow-Engine mehrmals schalten, bis die zweite Bildschirmmaske angezeigt werden kann. Deshalb müssen die Workflow-Engines der Terminals innerhalb von Millisekunden Prozesse weiterschalten können.

Die Hauptvorteile der Steuerung der Prozesse erster Ebene durch eine eigene Workflow-Engine sind die Möglichkeit der dynamischen Reaktion auf gewisse Hardware-Ausfälle sowie die Funktionalität, die nahezu alle Workflow-Engines zur Überwachung von Auswertung von Prozessen anbieten. Da Prozesse erster Ebene Werksbesuchen durch Lkw entsprechen, kann zu jedem Zeitpunkt sehr einfach erkannt werden, wie viele und welche Lkw gerade im Werk sind und an welchen Stellen sie sich befinden. Diese Informationen können auch die Unterstützung durch Werksmitarbeiter für Lkw-Fahrer, die Hilfe benötigen, verbessern helfen, da viele Informationen über die Prozessinstanz sofort verfügbar sind. So kann ein Werksmitarbeiter erkennen, wie weit ein Lkw-Fahrer mit der Abarbeitung seines Prozesses fortgeschritten ist und welches Terminal er gerade verwenden muss. Zusätzlich zur Überwachung der momentan aktiven Prozesse sind auch Informationen über bereits abgeschlossene Prozesse vorhanden, sodass statistische Analysen, z.B. über die Auslastung einzelner Terminals oder die Dauer von Werksbesuchen von Lkw möglich sind.

Diesen Vorteilen gegenüber steht vor allem der höhere Aufwand für die Implementierung eines Versandsystems mit Steuerung der Prozesse erster Ebene durch eine Workflow-Engine. Vor allem die Anpassung der Prozessvorlagen erster Ebene durch die Terminals anhand der Angaben eines Lkw-Fahrers ist relativ aufwendig, insbesondere wenn Abhängigkeiten zwischen einzelnen Prozessschritten beachtet werden müssen (z.B. dass ein Lkw erst entladen werden muss bevor er beladen werden kann). Ähnlich aufwendig ist die Abfrage, welches Terminal ein Lkw-Fahrer als nächstes verwenden muss. Allerdings muss beachtet werden, dass dieser Aufwand letztendlich nur einmal entsteht. Ist die Funktionalität einmal implementiert, muss sie nicht mehr verändert werden. Ein weiterer, wenn auch eher kleiner Nachteil einer zentralen Workflow-Engine ist, dass VAS durch die Einführung dieser Workflow-Engine komplexer und damit tendenziell weniger stabil wird. Die zentrale Workflow-Engine stellt eine weitere Komponente dar, die konfiguriert werden muss und ausfallen kann, zumal der Ausfall der zentralen Workflow-Engine sämtliche Versandvorgänge lahm legt.

Das Beispiel Unternehmen eins zeigt, dass es in manchen Fällen generell nicht sinnvoll ist, Prozesse erster Ebene von einer Workflow-Engine steuern zu lassen. Da in Unternehmen eins nur Ein-

[4] Messungen mit dem ADEPT-Prototypen und der Workflow-Engine Shark [Sha] haben durchschnittliche Zeiten für das Weiterschalten eines Prozesses von circa 350 ms bzw. 30 ms ergeben.

fahrt und Ausfahrt der Lkw automatisiert sind, die Beladung aber vollständig außerhalb von VAS abläuft, lohnt es sich nicht, die Prozesse erster Ebene von einer Workflow-Engine steuern zu lassen. Da in Unternehmen eins auch keine Lkw-Stammdaten verwaltet werden, ist es nicht einmal notwendig, sich zu merken, ob ein Lkw im Werk oder außerhalb ist. Dies wird allein durch die Ausgabe von Chipkarten bei der Einfahrt, deren Rücknahme bei der Ausfahrt und die Daten, die auf der Karte gespeichert sind, verwaltet.

Eine generelle Schwäche beider Ansätze ist, dass die Prozesse zweiter Ebene einige Aktivitäten enthalten (und enthalten müssen), die nicht direkt den Ablauf am Terminal selbst steuern, sondern der Verwaltung der Prozessinstanz erster Ebene dienen. Zudem müssen insbesondere die Aktivitäten zur Aktualisierung der Prozessinstanz erster Ebene an der richtigen Stelle in den Prozessen zweiter Ebene eingefügt werden. So kann z.B. erst ermittelt werden, welches Terminal ein Lkw als nächstes verwenden muss, wenn der zentralen Workflow-Engine das Ende der aktuellen Aktivität gemeldet wurde. Dies erschwert die Modellierung der Prozesse zweiter Ebene, da Prozessmodellierer immer daran denken müssen, auch den Zustand eines Lkw zu verwalten und, falls auch Prozesse erster Ebene von einer Workflow-Engine verwaltet werden, verstehen müssen, wie diese funktioniert.

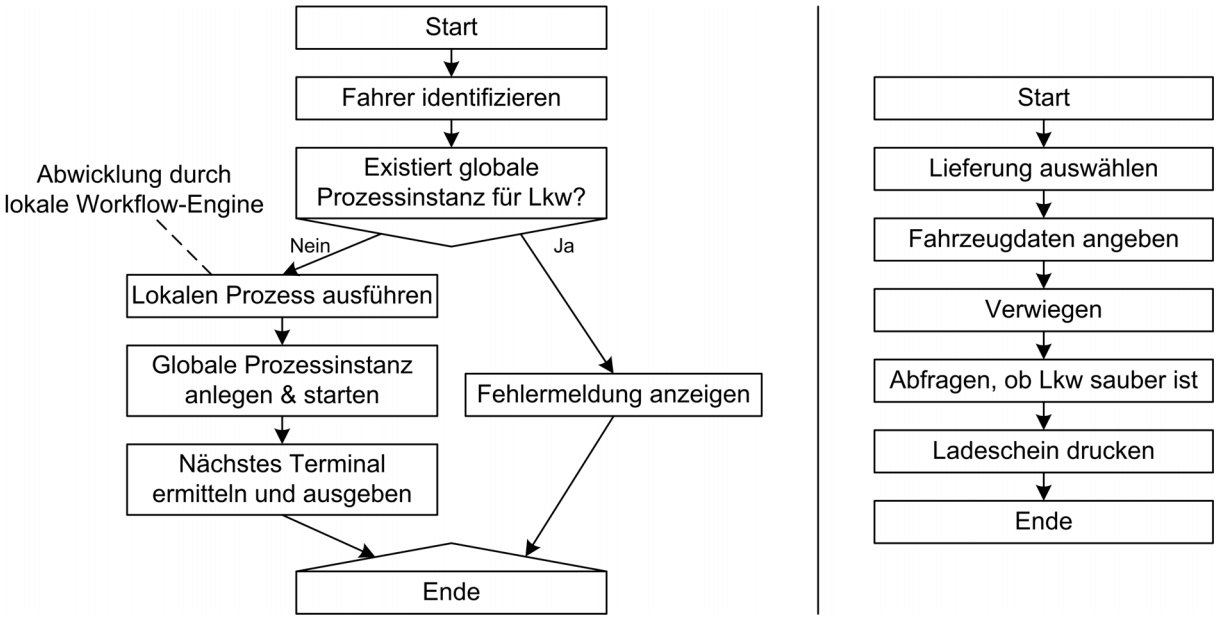

Abbildung 3-7: Ablauf in einem Einfahrtsterminal (links) und von der Workflow-Engine gesteuerter Prozess an diesem Terminal (rechts, hier am Beispiel von Unternehmen zwei), wenn Aktivitäten zur Verwaltung der Prozesse erster Ebene nicht von der Workflow-Engine gesteuert werden.

Eine Alternative dazu ist, die Aktivitäten zur Verwaltung der Prozesse erster Ebene nicht von der Workflow-Engine der Terminals steuern zu lassen, sondern im Terminalprogramm zu verankern. Dadurch müssen sich die Prozessmodellierer nicht mehr um diese Aktivitäten kümmern. Die Terminalprogramme führen dann den in Abbildung 3-7 links dargestellten Ablauf aus. Dieser wurde hier der

besseren Verständlichkeit halber als Prozess modelliert, beschreibt aber den fest programmierten Ablauf in der Terminalsoftware. Der Vorteil, dass sich der Modellierer der Prozesse zweiter Ebene nur um anwendungsspezifische Aspekte kümmern muss (der Prozess für die Einfahrt in Unternehmen zwei sieht wie in Abbildung 3-7 rechts dargestellt aus), wird durch einen anderen Nachteil aufgehoben. Da die Identifikation des Fahrers bzw. Lkw vor allen anderen Schritten ausgeführt werden muss, muss sie zwangsweise Teil des fest programmierten Terminalprogramms sein. Da Lkw aber bei unterschiedlichen Unternehmen auf sehr unterschiedliche Weise identifiziert werden, müssen auch unterschiedliche Terminalprogramme in den unterschiedlichen Unternehmen eingesetzt werden. Damit kann VAS weniger schnell an neue Werke angepasst werden, da so in der Regel nicht nur Anwendungsfunktionen und Prozessdefinitionen, sondern auch fest programmierte Teile der Terminalprogramme angepasst werden müssen. Zudem wird die Flexibilität der Prozessmodellierer eingeschränkt, es können keine Prozesse modelliert werden, bei denen z.B. ein Fahrer zuerst eine Lieferung auswählt und dann erst identifiziert wird.

Aus Sicht der Terminals bzw. der Prozesse zweiter Ebene kann die zentrale Workflow-Engine als eine Komponente betrachtet werden, die wie andere Komponenten auch Anwendungsfunktionen zur Verfügung stellt. Zwischen diesen Anwendungsfunktionen bestehen Abhängigkeiten. So muss z.B. eine Prozessinstanz erst erzeugt werden, bevor eine ihrer Aktivitäten gestartet werden kann. Diese Abhängigkeiten können formal beschrieben und ihre Einhaltung dann theoretisch automatisch überprüft werden. Dies könnte den Prozessmodellierer unterstützen. In der Praxis ist dies allerdings nicht möglich. Es kann z.B. an einem Beladeterminal eine Aktivität gestartet werden, ohne dass zuvor eine Prozessinstanz erzeugt wurde, wenn dies bereits an einem Einfahrtsterminal durchgeführt wurde. Zur Modellierzeit steht allerdings nicht fest, in welcher Reihenfolge Terminals verwendet werden. Damit kann auch nicht überprüft werden, ob die Prozesse zweiter Ebene die Abhängigkeiten zwischen den Funktionen der zentralen Workflow-Engine korrekt berücksichtigen.

3.4 Aspekte der Anpassung von VAS

Zur Umstellung von VAS auf eine Lösung, bei der Workflow-Engines eingesetzt werden, sind folgende Anpassungen notwendig. Zunächst muss alle Funktionalität der Terminalprogramme zu einer Menge von Anwendungsfunktionen abgeändert werden, die von einer Workflow-Engine angesprochen werden können. Dabei ist auch zu entscheiden, wie Daten zwischen einzelnen Anwendungsfunktionen übertragen werden sollen. Möglichkeiten dafür sind die Speicherung in der Datenbank, die Übertragung durch die Workflow-Engine und über andere Mechanismen innerhalb von VAS, z.B. über globale Variablen. Dabei sind Aspekte zu beachten wie die Frage, ob Daten für Entscheidungen an Verzweigungen im Prozessmodell relevant sein können oder ob es sinnvoll ist, Daten bereits vor Ende eines Prozesses in der Datenbank zu speichern, da dann bei einem Abbruch des Prozesses diese eventuell wieder gelöscht werden müssen. Allgemein ist es am sinnvollsten, möglichst alle Daten, die von

einer Anwendungsfunktion an andere weitergegeben werden sollen, durch die Workflow-Engine verwalten zu lassen. Dadurch werden auch externe Datenabhängigkeiten (siehe Abschnitt 4.2) so weit wie möglich vermieden. Allerdings können dadurch die Prozessvorlagen unübersichtlich werden, speziell, wenn es keine Möglichkeit gibt, komplexe Datentypen zu verwalten. Ein Vorteil der strengen Modularisierung der Anwendung in Anwendungsfunktionen liegt darin, dass Entwickler gezwungen werden, das System in relativ kleine, nur schwach gekoppelte Module zu unterteilen, wodurch Qualität und Wartbarkeit des Programmcodes verbessert werden können.

Zusätzlich zu den Anwendungsfunktionen, deren Funktionalität im Prinzip bereits vorhanden ist, müssen die Programmteile zur Verwaltung eines Terminals implementiert werden. Dazu gehören eine Schnittstelle zur lokalen Workflow-Engine und vor allem die Programmteile, die Prozesse starten. Diese müssen auch in der Lage sein, zu erkennen, wenn ein Benutzer das Terminal verlässt. Dazu müssen sie das Identifikationssystem und die Eingabegeräte (in der Regel einen Touchscreen) überwachen. Werden Prozessvorlagen an einer zentralen Stelle im System verwaltet, muss zusätzlich die Funktionalität zur Aktualisierung der lokal vorhandenen Prozessvorlage zur Verfügung gestellt werden.

Werden auch die Prozesse erster Ebene von einer Workflow-Engine gesteuert, müssen die Programme der Terminals darauf angepasst werden, dass sie mit dieser kommunizieren können. Dazu gehören auch Funktionen wie die Ermittlung des zu verwendenden Terminals für einen Lkw-Fahrer oder das Anpassen des Schemas für einen Prozess erster Ebene.

Die Prozesse in VAS können in der Regel mit von nahezu allen Prozessbeschreibungssprachen unterstützten Konstrukten modelliert werden. Die Prozesse beinhalten keine parallelen Verzweigungen, die Konstrukte Sequenz, alternative Verzweigung und Schleife reichen für die Modellierung der Prozesse aus. Allerdings kommen immer wieder Ausnahmen vor. Dies zeigt das folgende Beispiel. Ein Lkw-Fahrer kann sich in Unternehmen zwei am Einfahrtsterminal entweder durch eine ID-Karte oder durch Eingabe von Benutzername und PIN identifizieren. Dazu wird auf dem Bildschirm eine Maske angezeigt, die die Möglichkeit bietet, Benutzername und PIN einzugeben und darauf hinweist, dass alternativ auch die Identifikation per ID-Karte möglich ist. Beim Einschieben der ID-Karte soll die Bildschirmmaske durch eine andere ersetzt werden. Im Prinzip handelt es sich um zwei parallele Aktivitäten („Bildschirmmaske anzeigen“ und „auf ID-Karte warten“), wobei beim Ende einer der Aktivitäten die andere abgebrochen werden soll. Dies kann nicht in allen Prozessbeschreibungssprachen definiert werden, ADEPT und BPEL4WS (siehe dazu auch die Abschnitte 4.1.4 bzw. 4.1.2) bieten Möglichkeiten für die Modellierung dieses Falls, XPDL (siehe Abschnitt 4.1.3) dagegen nicht. Solche speziellen Anforderungen können nicht generell vorhergesehen werden, da sich die Abläufe von Unternehmen zu Unternehmen unterscheiden und daher beim Einsatz von VAS in einem neuen Unternehmen bisher unbekannte Abläufe implementiert werden müssen. Deshalb ist es sinnvoll, eine Workflow-Engine einzusetzen, die eine möglichst ausdrucksmächtige Prozessbeschreibungssprache

verwendet. Kommt es trotzdem vor, dass ein von der verwendeten Workflow-Engine nicht unterstütztes Konstrukt benötigt wird, gibt es zwei Möglichkeiten: Das Konstrukt kann, wie in Abschnitt 4.1 beschrieben, in einer Anwendungsfunktion implementiert werden. Alternativ dazu kann in einigen Fällen auch der Ablauf selbst so abgeändert werden, dass das Konstrukt nicht benötigt wird. Das trifft auch in dem oben erwähnten Beispiel zu. Fügt man der Maske für Eingabe von Benutzername und PIN eine Schaltfläche hinzu, mit der der Fahrer auswählen kann, dass er sich per ID-Karte identifizieren möchte, können die beiden Aktivitäten nacheinander ausgeführt werden. Die Bedienung des Terminals wird dadurch zwar etwas aufwendiger, da bei der Identifikation per ID-Karte ein zusätzlicher Knopfdruck notwendig wird, die grundsätzliche Funktionalität, dass sich Fahrer per ID-Karte und per Benutzername und PIN identifizieren können, bleibt aber erhalten.

4 Aspekte der Definition von Prozessen

In diesem Kapitel wird darauf eingegangen, wie die Modellierung von Prozessen möglichst einfach und sicher (im Sinne der Vermeidung von Fehlern) gemacht und der Gesamtaufwand für die Anpassung von VAS an neue Anforderungen minimiert werden kann.

Wichtigster Aspekt ist, dass Prozesse möglicht einfach modelliert werden können sollen, sodass dies von möglicht wenig qualifiziertem Personal ausgeführt werden kann. Endziel ist, Prozesse grafisch modellieren zu können und möglichst alle Fehlerquellen durch automatische Überprüfungen auszuschließen oder wenigstens auf potentielle Probleme hinzuweisen. Die Idee ist, dass zunächst die Abläufe in einem Werk analysiert und abstrakt modelliert werden, wobei nur die Reihenfolge von Aktivitäten definiert wird, ohne Datenfluss und Zuordnung von Bearbeitern oder Anwendungsfunktionen zu den Aktivitäten. Alternativ dazu können auch die Prozesse aus einem anderen Werk angepasst werden, wenn sie den Abläufen in einem neuen Werk so sehr ähneln, dass dies weniger Aufwand bereitet als die komplette Neu-Definition der Prozesse. Den Aktivitäten der Prozesse sollen dann in einem zweiten Schritt per „Plug & Play“ [AAD+04, RBFD01] Anwendungsfunktionen zugeordnet werden (diese müssen eventuell zuerst noch implementiert werden, wenn es sich um Funktionen handelt, die bisher noch nicht benötigt wurden). Die Trennung dieser beiden Schritte dient dazu, dass sich Prozessmodellierer, die die Abläufe definieren, nicht um Implementierungsdetails kümmern müssen, und umgekehrt diejenigen, die den Aktivitäten Anwendungsfunktionen zuordnen, nicht die genauen Abläufe kennen müssen. Bei der Zuordnung von Anwendungsfunktionen zu Aktivitäten sollte das Prozessmodellierungswerkzeug Unterstützung bieten, indem es Abhängigkeiten zwischen Anwendungsfunktionen (siehe Abschnitt 4.2) berücksichtigt und damit z.B. die Anordnung von Anwendungsfunktionen in nicht erlaubter Reihenfolge verhindert. Das Prozessmodellierungswerkzeug soll bei der Modellierung des Datenflusses durch die Auswertung der Schnittstelleninformationen der Anwendungsfunktionen unterstützen oder im Idealfall automatisch den benötigten Datenfluss ermitteln. Außerdem soll es ermitteln können, ob der modellierte Prozess überhaupt ausführbar ist. Dies alles soll versierte Prozessmodellierer unterstützen und es weniger erfahrenen Personen erst ermöglichen, Prozesse zu modellieren. Im Endeffekt lassen sich Prozesse schneller und kostengünstiger modellieren, da der Aufwand für die Modellierung selbst, aber auch der Aufwand für Tests gesenkt werden kann.

Im Abschnitt 4.1 werden verschiedene Prozessbeschreibungssprachen und ihre Eignung im Rahmen von VAS analysiert. In Abschnitt 4.2 werden Abhängigkeiten von Anwendungsfunktionen von Faktoren untersucht, die einer Workflow-Engine und einem Werkzeug zur Prozessmodellierung üblicherweise nicht bekannt sind. Darin wird beschrieben, um welche Abhängigkeiten es sich dabei handelt und wie sie beschrieben werden können, sodass ihre Einhaltung automatisch überprüft werden kann.

4.1 Prozessbeschreibungssprache

In diesem Abschnitt wird behandelt, welche Sachverhalte modelliert werden können müssen, um alle Aspekte der Prozesse in VAS abbilden zu können. Hintergrund ist VAS in seiner aktuellen Version (d.h. ohne Workflow-Engine). Es werden auch die beiden Prozessbeschreibungssprachen „Business Process Execution Language for Web Services“ (kurz BPEL4WS) [ACD+03] in der Version 1.1, spezifiziert von BEA Systems, IBM, Microsoft, SAP und Siebel und „XML Process Definition Language“ (kurz XPDL) [WfMC02] in der Version 1.0, spezifiziert von der Workflow Management Coalition betrachtet. Ebenso werden die in den Projekten ADEPT [Ade] bzw. AristaFlow [Ari] entwickelten Konzepte und Prototypen betrachtet.

Allgemein gilt, dass jeder in einer Programmiersprache definierbare Ablauf auch von einer Workflow-Engine abgewickelt werden kann. Unterstützt eine Workflow-Engine ein bestimmtes Konstrukt (z.B. Schleifen) nicht, kann dieses nachgebildet werden, in dem einige Anwendungsfunktionen zu einer einzigen zusammengefasst werden, in der das Konstrukt fest implementiert wird. Dies widerspricht natürlich dem Ziel, das mit dem Einsatz von Workflow-Engines erreicht werden soll, nämlich der Separation von Abläufen und Funktionen. Es zeigt aber, dass der Einsatz einer Workflow-Engine die Funktionalität einer Anwendung nicht prinzipiell einschränkt. Stattdessen ist der Einsatz einer Workflow-Engine mehr oder weniger sinnvoll und bedeutet mehr oder weniger Aufwand, je nachdem, welche Konstrukte von einer Anwendung benötigt werden und welche die Workflow-Engine anbietet.

4.1.1 Aspekte von Prozessbeschreibungssprachen

[Jabl95] definiert sechs sachliche Aspekte von Workflow-Modellen: Der *funktionale Aspekt* definiert, was ausgeführt werden soll, d.h. logische Verarbeitungseinheiten. Dabei handelt es sich um elementare Aktivitäten, denen eine Anwendungsfunktion zugeordnet ist, und um Aktivitäten, die selbst wieder einen Prozess repräsentieren. Der *operationale Aspekt* definiert die zur Ausführung einer Aktivität benötigten Programme und Ressourcen, d.h. Anwendungsfunktionen, Mitarbeiter und benötigte Hilfsmittel. Der *verhaltensbezogene Aspekt* definiert die Abhängigkeiten zwischen Aktivitäten, d.h. den Kontrollfluss eines Prozesses mit Sequenzen, alternativen und parallelen Verzweigungen, Schleifen und anderen Konstrukten. Außerdem definiert er die Zustandsübergänge von Prozessen und Aktivitäten zur Laufzeit. Der *informationsbezogene Aspekt* definiert den Datenfluss, d.h. Ein- und Ausgabeparameter von Anwendungsfunktionen und deren Abbildung auf Prozessvariablen. Außerdem werden die Datentypen festgelegt, die verwendet werden können. Der *organisatorische Aspekt* beschreibt die Aufbauorganisation eines Unternehmens oder einer Abteilung. Der *kausale Aspekt* macht Aussagen über die Gründe, warum ein Prozess auf eine bestimmte Weise modelliert wurde. Er erfasst beispielsweise Unternehmensvorschriften und gesetzliche Rahmenbedingungen. Aufgabe der Workflow-Engine ist, bei allen Prozessen zu überprüfen, ob die zu Grunde gelegten Kausalitäten noch zutreffen

oder verändert wurden. Zusätzlich zu diesen sechs sachlichen Aspekten werden noch zwei technische Aspekte definiert. Der *historische Aspekt* definiert, wann welche Ausführungsdaten protokolliert werden. Bei den meisten Workflow-Engines ist dies fest vorgegeben und damit für alle Prozesse gleich. Der *transaktionale Aspekt* legt Transaktionen innerhalb eines Prozesses, Kompensationsaktivitäten, etc. fest.

Im Folgenden wird untersucht, welche Funktionalität eine Workflow-Engine bzw. eine Prozessbeschreibungssprache in den einzelnen Aspekten bieten muss, um den Anforderungen von VAS gerecht zu werden.

Der funktionale Aspekt beschreibt eine Hierarchie geschachtelter Prozesse, wobei an unterster Ebene Prozesse liegen, die nur elementare Aktivitäten aufrufen, aber keine Sub-Prozesse. VAS stellt daran keine besonderen Anforderungen. Es bleibt ohnehin dem Prozessmodellierer überlassen, ob er einen Prozess in eine Reihe von Subprozessen unterteilt oder in einem einzigen großen Modell definiert.

Der operationale Aspekt ist bei der Definition der Prozesse erster Ebene wichtig, da für die Aktivitäten dieser Prozesse definiert werden muss, an welchem Terminal sie ausgeführt werden können. Für Prozesse zweiter Ebene ist der operationale Aspekt insofern relevant, dass den Aktivitäten der Prozesse zweiter Ebene eine Anwendungsfunktion zugeordnet werden können muss. Dies ist bei den Prozessen erster Ebene nicht notwendig und bei den Prozessen zweiter Ebene muss den Aktivitäten kein Bearbeiter zugeordnet werden, da die Prozesse vollständig an einem Terminal laufen und von einem Bearbeiter ausgeführt werden.

An den Kontrollfluss, d.h. den verhaltensbezogenen Aspekt, werden keine besonderen Anforderungen gestellt. Für die Prozesse zweiter Ebene müssen Schleifen und von den Werten von Prozessvariablen abhängige alternative Verzweigungen modelliert werden können. Für die Prozesse erster Ebene werden parallele Verzweigungen benötigt. Dies wird von den meisten Workflow-Engines bzw. Prozessbeschreibungssprachen unterstützt. Es ist allerdings von der Art der Modellierung des Kontrollflusses abhängig, wie gut Prozesse auf Korrektheit überprüft werden können. So sind z.B. mit regelbasierten Prozessbeschreibungen definierte Prozesse nur schlecht überblickbar und Deadlocks und andere Probleme können nur eingeschränkt analysiert werden. Die in IBM FlowMark und IBM WebSphere MQ Workflow [WMW] eingesetzten Aktivitätennetze erschweren die Modellierung von Prozessen dadurch, dass alternative und parallele Verzweigungen nicht getrennt voneinander modelliert werden. Dadurch kann es leicht passieren, dass einzelne Instanzen eines Prozesses ungewollt früh terminieren, ohne dass dies von der Workflow-Engine als fehlerhaft erkannt wird. Die blockstrukturierte Modellierung von Prozessen, wie sie im ADEPT-Projekt, BPEL4WS oder (eingeschränkt) in XPDL eingesetzt wird, kann korrekte Modellierung und Überprüfung der Prozesse gegenüber anderen Prozessbeschreibungssprachen stark vereinfachen. Allerdings reicht die Ausdrucksmächtigkeit blockstrukturierter Prozessbeschreibungssprachen nicht immer an die Mächtigkeit anderer Sprachen heran.

Der informationsbezogene Aspekt betrifft die Datentypen, die verwendet werden können, die Übergabe der Daten zwischen Anwendungsfunktionen und Workflow-Engine und die Möglichkeit, an alternativen Verzweigungen anhand von Daten zu entscheiden. Es gibt in allen hier betrachteten Prozessbeschreibungssprachen (BPEL4WS, XPDL und das ADEPT-Metamodell) die Möglichkeit, Prozessvariablen zu definieren, Ausgabeparameter von Anwendungsfunktionen in den Prozessvariablen zu speichern und Eingabeparameter von Anwendungsfunktionen mit den in Prozessvariablen gespeicherten Werten zu versorgen. Ebenso ist es möglich, Abbruchbedingungen von Schleifen und Entscheidungen alternativer Verzweigungen auf Basis der Werte von Prozessvariablen zu formulieren. Unterschiede gibt es allerdings bei den unterstützten Datentypen. Während ADEPT (zumindest momentan) nur Ganzzahlen und Zeichenketten unterstützt, werden in XPDL und BPEL4WS auch andere einfache Datentypen und komplexe Datentypen wie Listen und Records unterstützt. Welche Datentypen werden für VAS benötigt?

Auf jeden Fall werden die einfachen Datentypen String und Integer benötigt, um Zeichenketten und ganze Zahlen verwalten zu können. Ebenso ist ein Datentyp für Fließkomma-Zahlen notwendig.

Boolesche Variablen sind prinzipiell nicht notwendig, da sie durch Integer-Variablen ersetzt werden können. Allerdings hat das den Nachteil schlechterer Überprüfbarkeit: Bei einer alternativen Verzweigung, die anhand des Werts einer booleschen Variable entschieden werden soll, ist klar, dass nur zwei alternative Zweige existieren können. Wird aber anhand einer Ganzzahl-Variablen entschieden, sind mehr Zweige möglich, und es kann leicht passieren, dass die Entscheidungsvariable zur Laufzeit einen Wert hat, dem kein Zweig zugeordnet ist. Diese Problematik lässt sich zwar durch die Definition eines Default-Zweigs entschärfen, bei Verwendung von booleschen Variablen (wo dies möglich ist) tritt sie aber erst gar nicht auf.

Zusätzlich zu diesen einfachen Datentypen werden auch komplexe, recordwertige Datentypen benötigt. Hintergrund dafür ist, dass in VAS alle Daten in einer Datenbank gespeichert sind, wobei einige Tabellen sehr viele (teilweise über 100) Felder haben können. Eine in VAS eingesetzte Workflow-Engine muss Datensätze dieser Tabellen verwalten können. Können keine komplexen Daten von der Workflow-Engine verwaltet werden, müssen diese durch eine Menge einfacher Variablen ersetzt werden. Bei so großen Tabellen, wie sie in VAS vorhanden sind, kann dies aber keinem Prozessmodellierer zugemutet werden. Außerdem müssten dann auch die Schnittstellen der Anwendungsfunktionen, die solche Daten verarbeiten, sehr viele Parameter erhalten. Dies wird schnell unübersichtlich und erschwert die Programmierung der Anwendungsfunktionen. Da auf Basis einzelner Felder der Datensätze alternative Verzweigungen entschieden werden, ist es auch nicht möglich, die Daten außerhalb der Workflow-Engine zwischen den Anwendungsfunktionen zu übertragen.

Prinzipiell muss zwischen den von einer Prozessbeschreibungssprache unterstützten Datentypen und den Datentypen, die eine Workflow-Engine verwalten kann, unterschieden werden. Beispielsweise muss eine Workflow-Engine selbst nicht unbedingt recordwertige Datentypen verwalten können.

Sie kann intern recordwertige Variablen durch eine Menge von einfachen Variablen ersetzen. Allerdings muss bei der Kommunikation mit Anwendungsfunktionen dafür gesorgt werden, dass die Menge von einfachen Variablen zu einem recordwertigen Datum konvertiert wird. Ebenso können alle einfachen Datentypen in Zeichenketten umgewandelt werden, sodass eine Workflow-Engine prinzipiell nur diese verwalten können muss. Dies kann allerdings natürlich die Ausführungsgeschwindigkeit von Prozessen senken, da für die Konvertierung von Datentypen zusätzliche Zeit benötigt wird.

Der organisatorische Aspekt ist nur für die Prozesse erster Ebene von Relevanz. Für diese ist es notwendig, alle Terminals und deren Fähigkeiten, ausgedrückt als Rollen (z.B. „Einfahrtsterminal“, „Ausfahrtsterminal“) zu erfassen, sodass zur Laufzeit ermittelt werden kann, an welchen Terminals eine Aktivität eines Prozesses erster Ebene ausgeführt werden kann. Dafür reicht allerdings ein relativ einfaches Organisationsmodell aus. Anders als bei der Modellierung von Unternehmen, wobei sehr viele unterschiedliche Einheiten betrachtet werden müssen (Abteilungen, Stellen, Rollen, Mitarbeiter etc.), muss außer den Terminals nur deren Rolle erfasst werden.

Der kausale Aspekt ist im Rahmen von VAS vor allem bei der Beschreibung von Abhängigkeiten zwischen Anwendungsfunktionen relevant. Dazu sei auf Abschnitt 4.2 verwiesen.

Der historische Aspekt eines Workflow-Modells ist vor allem bei Prozessen erster Ebene interessant. Auf Basis der protokollierten Ausführungsdaten kann ermittelt werden, welche Lkw gerade im Werk sind und wo sie sich ungefähr befinden. Außerdem können statistische Auswertungen auf Basis der Historie einer Menge von Prozessen durchgeführt werden, um z.B. die durchschnittliche Verweildauer von Lkw im Werk oder besonders stark frequentierte Terminals zu ermitteln. In der Regel kann in einer Prozessbeschreibung nicht definiert werden, welche Ausführungsdaten protokolliert werden sollen. Stattdessen ist dies von den meisten Workflow-Engines fest vorgegeben oder kann für die Workflow-Engine konfiguriert werden.

Der transaktionale Aspekt ist sowohl für die Prozesse erster als auch die Prozesse zweiter Ebene relevant. Dabei geht es vor allem darum, den Abbruch von Prozessen zweiter Ebene oder das Fehlschlagen von Anwendungsfunktionen korrekt zu behandeln. Es muss immer sichergestellt sein, dass nach Abbruch eines Prozesse zweiter Ebene der Lkw trotzdem weiter behandelt werden kann und dass der Zustand eines Lkw immer dem Zustand seines Prozesses erster Ebene entspricht. So darf für einen Lkw, dessen Einfahrtsprozess abgebrochen wurde, keine Prozessinstanz erster Ebene existieren, da dies bedeutet, dass der Lkw im Werk ist. Damit würde jeder weitere Versuch ins Werk zu kommen scheitern.

Damit muss der Abbruch eines Prozesses zweiter Ebene oder einer Anwendungsfunktion immer angemessen behandelt werden, in der Regel durch das Zurücksetzen des Prozesses, z.B. mittels geeigneter Kompensationsaktivitäten. Ein weiterer Gesichtspunkt ist die Absicherung mehrere laufender Prozessinstanzen gegeneinander. So darf es nicht passieren, dass zwei Lkw, die gleichzeitig an unterschiedlichen Terminals ihre Einfahrtsabwicklung durchführen, die gleiche Lieferung auswählen.

Werden Transaktionen von einer Workflow-Engine nicht unterstützt, kann trotzdem mit anderen Mitteln ein gewisser Grad an Sicherheit erreicht werden. Werden z.B. die Daten, die während der Ausführung eines Prozesses verändert werden, nur im Arbeitsspeicher gehalten und erst am Ende des Prozesses persistent gespeichert, kann sichergestellt werden, dass die Daten des Systems nicht inkonsistent werden. Dafür müssen aber der Prozessmodellierer und die Entwickler von Anwendungsfunktionen sorgen.

4.1.2 BPEL4WS

BPEL4WS wurde entwickelt, um Prozesse zu beschreiben, die auf Web Services basieren. Der Fokus liegt deshalb auf Prozessen, die vollautomatisch, d.h. ohne Interaktion mit menschlichen Akteuren, ablaufen. Die einzelnen Prozessschritte sind Web Services, die in WSDL beschrieben sind, per UDDI aufgefunden werden können und mit denen per SOAP kommuniziert wird. Außerdem wird nicht nur der Fall betrachtet, in dem ein Prozess abgewickelt wird und dabei eine Anzahl von Anwendungsfunktionen aufgerufen werden, sondern es ist auch möglich, kommunizierende Prozesse zu definieren. Dies wird dadurch ermöglicht, dass Web Services sowohl synchron als auch asynchron aufgerufen werden können und außerdem Nachrichten versandt und auf eingehende Nachrichten gewartet werden kann. Damit können gleichzeitig laufende Prozesse synchronisiert werden.

BPEL4WS unterstützt den funktionalen Aspekt durch das Konzept von Aktivitäten. Diese dienen unter anderem dazu, einen Web Service aufzurufen, auf eine Nachricht zu warten oder eine abzuschicken. Da ein in BPEL4WS modellierter Prozess als ein Web Service verwendet werden kann, ist auch die Komposition mehrerer Prozesse zu einem neuen Prozess möglich.

Für den operationalen Aspekt, d.h. die Definition von Anwendungsfunktionen, verwendet BPEL4WS ein relativ komplexes Konzept. Es werden so genannte Partner Link Types eingeführt, die die Rollen zweier kooperierender Partner definieren. Eine Rolle verweist dabei auf einen WSDL Port Type, d.h. einen Web Service, bestehend aus einer Menge von Operationen. Basierend auf Partner Link Types werden Partner Links definiert, die dem Prozess und den Partner des Prozesses eine der beiden Rollen zuordnen. Um einer Aktivität eine Operation eines Web Service zuordnen zu können, werden bei dieser Aktivität ein Partner Link sowie ein Port Type und eine Operation einer WSDL-Definition angegeben. Damit ist definiert, welche Web Service Operation aufgerufen wird bzw., im Fall dass auf eine eingehende Nachricht gewartet wird, welche Web Service Operation für diesen Prozess aufgerufen werden muss.

Für die Definition des Kontrollflusses eines Prozesses bietet BPEL4WS umfangreiche Möglichkeiten. Es existieren besondere Aktivitäts-Typen, die Konstrukte wie Schleifen, alternative und parallele Verzweigungen oder eine Sequenz von Aktivitäten nachbilden. Da diese beliebig geschachtelt werden können, können die aus imperativen Programmiersprachen bekannten Konstrukte nachgebildet werden. Weitere Aktivitäts-Typen dienen dazu, eine definierte Zeit zu warten oder einen Prozess zu

beenden. Eine Besonderheit stellt dar, wie in BPEL4WS modellierte Prozesse gestartet werden können. Dazu werden eine oder mehrere Aktivitäten, die keine Vorgänger-Aktivitäten besitzen, zu initialen Aktivitäten erklärt. Diese können aufgerufen werden, auch wenn keine Prozess-Instanz existiert, und durch den Aufruf wird implizit eine Prozessinstanz erzeugt. Prozessinstanzen werden durch so genannte Correlation Sets identifiziert. Dabei handelt es sich um Anwendungsdaten, die eine Prozessinstanz eindeutig identifizieren können, wie z.B. eine Bestellnummer. Im Kontext der Prozesse in VAS könnte ein Correlation Set aus der Identifikationsnummer eines Lkw-Fahrers bestehen. Beim Erzeugen einer Prozess-Instanz durch den Aufruf einer initialen Aktivität wird ebenso wie beim Aufruf von Web Service Operations ein Correlation Set als Teil einer Nachricht mit übergeben.

Der Datenfluss wird in BPEL4WS vor allem durch die WSDL-Definitionen der verwendeten Web Services bestimmt. Diese definieren für alle Operationen eines Web Services Formate für ankommende und ausgehende Nachrichten. Da eine Web Service Operation eine Nachricht empfängt, wenn sie aufgerufen wird und eine Nachricht zurückschickt, wenn sie beendet ist, haben alle Aktivitäten in BPEL4WS (maximal) einen Eingabe- und einen Ausgabeparameter. Diese können aber beliebig komplex sein. Ebenso können Prozessvariablen in BPEL4WS beliebig komplex sein, da sie als Typ eine in WSDL definierte Nachricht oder ein beliebiges XML Schema haben können. Ein- und Ausgabeparameter einer von einer Aktivität aufgerufenen Web Service Operation werden mit dem Wert einer Prozessvariable versorgt bzw. in einer Prozessvariable gespeichert. Zusätzlich gibt es eine spezielle Aktivität, die es erlaubt, Prozessvariablen mit dem Wert einer anderen Variablen oder einer Konstanten zu belegen.

Der transaktionale Aspekt wird von BPEL4WS durch die Konzepte von Fault Handlers und Compensation Handlers unterstützt. Eine Web Service Operation kann anstatt einer normalen Nachricht auch eine Fehlernachricht senden, die das Fehlschlagen der Operation signalisiert. Für solche Fehlernachrichten können Fault Handler definiert werden, die aufgerufen werden, wenn die Fehlernachricht auftritt. Darin kann angemessen auf die Fehlernachricht reagiert werden. Compensation Handler definieren für eine Aktivität oder eine Gruppe von Aktivitäten einen Vorgang, der die Auswirkungen der Aktivitäten zurücknehmen kann. Ein Compensation Handler kann z.B. aus einem Fault Handler heraus aufgerufen werden, sodass das Fehlschlagen einer Aktivität zum Zurücksetzen dieser Aktivität oder eines größeren Bereichs führen kann. Mit diesen Mitteln kann relativ flexibel auf Fehler reagiert werden. Die Absicherung mehrerer Prozessinstanzen gegeneinander wird von BPEL4WS nicht unterstützt, d.h., es müssen andere Verfahren eingesetzt werden, um konkurrierende Zugriffe auf Daten zu verhindern.

Der organisatorische Aspekt wird in BPEL4WS nicht berücksichtigt, da nur automatisch ablaufende Prozesse betrachtet werden, an denen keine Personen beteiligt sind. Ebenso wird der historische Aspekt nicht berücksichtigt. Es bleibt den Anbietern von Workflow-Engines überlassen, welche Aus-

führungsdaten protokolliert werden. Der kausale Aspekt ist, wie oben erwähnt, in diesem Zusammenhang nicht von Belang.

BPEL4WS bietet insgesamt eine mächtige Sprache, um Prozesse zu definieren. Die Möglichkeiten der Kontrollflussmodellierung übersteigen die Anforderungen durch VAS, ähnliches gilt für den Datenfluss. Durch die Möglichkeiten, die Fault Handler und Compensation Handler bieten, ist auch der transaktionale Aspekt gut abgedeckt, wenn auch konkurrierende Zugriffe auf Daten durch unterschiedliche Prozessinstanzen nicht verhindert werden können, was allerdings auch mit anderen Prozessbeschreibungssprachen nicht spezifiziert werden kann. Für die Prozesse erster Ebene ist BPEL4WS ziemlich gut geeignet, da die Identifikation von Prozessinstanzen durch Correlation Sets die Zuordnung von Prozessinstanzen zu Lkw bzw. Lkw-Fahrern erlaubt. Die Möglichkeit, Prozessinstanzen implizit zu starten und Aktivitäten zu definieren, die auf externe Ereignisse warten, ist gut geeignet, um den Sachverhalt zu modellieren, dass ein Prozess zweiter Ebene starten soll, wenn ein Lkw-Fahrer ein Terminal zu bedienen beginnt. Die größte Schwäche von BPEL4WS für den Einsatz in VAS ist die fehlende Zuordnung von Aktivitäten zu Bearbeitern, wodurch die Zuordnung von Aktivitäten der Prozesse erster Ebene zu Terminals nicht unterstützt wird. Bei mit BPEL4WS modellierten Prozessen müssen alle Anwendungsfunktionen als Web Services implementiert sein und die Kommunikation zwischen Workflow-Engine und einzelnen Anwendungsfunktionen muss über das relativ aufwendige SOAP-Protokoll stattfinden. Das führt einerseits zu einem großen Implementierungsaufwand, da alle Anwendungsfunktionen, wie z.B. Bildschirmmasken, als Web Services implementiert werden müssen. Andererseits entsteht ein so großer Kommunikationsaufwand, dass es unwahrscheinlich ist, dass die Prozesse zweiter Ebene in akzeptabler Zeit weitergeschaltet werden. Ein weiterer vor allem für die Prozesse erster Ebene wichtiger Aspekt ist die Möglichkeit, laufende Prozessinstanzen zu verändern. Dies ist allerdings nicht abhängig von der verwendeten Prozessbeschreibungssprache, sondern von den Möglichkeiten, die eine Workflow-Engine bietet. Eine Prozessbeschreibungssprache kann Änderungen allenfalls mehr oder weniger gut unterstützen. Deshalb kann dazu keine Aussage zu BPEL4WS getroffen werden.

4.1.3 XPDL

XPDL wurde im Rahmen der Aktivitäten der Workflow Management Coalition definiert, um als Standardformat zur Definition von Prozessen und zum Import und Export der Prozessdefinitionen durch Workflow-Engines.

Der funktionale Aspekt wird von XPDL durch das Konzept von Aktivitäten unterstützt. Diesen kann sowohl eine Anwendungsfunktion als auch ein Sub-Prozess zugeordnet werden, wodurch die Komposition von Prozessen zu einem neuen Prozess unkompliziert möglich ist.

XPDL definiert Anwendungsfunktionen (Applikationen genannt) sehr abstrakt. Es gibt zwei Möglichkeiten: Entweder werden nur die formalen Parameter definiert, die eine Anwendungsfunktion hat,

und die konkrete Definition der Anwendungsfunktion wird den Workflow-Engines überlassen. Alternativ dazu kann auf ein externes Dokument verwiesen werden, das die Anwendungsfunktion definiert. Das kann z.B. ein WSDL-Dokument sein, das einen Web Service definiert oder eine Datei, die die Schnittstelle einer in z.B. C oder Java geschriebenen Komponente definiert. XPDL abstrahiert von diesen Details, um in unterschiedlichen Umgebungen einsetzbar zu sein. Offensichtlicher Nachteil der Lösung ist, dass im Prinzip beliebige Definitionen für Anwendungsfunktionen verwendet werden können, die die Portabilität der XPDL-Definitionen einschränken. In der Version 2 der XPDL-Spezifikation wird dies durch die Einführung von einigen Standard-Typen für Applikationen etwas verbessert. So können direkt in XPDL Web Services, Enterprise Java Beans und einige andere Applikationen definiert werden. Generell wird die Anbindung von Anwendungsfunktionen an die Workflow-Engine durch zwei andere Standards der WfMC bzw. der OMG [WfMC98b, OMG02] abgedeckt, weshalb dieses Thema in der XPDL-Spezifikation nicht detaillierter behandelt wird.

Der Kontrollfluss wird in XPDL durch eine Kombination von Aktivitäten und Transitionen definiert, wodurch ein gerichteter Graph definiert entsteht. Jede Aktivität kann prinzipiell beliebig viele eingehende und ausgehende Transitionen besitzen. Bei Aktivitäten mit mehr als einer ausgehenden Transition muss definiert werden, ob beim Ende der Aktivität nur einer der Transitionen gefolgt wird oder allen. Damit können alternative und parallele Verzweigungen modelliert werden. Durch Zyklen im Graphen können Schleifen modelliert werden. Durch die Notation von Bedingungen an Transitionen kann die Auswahl bestimmter Zweige, z.B. bei alternativen Verzweigungen erreicht werden. In XPDL sind drei Konformitätsklassen für Prozessdefinitionen mit unterschiedlichen Anforderungen an den Kontrollfluss definiert. Für die Konformitäts-Klasse *Non-Blocked* bestehen keine Einschränkungen. Der Kontrollfluss von Prozessen in der Konformitäts-Klasse *Loop-Blocked* bildet einen azyklischen gerichteten Graphen, es sind also keine Schleifen erlaubt. In der Konformitäts-Klasse *Full-Blocked* muss zu jeder alternativen oder parallelen Verzweigung eine Zusammenführung der Zweige gleichen Typs existieren. Außerdem sind für parallele Verzweigungen keine Bedingungen an den Transitionen erlaubt und bei alternativen Verzweigungen muss durch Definition eines Default-Zweigs sichergestellt werden, dass in jedem Fall genau ein Zweig ausgewählt wird. Durch die Definition der Konformitäts-Klassen erlaubt es XPDL, zwischen leichter überprüfbaren Prozessdefinitionen und größerer Beschreibungsmächtigkeit zu wählen. Eine Schwäche stellt dar, dass nur in der Konformitäts-Klasse Non-Blocked Schleifen modelliert werden können. Das heißt, dass alle Prozessmodelle, die Schleifen enthalten, nur schlecht überprüft werden können. Abhilfe dazu schafft die Version 2 von XPDL, in der für eine Aktivität (auch eine Aktivität, die einen Subprozess repräsentiert) definiert werden kann, dass sie mehrfach ausgeführt werden soll (z.B. bis ein bestimmtes Abbruchkriterium erreicht wurde). Dadurch können Schleifen in allen Konformitäts-Klassen modelliert werden.

Der Datenfluss wird in XPDL mittels Ein- und Ausgabeparameter von Anwendungsfunktionen und Prozessvariablen definiert. Bei der Definition von Anwendungsfunktionen (in XPDL Applikatio-

nen genannt) können die Ein- und Ausgabeparameter definiert werden, die die Anwendungsfunktion erwartet bzw. zurückgibt. Bei Aktivitäten wird definiert, mit den Werten welcher Prozessvariablen Eingabeparameter belegt werden und in welchen Prozessvariablen die Ausgabeparameter gespeichert werden sollen. In XPDL können neben einfachen Datentypen auch komplexe Datentypen (z.B. Listen, Records, Aufzählungstypen) definiert werden. Außerdem ist es möglich, Datentypen in XML Schema zu definieren.

Für die Definition des organisatorischen Aspekts bietet XPDL ein einfaches Modell. Es können so genannte Participants definiert werden, die einen Mitarbeiter, eine Rolle oder eine organisatorische Einheit (z.B. eine Abteilung) repräsentieren. Diese werden aber nicht weiter definiert, z.B. wird nicht festgelegt, welche Mitarbeiter eine Rolle innehaben oder zu einer Abteilung gehören. Dies wird der Workflow-Engine überlassen. Einer Aktivität kann ein Participant zugeordnet werden. Für VAS reichen die Möglichkeiten zur Organisationsmodellierung aus, falls eine Workflow-Engine eingesetzt wird, die die Auflösung von Rollen auf einzelne Mitarbeiter unterstützt.

Der historische, der transaktionale und der kausale Aspekt werden von XPDL nicht definiert, dies wird den Workflow-Engines überlassen bzw. im Fall des historischen Aspekts, von einer anderen Spezifikation der WfMC [WfMC98] abgedeckt. Die Definition von Transaktionen wird in der Version 2 von XPDL möglich, auch wenn es noch keine vollständige Lösung dafür gibt. In der Spezifikation wird konstatiert: „Das exakte Verhalten und die genaue Notation für die Definition von Transaktionen sind noch ein ungelöstes Problem.“

XPDL ist, was Kontroll- und Datenfluss betrifft, bei weitem mächtig genug, um die Anforderungen von VAS zu erfüllen. Gesichtspunkte wie die Anbindung von Anwendungsfunktionen, die Protokollierung von Ausführungsdaten oder (teilweise) die Zuordnung von Akteuren zu Aktivitäten werden von XPDL nicht spezifiziert. Deshalb kann bei einer Workflow-Engine, die XPDL unterstützt, generell keine Aussage getroffen werden, wie gut sie für den Einsatz in VAS geeignet ist. Vielmehr muss ihre Funktionalität bezüglich dieser Aspekte mit berücksichtigt werden. Die Unterstützung der Änderung von laufenden Prozessinstanzen ist von der verwendeten Prozessbeschreibungssprache unabhängig. Allerdings sind keine Workflow-Engines bekannt, die XPDL und Änderungen an laufenden Instanzen unterstützen (gleiches gilt für BPEL4WS).

4.1.4 ADEPT

Das ADEPT-Metamodell wurde unter anderem mit den Zielen Formalisierbarkeit und Analysier- bzw. Validierbarkeit entwickelt, sodass präzise Aussagen über das Systemverhalten möglich sind und automatisch ermittelt werden können. Damit sollen auch Fehlerquellen bereits zur Modellierzeit so weit wie möglich ausgeschlossen werden. Allerdings muss ein Kompromiss zwischen Umfang und Komplexität der Korrektheitsanalysen und der Ausdrucksmächtigkeit des Metamodells gefunden werden.

Ein weiteres Ziel war die Verständlichkeit der modellierten Prozesse, sodass diese semantisch auf Benutzerebene validiert werden können [Reic00].

Der funktionale Aspekt wird dadurch unterstützt, dass Aktivitäten, die elementaren Bausteine eines ADEPT-Prozessmodells, entweder elementare Anwendungsfunktionen oder Subprozesse repräsentieren können.

Das ADEPT-Metamodell kennt keine so starke Trennung zwischen Aktivität und Anwendungsfunktion wie z.B. XPDL. Stattdessen können so genannte Aktivitätenvorlagen definiert werden, die der Definition der Anbindung einer Anwendungsfunktion entsprechen, die also z.B. festlegen, wie eine Anwendungsfunktion aufgerufen wird und welche Ein- und Ausgabeparameter sie erhält. Eine Aktivität in einem Prozessschema besteht dann im Prinzip aus einem Namen für die Aktivität und der Referenz auf eine Aktivitätenvorlage. Da die Aktivitätenvorlagen unabhängig von Prozessschemata definiert werden, müssen sie nur einmal angelegt werden und können dann in allen Prozessschemata verwendet werden.

Der Kontrollfluss wird in ADEPT im Prinzip als gerichteter Graph aus Aktivitäten und Kanten definiert. Allerdings bestehen sehr enge Restriktionen, sodass nur blockstrukturierte Graphen dem ADEPT-Modell entsprechen. Diese können mit XPDL-Prozessdefinitionen verglichen werden, die der Konformitätsklasse Full-Blocked entsprechen. Als Konstrukte werden außer Sequenzen, alternativen und parallelen Verzweigungen und Schleifen auch so genannte Synchronisationskanten unterstützt, die der Synchronisation zwischen Aktivitäten paralleler Zweige dienen. Außerdem werden Konstrukte zur Behandlung bestimmter Ausnahmesituationen und zur Behandlung von Fehlern angeboten, die hier aber weniger relevant sind.

Der Datenfluss wird im ADEPT-Metamodell wie in XPDL und BPEL4WS modelliert. Es existieren Prozessvariablen und bei jeder Aktivität wird definiert, mit den Werten welcher Variablen die Eingabeparameter versorgt werden bzw. in welche Prozessvariablen die Ausgabeparameter geschrieben werden. Der aktuell verfügbare ADEPT-Prototyp unterstützt nur die Datentypen String und Integer sowie einen Datentyp für Referenzen auf externe Daten. Dies erschwert den Einsatz in VAS, da alle nicht unterstützten Datentypen wie z.B. Fließkommazahlen von Anwendungsfunktionen in Zeichenketten umgewandelt werden müssen, bevor sie der Workflow-Engine übergeben werden können. In späteren Versionen des Prototyps soll dieser Mangel allerdings behoben werden.

Für die Organisationsmodellierung bietet der ADEPT-Prototyp ein relativ mächtiges Metamodell. Darin werden hierarchisch geordnete Organisationseinheiten und diesen zugeordnete Stellen verwaltet. Den Stellen können, bezogen auf einen bestimmten Zeitraum, Mitarbeiter zugeordnet werden. Dies wird unterstützt durch die Zuordnung von beschreibenden Rollen zu Stellen und Mitarbeitern. Außerdem werden Rollen und Mitarbeitern Fähigkeiten zugeordnet. Die Zuordnung von Bearbeitern zu Aktivitäten wird mittels einer eigenen Sprache festgelegt, die auf Basis der Konzepte des Organisationsmodells (Organisationseinheiten, Rollen etc.) jede beliebige Zuordnung von Mitarbeitern zu Terminals

ermöglicht. Dazu kann eine Menge von Eigenschaften (Zugehörigkeit zu einer bestimmten Organisationseinheit, Besitz einer bestimmten Rolle oder bestimmter Fähigkeiten, …) definiert werden, die ein Mitarbeiter besitzen muss, der die Aktivität ausführen darf. Außerdem kann eine Menge solcher Definitionen, versehen mit einer Priorität, angegeben werden. Dann wird ein Mitarbeiter ausgewählt, der der Definition mit der höchsten Priorität entspricht, und falls kein solcher anwesend ist, ein Mitarbeiter, der einer Definition mit niedrigerer Priorität entspricht.

Was den transaktionalen Aspekt angeht, werden im ADEPT-Prototypen Ausführungsdaten sehr ausführlich protokolliert. Dies dient dazu, Änderungen an laufenden Prozessinstanzen zu ermöglichen und dabei eine ganzen Reihe von Korrektheitskriterien überprüfen zu können. Es ermöglicht auch, eine Prozessinstanz in jeden beliebigen Zustand, der bisher durchlaufen wurde, zurücksetzen zu können. Das Zurücksetzen einer Prozessinstanz wird durch so genannte Fehlerrücksprungkanten im Prozessmodell festgelegt. Diese definieren, wie auf einen Fehler in einer Aktivität reagiert werden soll, indem festgelegt wird, welche Aktivitäten zurückgesetzt werden sollen. Im Extremfall kann eine ganze Prozessinstanz abgebrochen und zurückgesetzt werden. Darüber hinaus existieren keine weiteren Konzepte für Transaktionen. Allerdings wird gerade daran gearbeitet, das Konzept der Fehlerrücksprungkanten durch ein mächtigeres Transaktionskonzept zu ersetzen.

Im ADEPT-Projekt wurde eine Reihe interessanter Konzepte entwickelt, die teils die im Abschnitt 4.1.1 dargestellten Aspekte betreffen, teils aber auch darüber hinausgehen. So kann z.B. für alle Ein- und Ausgabeparameter von Aktivitätenvorlagen definiert werden, ob sie obligatorisch oder optional sind. Eine Aktivität kann ausgeführt werden, wenn alle obligatorischen Eingabeparameter mit Werten versorgt sind. Für die Überprüfung dieser und einiger anderer Prozesseigenschaften wurden Algorithmen entwickelt. So kann zur Modellierzeit anhand eines Prozessmodells sichergestellt werden, dass alle obligatorischen Eingabeparameter von Aktivitäten mit Werten versorgt werden oder dass ein Prozessmodell keine Deadlocks enthält und immer korrekt terminiert. Ein weiteres Konzept stellen Priorisierungskanten und Aktivitätenprioritäten dar. Diese dienen dazu, zur Modellierzeit festzulegen, dass in Ausnahmen bestimmte Aktivitäten im Prozess übersprungen werden können und, je nach Modellierung, später nachgeholt oder ganz ausgelassen werden. Weiter lassen sich zeitlichen Beschränkungen im Prozess festlegen, wie Mindest- oder Maximalabstände zwischen Aktivitäten oder die maximale Ausführungsdauer einer Aktivität.

Eines der zentralen Konzepte von ADEPT stellen Ad-hoc-Änderungen dar, d.h. die Änderung laufender Prozessinstanzen. Dabei kann das Schema einer laufenden Prozessinstanz in den noch nicht ausgeführten Teilen verändert werden. So können Aktivitäten entfernt oder neue Aktivitäten eingefügt werden. Dabei wird dafür gesorgt, dass die zur Modellierzeit gegebenen Garantien (z.B. keine Deadlocks, Versorgung aller obligatorischer Eingabeparameter aller Aktivitäten) nicht verletzt werden. Auf diesem Konzept aufbauend wurde ein Konzept entwickelt, das Änderungen an einem Prozessschema auf alle laufenden Instanzen des Schemas propagiert (zumindest alle Instanzen, die in ihrer Ausfüh-

rung nicht zu weit fortgeschritten sind). Dies ist auch bei Instanzen möglich, an denen bereits Ad-hoc-Änderungen ausgeführt wurden.

Die im ADEPT-Projekt entwickelten Konzepte sind sehr gut für eine in VAS eingesetzte Workflow-Engine geeignet, und zwar gleichermaßen für die Steuerung der Prozesse erster und zweiter Ebene. Einzige Schwäche stellt momentan das relativ schwache Transaktionskonzept dar. Allerdings weist der momentan existierende Prototyp von ADEPT deutliche Schwächen auf. Dazu gehört die Unterstützung für nur sehr wenige Datentypen und ein unzureichendes Werkzeug zur Modellierung von Prozessen. Dies soll allerdings in der nächsten Version des Prototypen, ADEPT2, deutlich verbessert werden.

4.2 Abhängigkeiten von Anwendungsfunktionen

Bisher wurden Anwendungsfunktionen als isoliert und unabhängig voneinander betrachtet. Dies ist aber in vielen Fällen und auch bei VAS nicht richtig, denn oft werden mehrere Anwendungsfunktionen von ein und derselben Komponente zur Verfügung gestellt. Diese können z.B. vom Zustand der Komponente abhängig sein oder müssen in einer bestimmten Reihenfolge aufgerufen werden, um erfolgreich ausgeführt werden zu können. Diese Abhängigkeiten erschweren die Modellierung von Prozessen. Kann ihre Einhaltung durch automatische Überprüfungen gesichert werden, können auch Personen, die die Abhängigkeiten zwischen Anwendungsfunktionen nicht genau kennen, Prozesse modellieren.

Automatisierte Überprüfungen sind überall dort sinnvoll, wo der Prozessmodellierer in seiner Freiheit, Prozesse beliebig zu modellieren, eingeschränkt ist. Solche Einschränkungen sind durch Bedingungen im Kontrollfluss (z.B. dürfen keine Deadlocks modelliert werden) oder dadurch gegeben, dass ein modellierter Prozess zur Laufzeit zu einer fehlerhaften Ausführung führen kann (z.B. wenn Eingabeparameter einer Aktivität nicht versorgt sind). Können Fehler oder potentielle Probleme in einem Prozessmodell zur Modellierzeit automatisch erkannt werden, wird verhindert, dass sie später während der Laufzeit einer Prozessinstanz auftreten. In [Reic00] wird vorgestellt, wie die Einhaltung vieler solcher Einschränkungen bereits bei der Modellierung oder durch Überprüfungen sichergestellt werden kann. Damit wird z.B. bereits zur Modellierzeit die Versorgung der Eingabeparameter aller Aktivitäten mit Werten und das korrekte Terminieren eines Prozesses garantiert. In den nächsten Abschnitten wird behandelt, was für Abhängigkeiten von Anwendungsfunktionen in VAS vorkommen und wie diese beschrieben werden können, sodass ihre Einhaltung automatisch überprüft werden kann.

4.2.1 Abhängigkeiten der Anwendungsfunktionen in VAS

In VAS wurden drei Arten von Abhängigkeiten erkannt. Sehr viele Anwendungsfunktionen in VAS greifen auf die gemeinsam genutzte Datenbank zu. Dabei greifen im Prozess später aufgerufene Anwendungsfunktionen auf Daten zu, die von früher im Prozess aufgerufenen Anwendungsfunktionen

geschrieben wurden. Sind die Daten, die eine Anwendungsfunktion lesen will, nicht vorhanden, tritt ein Fehler auf. Abhängigkeiten dieser Art, bei denen der Erfolg der Ausführung einer Anwendungsfunktion von der Existenz externer Daten abhängt, sollen im Folgenden *externe Datenabhängigkeiten* genannt werden.

Die zweite Art von Abhängigkeiten betrifft das Subsystem zur Erkennung von Fahrzeugen, das unterschiedliche Geräte zur Identifikation (Geräte zum Lesen von RFID-Tags, Barcode-Leser, Kartenspender und Karteneinzugsgeräte) verwendet. Die Anwendungsfunktionen von Kartenspendern und Karteneinzugsgeräten sind dabei vom internen Zustand des Geräts abhängig.

Die verwendeten Kartenspender und Karteneinzugsgeräte sind sehr ähnlich aufgebaut: Sie besitzen beide einen Stapel von Karten. Ein Karteneinzugsgerät kann eingezogene Karten zu seinem Stapel hinzufügen, ein Kartenspender kann eine Karte von seinem Stapel entnehmen. Zwischen Kartenstapel und dem Karteneinzugs- bzw. -ausgabeschlitz befindet sich eine Vorrichtung, um den Speicher von Karten zu beschreiben und auszulesen. Eine Karte kann nur gelesen bzw. beschrieben werden, wenn sie sich in der richtigen Position befindet, d.h. in der Schreib-/Lese-Vorrichtung. Beide Geräte können Karten einziehen (der Kartenspender auch von seinem Kartenstapel aus) und in die Schreib-/Lese-Position bringen, ebenso können sie Karten aus der Schreib-/Leseposition an den Benutzer ausgeben. Natürlich kann keine Karte eingezogen werden, wenn sich eine andere in der Schreib-/Lese-Position befindet, und es kann keine ausgegeben werden, wenn sich keine in der Schreib-/Lese-Position befindet. Alle Funktionen der beiden Geräte (Karte lesen, schreiben, einziehen oder auswerfen, Karte vom Stapel nehmen bzw. auf den Stapel legen) sind somit davon abhängig, ob sich gerade eine Karte in Schreib-/Lese-Position befindet oder nicht. Zusätzlich verändern einige der Funktionen (z.B. Karte einziehen) diesen Zustand. Diese Art von Abhängigkeiten, bei denen eine Anwendungsfunktion vom internen Zustand einer Komponente abhängig ist, sollen *Zustandsabhängigkeiten* genannt werden.

In VAS existiert eine Reihe von Software-Komponenten, von denen jede einzelne zur Ansteuerung einer Gruppe von Hardware-Komponenten mit ähnlicher Funktionalität dient. Sie bieten Funktionen zur Steuerung aller Funktionen der Hardware an. Da die durch die Software-Komponente ansteuerbaren Hardware-Komponenten nicht ganz identische Funktionalität besitzen, gibt es Software-Funktionen, die nur zusammen mit bestimmter Hardware ausführbar sind, mit anderer Hardware aber nicht. Zum Beispiel werden sämtliche Kartenleser, Kartenspender und Karteneinzugsgeräte von einer Software-Komponente angesteuert. Diese Komponente hat deshalb unter anderem Funktionen zum Einziehen oder Auswerfen einer Karte, die aber nur mit Kartenspendern bzw. Karteneinzugsgeräten funktionieren, nicht aber mit normalen Kartenlesern. Solche Abhängigkeiten, bei denen die erfolgreiche Ausführung von Funktionen einer Software-Komponente von der gerade vorhandenen Hardware abhängig sind, werden *Hardware-Abhängigkeiten* genannt.

Damit ein Prozessmodellierungswerkzeug eine Prozessvorlage auf die Berücksichtigung von Abhängigkeiten zwischen Anwendungsfunktionen überprüfen kann, müssen ihm diese Abhängigkeiten

bekannt gemacht werden. Es müssen also für jede Anwendungsfunktion zusätzlich zur Beschreibung von Ein- und Ausgabeparametern weitere Informationen zur Verfügung gestellt werden. In diesem Abschnitt wird dargelegt, wie diese Informationen für die im letzten Abschnitt genannten Abhängigkeiten beschrieben werden können und es wird gezeigt, wie überprüft werden kann, ob eine Prozessvorlage unter diesen Abhängigkeiten gültig ist.

4.2.2 Externe Datenabhängigkeiten

Datenabhängigkeiten in VAS entstehen dadurch, dass Anwendungsfunktionen Datensätze oder einzelne Felder eines Datensatzes in der Datenbank speichern, auf die andere Anwendungsfunktionen dann zugreifen. Außerdem kommt es vor, dass Anwendungsfunktionen auf Datensätze zugreifen, die bereits vor Prozessstart in der Datenbank gespeichert sind. Bei manchen dieser Datensätze kann von ihrer Existenz ausgegangen werden, weil es sich dabei zum Beispiel um Daten zur Systemkonfiguration handelt (Im Prinzip kann die Existenz keines Datensatzes mit völliger Sicherheit angenommen werden. Jedoch reicht es bei diesen Datensätzen aus, ihr Fehlen wie andere unerwartete Fehler zu behandeln). Die Existenz anderer Datensätze ist dagegen nicht sicher. Bei diesen Datensätzen gibt es aber Anwendungsfunktionen, nach deren Aufruf die Existenz garantiert werden kann. Ein Beispiel: Ein Lkw, der ins Werk fährt, muss sich zuerst identifizieren. Dazu kann er z.B. Benutzername und Passwort angeben, oder es wird die Nummer seiner Identifikationskarte gelesen. Dann wird anhand von Benutzername und Passwort bzw. anhand der Identifikationsnummer untersucht, ob ein entsprechender Fahrzeug-Datensatz im System vorhanden ist. Je nach Ausgang dieser Überprüfung, kann ab sofort davon ausgegangen werden, dass ein Datensatz für den Lkw existiert bzw. es ist sicher, dass kein solcher Datensatz existiert. Ebenso wird beim Eingeben einer Liefernummer durch einen Lkw-Fahrer überprüft, ob eine entsprechende Lieferung im System existiert. Existiert keine solche Lieferung, muss der Fahrer eine andere Nummer eingeben. Hat er die Nummer einer existierenden Lieferung eingegeben, kann im Prozess fortgeschritten werden und nachfolgend aufgerufene Anwendungsfunktionen können davon ausgehen, dass die Lieferung existiert. Andererseits wird nie überprüft, ob die Daten vorhanden sind, die ein Silo einer Ladestraße zuordnen. Diese werden einmal während der Inbetriebnahme von VAS angelegt und, wenn überhaupt, nur von einem Administrator verändert. Deshalb kann davon ausgegangen werden, dass sie existieren. Die Abhängigkeit einer Anwendungsfunktion von der Existenz solcher Daten wird auch nicht dem Prozessmodellierungswerkzeug bekannt gemacht, da sie nicht überprüft werden muss.

Externe Datenabhängigkeiten können betrachtet werden wie explizit modellierte Datenflüsse. In beiden Fällen sind Anwendungsfunktionen nur dann fehlerfrei ausführbar, wenn alle benötigten Daten vorhanden sind, sprich die benötigten Eingabeparameter versorgt sind. Deshalb können ganz ähnliche Algorithmen zur Überprüfung der Versorgung aller benötigten Eingabeparameter verwendet werden wie bei der Überprüfung des explizit modellierten Datenflusses. Allerdings reicht dies nicht aus, um

externe Datenabhängigkeiten, wie sie in VAS auftreten, komplett zu beschreiben. Ein Problem stellt dar, dass zur Modellierzeit meist nicht genau festgelegt werden kann, an welchem Ort eine Anwendungsfunktion einen Datensatz speichert. Zwar sind die Datenbank und die Tabelle, in der der Datensatz gespeichert wird, bekannt, aber ohne den Primärschlüsselwert des Datensatzes kann dieser nicht wieder sicher gefunden werden. Deshalb wird dieser auf einem anderen Weg übergeben, in der Regel als normale Prozessvariable. Dies entspricht der Übergabe eines Zeigers auf den Datensatz. Um Datenabhängigkeiten dann überprüfen zu können, muss die Beschreibung von Anwendungsfunktionen so erweitert werden, dass für jede Abhängigkeit von Daten in einer Datenbank angegeben wird, welcher Eingabe- oder Ausgabeparameter als Referenz auf den Datensatz verwendet wird. Bei der Überprüfung muss diese Information dann zusätzlich mit ausgewertet werden, um sicherstellen zu können, dass der Datensatz, den eine Anwendungsfunktion liest, auch der gleiche ist, den eine andere zuvor geschrieben hat. Die Überprüfung wird zusätzlich dadurch verkompliziert, dass der Inhalt einer Prozessvariable, die als Referenz auf einen externen Datensatz verwendet wird, überschrieben werden kann, sodass die Referenz auf einen anderen Datensatz zeigt. Dies wird in Abbildung 4-1 dargestellt.

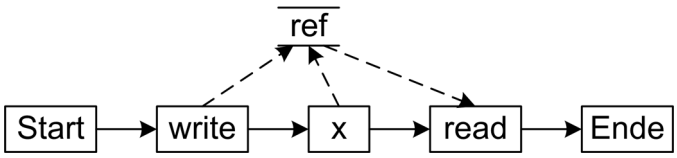

Abbildung 4-1: Überschreiben einer Referenz

Zuerst wird die Aktivität „write“ ausgeführt, die einen externen Datensatz schreibt und eine Referenz darauf in der Prozessvariablen „ref“ ablegt. Anschließend wird die Aktivität „x“ ausgeführt, die nicht auf externe Daten zugreift, aber einen Wert in der Prozessvariablen „ref“ ablegt. Dann kann bei Ausführung der Aktivität „read“ nicht mehr davon ausgegangen werden, dass der durch „ref“ referenzierte Datensatz existiert, denn „ref“ kann auf irgendeinen beliebigen, insbesondere auch einen nicht existierenden Datensatz zeigen.

Ein weiterer, von dem Verfahren nicht berücksichtigter Aspekt ist, dass Anwendungsfunktionen Datensätze auch löschen können. Dies erschwert die Überprüfung erheblich. Die Garantie, dass ein Datensatz geschrieben wurde, bevor er gelesen wird, reicht nicht aus, zusätzlich muss garantiert werden, dass der Datensatz nicht in der Zwischenzeit wieder gelöscht wird. Dazu müssen auch zu der schreibenden und der lesenden Aktivität parallele Zweige überprüft werden. Dies zeigt Abbildung 4-2. Da unter anderem die Ausführungsreihenfolge „write – delete – read“ möglich ist, kann nicht garantiert werden, dass die von der Aktivität „read“ benötigten Daten zur Verfügung stehen, wenn sie ausgeführt wird. Noch komplizierter wird die Überprüfung, da auch bedacht werden muss, dass zwei Prozessvariablen auf den gleichen Datensatz zeigen können. Wird der Datensatz gelöscht, auf den eine

Prozessvariable zeigt, kann anschließend nicht mehr garantiert werden, dass ein Datensatz existiert, auf den eine andere Prozessvariable zeigt.

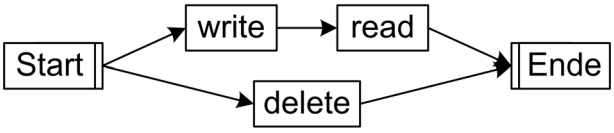

Abbildung 4-2: Löschende Aktivität in einem parallelen Zweig zu schreibender und lesender Aktivität. Mögliche Ausführungsreihenfolgen: write – read – delete, write – delete – read, delete – write – read.

Die in [Reic00] vorgestellten Algorithmen zur Überprüfung der Versorgung von Eingabeparametern überprüfen, ob garantiert werden kann, dass eine Prozessvariable beschrieben wurde, bevor sie gelesen wird. Diese Überprüfung reicht bei externen Daten nicht aus. Um die Versorgung einer Aktivität mit allen externen benötigten Daten garantieren zu können, muss sichergestellt werden, dass ein externes Datum geschrieben wurde, bevor es gelesen wird. Zusätzlich muss sichergestellt werden, dass zwischen der letzten schreibenden Aktivität und der lesenden Aktivität keine andere Aktivität aufgerufen werden kann, die das externe Datum löscht oder die als Referenz auf das externe Datum dienende Prozessvariable beschreibt.

Viele Prozesse enthalten alternative Verzweigungen, die anhand der Existenz bestimmter Daten entschieden werden. So kann zum Beispiel bei der Einfahrt eines Lkw ins Werk überprüft werden, ob alle Daten des Lkw bereits im System vorhanden sind und davon abhängig eine Bildschirmmaske geöffnet werden, auf der die fehlenden Daten eingegeben werden können. Die Entscheidungsgrundlage für die alternative Verzweigung wird von einer Anwendungsfunktion in einem Ausgabeparameter geliefert, der z.B. booleschen Wertebereich hat und so viel bedeutet wie „alle benötigten Daten vorhanden“ bzw. „nicht alle benötigten Daten vorhanden“. Auch wenn die Anwendungsfunktion keine Daten schreibt, so garantiert sie doch die Existenz der Daten, wenn ihr Ausgabeparameter den richtigen Wert hat. Dies lässt sich für verfeinerte Überprüfungen ausnutzen.

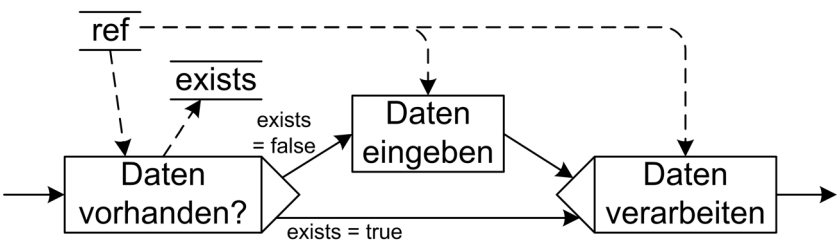

Abbildung 4-3: Ausschnitt aus einem Prozess, in dem die Existenz von bestimmten externen Daten überprüft wird, die Daten bei Bedarf angelegt und anschließend verarbeitet werden.

Abbildung 4-3 stellt ein Beispiel dazu dar. Darin wird zunächst überprüft, ob bestimmte externe Daten vorhanden sind. Bei Bedarf werden die Daten angelegt und anschließend werden sie auf irgendeine Art verarbeitet. Dabei schreibt die Aktivität „Daten eingeben“ den Datensatz, auf den „ref“ zeigt und „Daten verarbeiten“ liest den Datensatz. „Daten vorhanden?“ prüft, ob der Datensatz vorhanden ist, und setzt die Variable „exists“ auf „true“, falls der Datensatz vorhanden ist und andernfalls auf „false“. Ein Betrachter erkennt relativ schnell, dass für die Aktivität „Daten verarbeiten“ garantiert werden kann, dass der Datensatz bei ihrer Ausführung existiert. Um dies auch durch eine automatische Überprüfung feststellen zu können, muss die Beschreibung der Anwendungsfunktionen erweitert werden. Es muss möglich sein anzugeben, dass ein Datensatz existiert, wenn ein Ausgabeparameter einen bestimmten Wert hat. Der Prüfalgorithmus muss diese Informationen dann auswerten und mit den Bedingungen vergleichen, die alternative Verzweigungen entscheiden.

Es reicht nicht aus, die Bedingungen, unter denen die Existenz von Daten garantiert werden kann mit den Bedingungen an alternativen Verzweigungen auf Gleichheit zu untersuchen, wie Abbildung 4-4 zeigt. Darin wird durch die Aktivität „Daten verändern?“ zusätzlich ein Benutzer gefragt, ob er Daten verändern will, auch wenn sie bereits existieren. Die Aktivität „Daten eingeben“ wird dann ausgeführt, wenn keine Daten vorhanden sind oder der Benutzer angegeben hat, dass sie in jedem Fall verändert werden sollen. Die Bedingung für die Existenz der Daten, definiert von der Anwendungsfunktion der Aktivität „Daten vorhanden?“, lautet „exists = true“. Diese unterscheidet sich von den Bedingungen der alternativen Verzweigung, aber so, dass trotzdem in jedem Fall garantiert ist, dass bei Ausführung der Aktivität „Daten verarbeiten“ die Daten vorhanden sind.

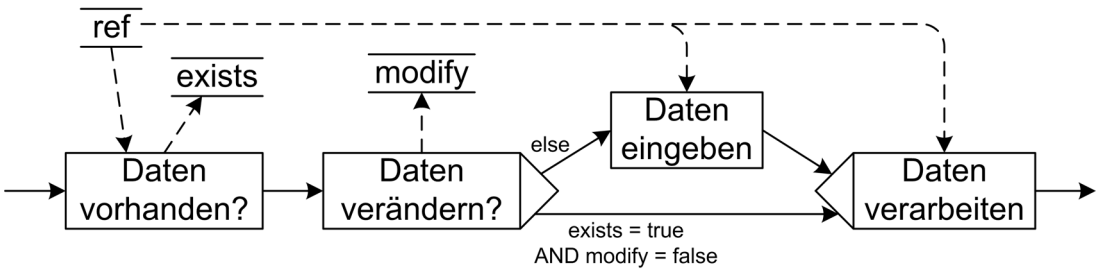

Abbildung 4-4: Ausschnitt aus einem Prozess, bei dem die Bedingung für die Existenz von Daten allgemeiner ist als die Bedingung an einer Verzweigung.

Die Überprüfungen von Datenabhängigkeiten lassen sich noch weiter verfeinern. Ähnlich wie bei Anwendungsfunktionen, die unter bestimmten Bedingungen die Existenz bestimmter externer Daten garantieren, können Bedingungen von alternativen Zweigen analysiert werden, in denen schreibende (oder löschende) Anwendungsfunktionen aufgerufen werden (siehe Abbildung 4-5). Zur Modellierzeit kann nicht garantiert werden, dass die Aktivität „b“ sicher die Daten lesen kann, die von der Aktivität „write“ geschrieben wurden, weil nicht garantiert werden kann, dass die Aktivität „write“ aufgerufen

wird. Für alle Aktivitäten im oberen Zweig der Alternativen Verzweigung nach der Aktivität „b“, d.h. Aktivitäten im Zweig mit der Bedingung „condition AND otherCondition“ kann dies aber garantiert werden.

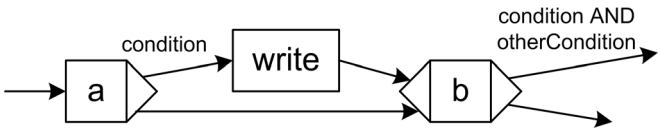

Abbildung 4-5: Ausschnitt aus einem Prozess, in dem nur unter bestimmten Bedingungen bestimmte externe Daten geschrieben werden (Aktivität „write“)

Es sind bestimmt noch weitere Fälle denkbar, die durch die bisher vorgestellten Überprüfungen nicht abgedeckt sind, obwohl sie theoretisch automatisch überprüft werden könnten. Allerdings wird der Aufwand für die Überprüfungen mit deren Genauigkeit steigen. Allein schon die Betrachtung von Bedingungen an alternativen Verzweigungen kann sehr komplex werden, wenn für deren Formulierung eine einigermaßen ausdrucksstarke Sprache verwendet wird. Zudem sind Konstruktionen wie die in Abbildung 4-5 dargestellte relativ selten, was den Nutzen solcher Überprüfungen verringert.

Zur Beschreibung von Anwendungsfunktionen unter der Berücksichtigung von Datenabhängigkeiten wird die in Tabelle 4-1 dargestellte Syntax eingeführt. Zur Darstellung wird eine BNF-Notation unter Berücksichtigung der folgenden Konventionen verwendet:

- Schlüsselworte sind groß und in einer `SCHRIFT MIT FESTER BREITE` geschrieben. Einzelne Zeichen sind mit einfachen Anführungszeichen markiert (z.B. ’;’).
- Syntaxelemente, die an anderer Stelle genauer definiert werden, sind klein und in *kursivschrift* geschrieben.
- Vertikale Striche (|) trennen Alternativen voneinander.
- Runde Klammern (()) dienen zur Strukturierung.
- Syntaxelemente in eckigen Klammern ([]) sind optional.
- Syntaxelemente in geschweiften Klammern ({}) können beliebig oft (null Mal oder öfter) wiederholt werden.
- Syntaxelemente, die für Bezeichner oder Werte stehen, sind nicht formal spezifiziert, sondern in Worten erklärt. Diese Erklärungen sind in serifenloser Schrift geschrieben.

function-definition	::=	`DEFINE FUNCTION` *function-name* '(' [*parameter-list*] ')'
		[*precondition-def*]
		[*postcondition-def*]
		`END`';'
parameter-list	::=	*parameter-def* {',' *parameter-def*}
parameter-de	::=	(`IN | OUT`) *type parameter*
precondition-def	::=	`PRECONDITION` *condition-term-list*
condition-term-list	::=	(*condition-term* {*condition-term*} | ';')
condition-term	::=	`TABLE` *tablename* `RECORD` *parameter*
		([`NOT`] `EXISTS` | `FIELDS` *feld* {',' *feld*}) ';'
postcondition-def	::=	`POSTCONDITION` (*condition-term-list* | *conditional-expr* | ';')
conditional-expr	::=	`CASE` '(' *expr* ')' *condition-term-list*
		{`CASE` '(' *expr* ')' *condition-term-list}*
		[`ELSE` *condition-term-list*]
function-name	::=	Name der zu definierenden Funktion
type	::=	Typ eines Parameters, z.B. Integer oder String oder ein komplexer Typ
parameter	::=	Name eines Parameters. In *condition-term* als Referenz auf einen in *parameter-list* definierten Parameter.
tablename	::=	Name einer Tabelle, auf die die Anwendungsfunktion zugreift.
feld	::=	Name eines Felds in der Tabelle.
expr	::=	prädikativer Ausdruck, wie er in der verwendeten Prozessbeschreibungssprache bei bedingten Verzweigungen verwendet wird.

Tabelle 4-1: Syntax zur Definition von Datenabhängigkeiten

Für die Bedingungen (*expr*) in Anwendungsfunktions-Definitionen soll die gleiche Sprache verwendet werden wie für die Beschreibung der Bedingungen an Kanten bedingter Verzweigungen, mit dem einen Unterschied, dass hier in jedem Vergleich ein Ausgabeparameter der Funktion enthalten sein muss, während in der Prozessdefinitionen die Vergleiche auf Basis von Prozessvariablen formuliert werden. Dadurch ist gesichert, dass das Prozessmodellierungswerkzeug die Bedingungen analysieren und mit den Bedingungen an Verzweigungen vergleichen kann. Außerdem soll für die einzelnen `CASE`-Statements (in einer *conditional-expr*) gelten, dass sie in angegebener Reihenfolge nacheinander evaluiert werden. Trifft eines zu, werden alle nachfolgenden nicht mehr evaluiert. Dadurch kann ver-

mieden werden, dass Modellierungsfehler zu widersprüchlichen Aussagen führen. Ohne diese Vorkehrung könnte z.B. formuliert werden, dass für manche Werte der Ausgabeparameter sowohl die Existenz als auch die Nichtexistenz eines Datensatzes garantiert wird. Alternativ dazu müssten die einzelnen Bedingungen auf Disjunktheit überprüft werden, was bei dem gegebenen Freiheitsgrad in der Formulierung sehr aufwendig wäre.
Für Tabellen- und Feldnamen ist es im Prinzip nicht wichtig, dass sie gleich lauten wie die tatsächlich in der Datenbank verwendeten. Es muss lediglich sichergestellt werden, dass die verwendeten Namen eindeutig (im Sinne einer injektiven Abbildung) den Tabellen und Feldern in der Datenbank zugeordnet werden können. Für die Definition von Parametertypen wird hier der Einfachheit halber angenommen, dass sie an anderer Stelle vorgenommen werden kann, z.B. im Prozessmodell selbst. Ebenfalls der Einfachheit halber betrachtet die Definition der Anwendungsfunktionen nicht, wie diese an die Workflow-Engine angebunden werden können (z.B. durch Aufruf einer lokalen Funktion, durch entfernte Prozedur-Aufrufe, mittels Middleware oder als Web-Service).

4.2.3 Zustandsabhängigkeiten

Um die Einhaltung von Zustandsabhängigkeiten überprüfen zu können, müssen der Workflow-Engine bzw. dem Werkzeug zur Prozessmodellierung zum einen die Abhängigkeiten von Anwendungsfunktionen von einem bestimmten Zustand, zum anderen die Zustandsänderungen, die Anwendungsfunktionen durchführen, bekannt gemacht werden.

Dazu muss zunächst identifiziert werden, welche Anwendungsfunktionen zu einer Komponente gehören, denn diese (und nur diese) können vom Zustand der Komponente abhängig sein. Zusätzlich muss die Menge der möglichen Zustände der Komponente identifiziert werden. Dann muss für jede Anwendungsfunktion der Komponente definiert werden, in welchem Zustand sie ausgeführt werden darf und in welchen Zustand die Komponente bei Ausführung der Anwendungsfunktion überführt wird. Letztendlich muss für die Komponente ein komplettes Zustandsdiagramm verfügbar sein. Dazu erweitern wir die in Tabelle 4-1 eingeführte Syntax zur Beschreibung von Anwendungsfunktionen wie in Tabelle 4-2 dargestellt.

function-definition ::= `DEFINE FUNCTION` *function-name* '(' [*parameter-list*] ')'
[*precondition-def*]
[*postcondition-def*]
[*state-dependencies-def*]
`END`';'

state-dependencies-def ::= `MEMBER OF` *component transition-def* { *transition-def* }

transition-def ::= `TRANSITION` { `FROM` *state* } `TO` *state* ';'

component	::=	Name einer Komponente. Dient dazu, eine Gruppe von Anwendungsfunktionen zu definieren, die alle vom Zustand der gleichen Komponente abhängig sind.
state	::=	Ein bestimmter Zustand einer Komponente, repräsentiert durch eine beliebige Zeichenkette.

Tabelle 4-2: Erweiterung der Syntax zur Beschreibung von Anwendungsfunktionen um die Beschreibung von Zustandsabhängigkeiten

Für jede Anwendungsfunktion, die vom Zustand einer Komponente abhängig ist oder den Zustand einer Komponente verändert, muss dies angegeben werden. Mit der *transition-def* wird angegeben, in welchen Zustand eine Komponente bei der Ausführung der Anwendungsfunktion überführt wird, wenn sie zuvor in einem anderen Zustand ist. Ist eine Anwendungsfunktion nicht vom Zustand der Komponente abhängig, kann der Teil „`FROM` *state*“ weggelassen werden. In diesem Fall ist aber nur eine *transition-def* für die Anwendungsfunktion erlaubt. Ist eine Anwendungsfunktion von einem bestimmten Zustand einer Komponente abhängig, verändert diesen aber nicht, kann dies definiert werden, indem in der *transition-def* im `FROM`-Teil der gleiche Zustand angegeben wird wie im `TO`-Teil. Es ist nicht wichtig, dass ein Bezug zwischen den Bezeichnungen für Komponenten bzw. Zustände und den tatsächlich existierenden Komponenten und deren Zuständen existiert. Es ist lediglich wichtig, dass für Anwendungsfunktionen, die vom gleichen Zustand abhängig sind, die gleichen Bezeichnungen verwendet werden. In diesem Zusammenhang kann an einer zentralen Stelle eine für alle Anwendungsfunktionen gültige Definition von Komponenten eingeführt werden, an der definiert wird, welche Zustände diese Komponenten annehmen können. Dadurch kann verhindert werden, dass Zusammenhänge nicht erkannt werden, weil bei der Definition der Zustandsabhängigkeiten einer Anwendungsfunktion ein Schreibfehler passiert ist.

Die Überprüfung eines Prozessschemas auf Einhaltung von Zustandsabhängigkeiten kann ähnlich wie die Überprüfung von Datenabhängigkeiten erfolgen. Im Prinzip könnte jeder Zustand einer Komponente in einer Prozessvariablen gespeichert werden. Jede Anwendungsfunktion, die von dem Zustand abhängig ist, liest diesen Zustand und jede Anwendungsfunktion, die den Zustand verändert, schreibt die Prozessvariable entsprechend (um genau zu sein, bestehen die Schreib- und Lesekanten nicht zwischen Prozessvariablen und Anwendungsfunktionen, sondern zwischen Prozessvariablen und Aktivitäten). Die Zustandsabhängigkeiten werden genau dann eingehalten, wenn für jede Anwendungsfunktion garantiert werden kann, dass bei ihrem Aufruf die von ihr gelesene, den Zustand einer Komponente repräsentierende Prozessvariable den richtigen Wert hat. Das heißt, dass die Prozessvariable zuvor von einer anderen Aktivität mit dem richtigen Wert beschrieben werden musste.

4.2.4 Hardware-Abhängigkeiten

Um die Einhaltung von Hardware-Abhängigkeiten überprüfen zu können, müssen Informationen über die verwendete Hardware zur Verfügung stehen. Dies ist zur Modellierzeit prinzipiell nicht gegeben, oft werden Prozesse für mehrere Terminals oder mehrere Werke modelliert, in denen jeweils mit unterschiedlicher Hardware-Ausrüstung gerechnet werden muss. Allerdings kann überprüft werden, ob ein modellierter Prozess konsistent bezüglich der Hardware-Abhängigkeiten ist. Dazu muss für jede Funktion einer Software-Komponente, die mehrere Hardware-Komponenten steuern kann, definiert werden, mit welchen Hardware-Komponenten (genauer: Typen von Hardware-Komponenten) sie ausführbar ist. Auf dieser Basis kann überprüft werden, ob ein Prozess Funktionen benötigt, die nur von unterschiedlichen Hardware-Komponenten unterstützt werden. Ein solcher Prozess wäre nicht ausführbar.

Die Überprüfung auf Konsistenz eines Prozesses bezüglich Hardware-Abhängigkeiten kann dahingehend erweitert werden, dass einem Prozessmodellierer mitgeteilt wird, welche Hardware-Komponenten zur Ausführung eines Prozesses verfügbar sein müssen. Dazu muss auf Basis der Definitionen der verwendeten Funktionen einer Software-Komponente die Schnittmenge der Hardware-Komponenten ermittelt werden, mit denen diese Funktionen ausgeführt werden können. Diese Schnittmenge gibt für jede Software-Komponente die Hardware-Komponenten an, mit denen der Prozess ausführbar ist. Ist die Schnittmenge für eine Software-Komponente leer, ist der Prozess inkonsistent bezüglich der Hardware-Abhängigkeiten dieser Software-Komponente. Ein Beispiel: In einem Prozess werden die Funktionen s_1, s_2 und s_3 der Software-Komponente s und die Funktionen t_1, t_2 und t_3 der Software-Komponente t aufgerufen. Die Software-Komponente s kann die Hardware-Komponenten h_1 und h_2 ansteuern, während t die Hardware-Komponenten i_1 und i_2 ansteuern kann. Tabelle 4-3 zeigt, welche Funktionen mit welcher Hardware ausführbar sind.

Software-Komponente	s			t		
Funktion	s_1	s_2	s_3	t_1	t_2	t_3
Mögliche Hardware-Komponenten	h_1, h_2, h_3	h_1, h_2	h_1, h_2	i_1	i_2	i_1, i_2

Tabelle 4-3: Beispielhafte Hardware-Abhängigkeiten

Die Schnittmenge der Hardware-Komponenten für Software-Komponente s ist $\{h_1, h_2\}$, das heißt, der Prozess im Beispiel kann nur ausgeführt werden, wenn eine Hardware-Komponente vom Typ h_1 oder vom Typ h_2 vorhanden ist. Die Schnittmenge der Hardware-Komponenten für Software-Komponente t ist leer, das heißt, der Prozess im Beispiel ist nicht konsistent bezüglich der Hardware-Abhängigkeiten der Software-Komponente t und kann deshalb nicht ausgeführt werden.

Wie oben erwähnt, stehen zur Modellierzeit keine Informationen darüber zur Verfügung, welche Hardware-Komponenten zur Laufzeit einem Prozess zur Verfügung stehen. Werden die Vorlagen der Prozesse zweiter Ebene auf den Terminals selbst hinterlegt, steht dies aber zum Zeitpunkt der Ablage der Prozessdefinition fest. Somit kann zu diesem Zeitpunkt auch überprüft werden, ob die benötigte Hardware verfügbar ist. Dazu ist allerdings eine Beschreibung der vorhandenen Hardware eines Terminals notwendig, mit der die Beschreibung der Hardware-Abhängigkeiten der Anwendungsfunktionen abgeglichen werden kann. Somit kann auf fehlende Hardware bzw. fehlerhaft modellierte Prozesse reagiert werden, bevor Instanzen einer Prozessdefinition erzeugt und verwendet werden.

Werden sämtliche Hardware-Komponenten erfasst, die an allen Terminals eines Werks zur Verfügung stehen, kann jederzeit festgestellt werden, auf welchen Terminals eine beliebige Prozessdefinition zur Ausführung gebracht werden kann. Da sich die Hardware-Konfiguration eines Werks nur äußerst selten ändert, kann damit bereits zur Modellierzeit oder bei der Änderung einer Prozessvorlage festgestellt werden, auf welchen Terminals Prozesse dieser Vorlage zur Ausführung gebracht werden können.

function-definition	::=	DEFINE FUNCTION *function-name* '(' [*parameter-list*] ') '
		[*precondition-def*]
		[*postcondition-def*]
		[*state-dependencies-def*]
		[*hw-dependencies-def]*
		END';'
hw-dependencies-def	::=	REQUIRES HARDWARE *hardware-term* {OR *hardware-term*} ';'
hardware-term	::=	'(' *hardware-component* {AND *hardware-component*} ')'
hardware-component	::=	Name einer Hardware-Komponente

Tabelle 4-4: Syntaxerweiterung für die Definition von Hardware-Abhängigkeiten

Zur Definition der Hardware-Abhängigkeiten von Anwendungsfunktionen soll die in Tabelle 4-1 vorgestellte und in Tabelle 4-2 erweiterte Syntax wie in Tabelle 4-4 dargestellt noch einmal erweitert werden.

Für jede Anwendungsfunktion kann angegeben werden, welche Hardware-Komponenten sie zu ihrer Ausführung benötigt. Um nicht nur die Abhängigkeit von Anwendungsfunktionen von einer einzelnen Hardware-Komponente beschreiben zu können, ist es möglich, mit logischem UND und logischem ODER verknüpfte Ausdrücke zu verwenden.

Die in den letzten Abschnitten vorgestellte Syntax zur Beschreibung von Anwendungsfunktionen reicht nicht aus, um alle Aspekte einer Anwendungsfunktion zu definieren. Z.B. fehlen Beschreibungsmöglichkeiten für Fehler, die während der Ausführung der Anwendungsfunktion auftreten können und für transaktionale Eigenschaften. Diese wurden aber bewusst weggelassen, da sie nicht Thema dieser Arbeit sind.

5 Prozessvarianten

Dieses Kapitel geht der Frage nach, inwiefern sich die Prozesse in unterschiedlichen Unternehmen ähneln, und ob es nützlich sein kann, eventuell existierende Ähnlichkeiten und Beziehungen zwischen diesen Prozessen zu dokumentieren. Dazu werden zunächst in Abschnitt 5.1 die Gemeinsamkeiten und Unterschiede der Prozesse analysiert, basierend auf den Prozessen aus Unternehmen eins, zwei und drei, wie sie in Abschnitt 2.2 beschrieben sind. Darauf aufbauend wird in Abschnitt 5.2 überprüft, welchen Nutzen die Dokumentation der Zusammenhänge der Prozesse in unterschiedlichen Werken hat.

5.1 Unterschiede und Gemeinsamkeiten der Prozesse

In diesem Abschnitt werden Unterschiede und Gemeinsamkeiten zwischen den Prozessen erster und zweiter Ebene sowie den Anwendungsfunktionen unterschiedlicher Unternehmen analysiert. Dabei wird mit den am wenigsten detaillierten Prozessen erster Ebene begonnen, und nach den Prozessen zweiter Ebene werden die Anwendungsfunktionen, die sich auf der untersten Detailstufe befinden, betrachtet.

5.1.1 Unterschiede und Gemeinsamkeiten der Prozesse erster Ebene

Die Prozesse erster Ebene sind, abstrakt betrachtet, in allen Werken gleich: Sie bestehen aus den drei Teilen Einfahrtsabwicklung, Be- und Entladung sowie Ausfahrtsabwicklung. Betrachtet man sie aber genauer, sind durchaus Unterschiede festzustellen. So bestehen in Unternehmen eins die Prozesse erster Ebene für den Versand loser und verpackter Ware zwar aus den genannten drei Schritten, allerdings werden jeweils andere Schritte verwendet: Beim Versand loser Ware werden Ein- und Ausfahrt vom Lkw-Fahrer am Selbstbedienungsterminal ausgeführt, beim Versand verpackter Ware werden Ein- und Ausfahrt von einem Werksmitarbeiter abgewickelt. Noch stärker sind die Unterschiede bei Unternehmen drei: Da Lkw mehrere Produkte gleichzeitig anliefern und abholen können, unterscheiden sich die Prozesse erster Ebene von Lkw zu Lkw, jede Prozessinstanz hat ihr eigenes Prozessschema und unterscheidet sich von den anderen Prozessinstanzen. Betrachtet man auch die Zuordnung von Aktivitäten zu Terminals, unterscheiden sich die Prozesse erster Ebene innerhalb jedes Werks: Je nachdem, was für ein Produkt angeliefert wird bzw. abgeholt werden soll, muss an einer anderen Stelle im Werk entladen bzw. beladen werden, d.h. er wird an einem anderen Terminal ausgeführt. Im Allgemeinen lassen sich alle Prozesse erster Ebene wie in Abbildung 5-1 darstellen. Dies kann allerdings nicht als besondere Gemeinsamkeit betrachtet werden, beschreibt es doch nur den Sachverhalt, dass ein Lkw erst ins Werk kommen muss, dort eventuell entladen muss, dann beladen wird und abschließend das Werk wieder verlässt. Der Prozess ist so allgemein gehalten, dass sich alle denkbaren

Versand- und Anlieferprozesse darin widerspiegeln. Zumal auch in diesem Prozess nicht berücksichtig ist, dass die einzelnen Be- und Entladeschritte unterschiedlichen Terminals zugeordnet sein können.

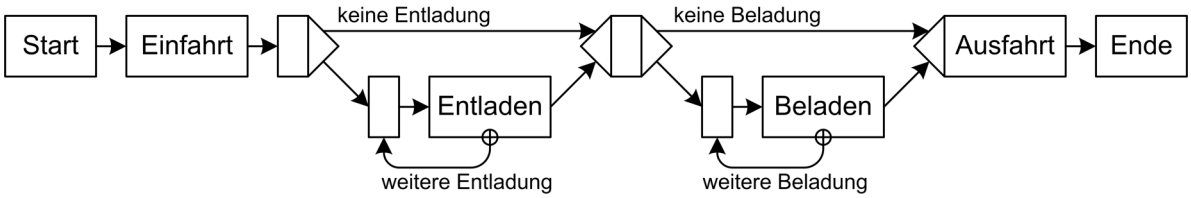

Abbildung 5-1: Abstrahierte Darstellung des Prozesses erster Ebene

5.1.2 Unterschiede und Gemeinsamkeiten der Prozesse zweiter Ebene

Bei den Prozessen zweiter Ebene, d.h. den Abläufen innerhalb eines Terminals, lassen sich ebenfalls Gemeinsamkeiten feststellen, wenn auch teils deutliche Unterschiede bestehen. Bei Prozessen zweiter Ebene treten keine Unterschiede zwischen einzelnen Instanzen eines Prozessschemas auf. Für Einfahrts- und Ausfahrtsprozess können Standard-Prozesse angegeben werden. Diese sind in Abbildung 5-2 dargestellt.

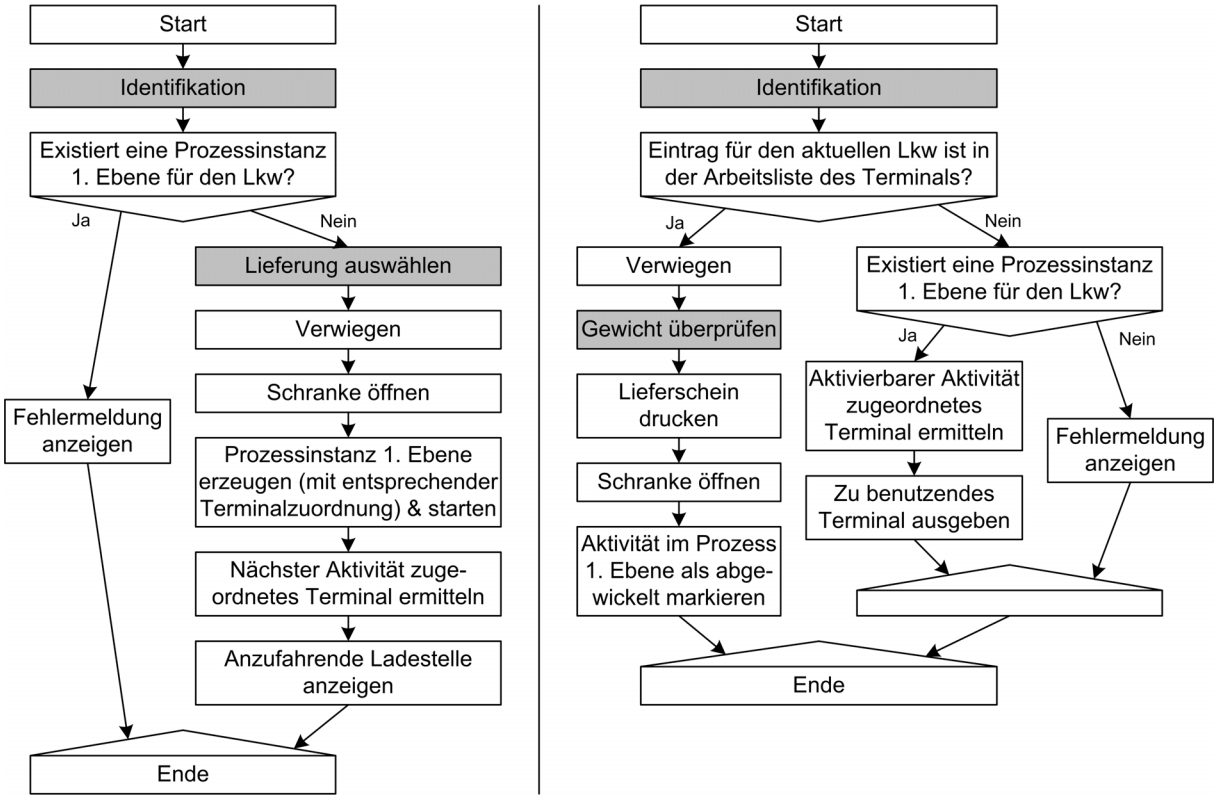

Abbildung 5-2: Standard-Prozesse der zweiten Ebene für Einfahrt (links) und Ausfahrt (rechts). Prozessschritte, hinter denen sich Subprozesse verbergen können, sind grau hinterlegt.

Die Standard-Prozesse enthalten die Prozessschritte und Subprozesse, die in (fast) allen Werken verwendet werden, und zwar in der Reihenfolge, in der sie meistens verwendet werden. Die Standard-Prozesse können als gemeinsame Basis der Prozesse in allen Werken betrachtet werden. Der Unterschied zwischen einem Standard-Prozess und einem Prozess in einem Unternehmen besteht darin, dass die Unternehmensprozesse weitere Schritte enthalten, selten auch einen Schritt des Standardprozesses weglassen oder zwei Schritte in anderer Reihenfolge ausführen. Nur der Ausfahrtsprozess in Unternehmen drei unterscheidet sich deutlich vom Standard-Prozess, in diesem Unternehmen werden Verwiegung, Gewichtsüberprüfung und Lieferscheindruck in der Regel bei der Verladung durchgeführt. Deshalb wird im Ausfahrtsprozess bei Unternehmen drei zunächst überprüft, ob Verwiegung und Lieferscheindruck notwendig sind, und diese Schritte werden nur bei positivem Ergebnis der Überprüfung ausgeführt. Die in der Abbildung grau hinterlegten Aktivitäten repräsentieren bei den meisten Unternehmen einen Subprozess, d.h., sie bestehen aus mehreren Aktivitäten. Betrachtet man diese, erkennt man, dass sich die Prozesse der unterschiedlichen Unternehmen nur auf der hier vorgestellten, relativ abstrakten Ebene so stark ähneln. Speziell die Identifikation ist von Unternehmen zu Unternehmen völlig unterschiedlich. So wird in Unternehmen eins bei der Einfahrt nur das Kfz-Kennzeichen des Fahrzeugs eingegeben, während in Unternehmen drei Fahrzeuge mit entfernt auslesbaren Transpondern identifiziert werden. In Unternehmen zwei werden nicht Fahrzeuge, sondern Fahrer identifiziert, und zwar entweder mit einer Identifikationskarte oder durch Eingabe von Benutzername und Passwort. Ähnlich große Unterschiede bestehen bei der Auswahl einer Lieferung. In einem Unternehmen besteht nur die Möglichkeit, eine Auftragsnummer einzugeben, in anderen Unternehmen ist eine Suchfunktion vorhanden oder es kann aus dem aktuellen Lkw zugeordneten Lieferungen ausgewählt werden. Auch die Gewichtsüberprüfung unterscheidet sich von Unternehmen zu Unternehmen, je nach gesetzlichen Anforderungen und Unternehmensregeln. Betrachtet man die Prozesse zweiter Ebene in dem Detaillierungsgrad, in dem sie als Vorlage für laufende Prozesse dienen können, überwiegen die Unterschiede zwischen den Prozessen einzelner Werke im Vergleich zu den Gemeinsamkeiten.

Für Be- und Entladung können keine sinnvollen Standard-Prozesse angegeben werden. Diese Prozesse werden von Werk zu Werk sehr unterschiedlich gehandhabt. In manchen Fällen werden sie komplett außerhalb des Systems durch Werksangestellte durchgeführt (z.B. Unternehmen eins), teils komplett automatisiert ohne Unterstützung durch Werksangehörige (z.B. Beladung mit loser Ware in Unternehmen drei). Alle Zwischenstufen zwischen diesen beiden Extremen sind ebenso denkbar.

5.1.3 Unterschiede und Gemeinsamkeiten der Anwendungsfunktionen

Im Vergleich zu den Prozessen erster und zweiter Ebene haben die Anwendungsfunktionen die größten Gemeinsamkeiten zwischen unterschiedlichen Werken. Dies hat vor allem zwei Gründe: Viele Module zur Anbindung von Hardware-Komponenten sind konfigurierbar, sodass sie unterschiedliche Hardware-Komponenten ansprechen können. So existiert ein Modul für die Kommunikation mit Waa-

gen aller Art, das je nach verwendeter Waage konfiguriert werden muss. Ähnliche Module existieren für die Ansteuerung von Druckern, Kartenlesegeräten und anderen Hardware-Komponenten. Außerdem ist das Datenbankschema bei allen Unternehmen nahezu gleich, meist gibt es nur Unterschiede in der Interpretation mancher Felder. Dadurch können Datenbankzugriffe weitgehend einheitlich gehalten werden, wenn nicht unterschiedliche Funktionalität benötigt wird.

Unterschiede zwischen Anwendungsfunktionen beschränken sich im Wesentlichen auf die Benutzerschnittstelle. Diese muss immer an die Wünsche von Kunden angepasst werden, was neben den Anpassungen in den Abläufen ein Hauptgrund für den Anpassungsaufwand von VAS an neue Werke darstellt. Weitere Unterschiede liegen meist nur darin, dass aus einer Menge von Anwendungsfunktionen, die in unterschiedlichen Werken einsetzbar sind, unterschiedliche Teilmengen verwendet werden. So wird z.B. in Werken ohne automatisierte Beladung die Funktionalität zur Ansteuerung von Beladeanlagen nicht benötigt, und in einem Werk, an dem Ein- und Ausfahrt nicht mit Schranke versehen sind, wird auch die Funktionalität zu deren Ansteuerung nicht benötigt.

5.1.4 Fazit

Die Prozesse erster Ebene unterscheiden sich bereits von Werksbesuch zu Werksbesuch eines Lkw in ein und demselben Werk, da für unterschiedliche Produkte unterschiedliche Beladeterminals verwendet werden müssen. Dementsprechend können Gemeinsamkeiten zwischen den Prozessen erster Ebene unterschiedlicher Unternehmen nur auf sehr abstrakter Ebene gefunden werden.

Für die Prozesse zweiter Ebene lassen sich zumindest für Ein- und Ausfahrt Gemeinsamkeiten feststellen. Es können Standard-Prozesse definiert werden, die das „Grundgerüst“ der Ein- und Ausfahrtsprozesse aller Werke bilden. Die Prozesse in den Werken weichen von diesen Standardprozessen vor allem darin ab, dass sie weitere Prozessschritte enthalten. Außerdem unterscheiden sich die Subprozesse der Standardprozesse (z.B. „Identifikation“) in unterschiedlichen Werken.

Die größte Gemeinsamkeit zwischen den Prozessen unterschiedlicher Unternehmen besteht auf Ebene der Anwendungsfunktionen, von denen viele unverändert in mehreren Unternehmen verwendet werden können.

Eventuell könnten weitere Gemeinsamkeiten zwischen Prozessen unterschiedlicher Werke gefunden werden, wenn diese in Gruppen eingeteilt werden. Eventuell könnten zwischen Werken in einem Land (wegen gleicher gesetzlicher Bestimmungen), einer Branche oder vielleicht auch zwischen Werken mit ähnlichem Automatisierungsgrad Gemeinsamkeiten gefunden werden. Um dies zu untersuchen, reichen allerdings die vorhandenen Unterlagen nicht aus, die nur für die Abläufe in drei Unternehmen verfügbar sind. Jedoch lässt sich daraus erkennen, dass die Abläufe in unterschiedlichen Werken eines Unternehmens sehr ähnlich sind. In Unternehmen eins und drei sind die Prozesse so ähnlich, dass momentan für alle Werke der Unternehmen die identische Software eingesetzt wird. In Unternehmen zwei sind die Prozesse ebenfalls sehr ähnlich. Allerdings läuft die Beladung in manchen Wer-

ken des Unternehmens automatisiert ab, in anderen wird sie von Werksmitarbeitern durchgeführt. Dadurch entsteht auch ein kleiner Unterschied zwischen den Einfahrtsprozessen in Werken mit und ohne automatisierte Beladung.

5.2 Erfassung der Zusammenhänge

In diesem Abschnitt wird diskutiert, inwiefern es sinnvoll ist, Abhängigkeiten zwischen Prozessdefinitionen zu dokumentieren oder gar Vererbungskonzepte zwischen verschiedenen Prozessdefinitionen einzuführen, ähnlich wie sie aus objektorientierten Programmiersprachen als Beziehung zwischen Klassen bekannt sind.

Wartung und Weiterentwicklung von Prozessen und Anwendungsfunktionen werden erleichtert, wenn Informationen über die in den Werken eines Unternehmens eingesetzten Prozesse und Anwendungsfunktionen ebenso wie die vorhandene Hardware verfügbar sind. Dies gilt, wenn ein Prozess oder eine Anwendungsfunktion für ein Unternehmen oder ein Werk angepasst werden soll, aber in weiteren Unternehmen bzw. Werken eingesetzt ist, in denen die Anpassung nicht gewünscht ist oder sogar zu Fehlfunktionen führen würde. Bei Prozessdefinitionen kann dies kaum vorkommen, da sich die Prozesse unterschiedlicher Unternehmen in der Regel unterscheiden, sodass jedes Unternehmen seine eigenen Prozessdefinitionen hat. Anders ist dies allerdings bei Anwendungsfunktionen, von denen viele in mehreren Prozessen und mehreren Unternehmen eingesetzt werden. Um zu überblicken, auf welche Prozesse, Werke und Unternehmen eine Änderung an einer Anwendungsfunktion Auswirkungen hat, sollte dokumentiert sein, in welchen Prozessen und Unternehmen die Anwendungsfunktion verwendet wird. Letztendlich sollte für jedes Werk jedes Unternehmens eine vollständige Dokumentation über die auf jedem einzelnen Terminal verwendeten Prozesse und Anwendungsfunktionen vorhanden sein. Daraus kann auch leicht die Information generiert werden, in welchen Prozessen und welchen Werken eine Anwendungsfunktion oder eine Prozessdefinition eingesetzt wird. Im Gegensatz zu dieser Dokumentation, die nur der Unterstützung von Entwicklern dient, dient die Dokumentation von Prozessvarianten der automatischen Unterstützung von Änderungen an Prozessen.

Werden in mehreren Werken eines Unternehmens sehr ähnliche Prozesse verwendet, ist es sinnvoll, diese Prozesse nicht unabhängig voneinander zu definieren, sondern die gemeinsamen Teile einmal festzulegen und dann nur die Abweichungen zu definieren. Zum Beispiel gibt es in Unternehmen zwei Werke mit vollautomatisierter Beladung und Werke, in denen Lkw von Werksmitarbeitern beladen werden. Der Einfahrtsprozess in den beiden Werkstypen unterscheidet sich geringfügig, weil in Werken mit automatisierter Beladung bei der Einfahrt in der Datenbank gespeichert werden muss, mit welchem Produkt und in welcher Menge ein Lkw beladen werden soll. Dies ist in Werken, in denen Werksmitarbeiter beladen, nicht notwendig. Ansonsten sind die Einfahrtsprozesse in den beiden Werkstypen identisch. Dabei ist es sinnvoll, den Prozess für die Einfahrt in Werken mit automatisierter Beladung auf Basis des Prozesses der anderen Werke zu definieren. Dies verringert den Aufwand

für die Erstellung der Prozessdefinitionen und stellt gleichzeitig sicher, dass sich die Definitionen der beiden Prozesse nur an den gewünschten Punkten unterscheiden. Soll der Einfahrtsprozess in allen Werken von Unternehmen zwei an neue gesetzliche Bestimmungen oder Unternehmensrichtlinien angepasst werden, muss nur die Prozessdefinition für Werke, in denen Werksmitarbeiter beladen, geändert werden. Diese Änderungen gelten auch automatisch für die Definition des Prozesses für Werke mit automatisierter Beladung. Um solche automatischen Ableitungen unterstützen zu können, müssen im System die Beziehungen zwischen den Prozessdefinitionen gespeichert sein.

Eine Möglichkeit der Dokumentation der Beziehungen stellt das in [Buss99] vorgestellte Konzept der Vererbung zwischen Prozessdefinitionen dar, das dem Konzept der Vererbung in objektorientierten Programmiersprachen ähnelt. Dabei werden Prozessdefinitionen von einer Basisdefinition abgeleitet. Für die abgeleiteten Definitionen werden keine vollständigen Prozesse definiert, sondern nur die Abweichungen von der Basisdefinition. Dies bedeutet unter anderem, dass abgeleitete Prozessdefinitionen nur verstanden werden können, wenn auch die Definition des Basisprozesses berücksichtigt wird.

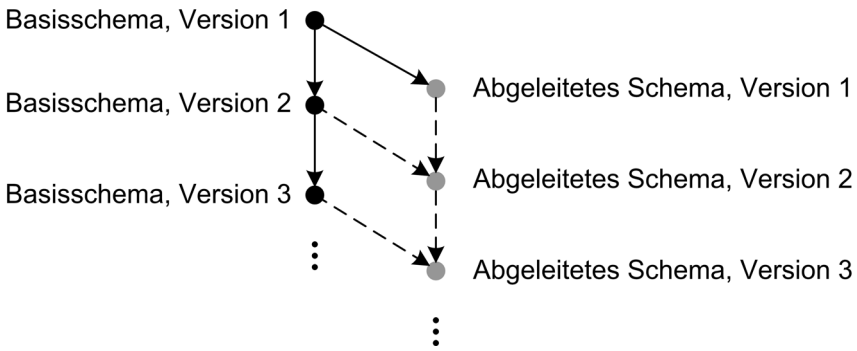

Abbildung 5-3: Beziehungen zwischen einem Basisschema und einem daraus abgeleiteten Prozessschema in mehreren Versionen. Die von einem Prozessmodellierer durchgeführten Änderungen sind durch durchgezogene Pfeile dargestellt, die automatische Kombination der beiden Schemata durch gestrichelte Kanten.

Eine andere Möglichkeit sind die im ADEPT-Projekt entwickelten Konzepte zur Propagation von Änderungen an einem Prozessschema auf laufende, eventuell auch veränderte Prozessinstanzen [RRD04]. Dazu müssen die Ableitungsbeziehungen und Versionsinformationen gespeichert werden, da bei der Propagation der Änderungen eines Prozessschemas auf ein anderes Prozessschema zusätzlich die gemeinsame Basis der neuen Version und des abgeleiteten Schemas benötigt wird. Abbildung 5-3 stellt dar, wie die Änderungen an einem Basisschema auf ein davon abgeleitetes Schema propagiert werden. Von einem Basisschema leitet ein Prozessmodellierer ein anderes Schema (grau dargestellt) ab. Irgendwann wird eine neue Version des Basisschemas erstellt. Aus dieser neuen Version und dem bisherigen abgeleiteten Schema wird automatisch eine neue Version des abgeleiteten Schemas erzeugt. Dazu werden die zweite Version des Basisschemas und die erste Version des abgeleiteten Schemas mit der ersten Version des Basisschemas verglichen, um herauszufinden, welche Änderun-

gen jeweils durchgeführt wurden. Daraus wird dann die zweite Version des abgeleiteten Schemas erstellt. Ein Vorteil dieser Lösung gegenüber den in [Buss99] vorgestellten Konzepten ist, dass jedes Prozessschema allein einen vollständigen Prozess definiert, ohne auf die Definitionen eines anderen Schemas zurückzugreifen. Damit braucht einerseits eine Workflow-Engine immer nur eine Prozessdefinition, andererseits ist dies auch für Prozessmodellierer leichter verständlich, wie wenn eine Basis-Prozessdefinition und darauf aufbauend die Definition von Änderungen an der Definition betrachtet werden müssen, um das Schema eines abgeleiteten Prozesses zu verstehen.

6 Zusammenfassung, Diskussion

In dieser Arbeit wurde vorgestellt, wie Workflow-Engines in VAS eingesetzt werden können, um die Anpassbarkeit und Wartbarkeit des Systems zu erhöhen. Dazu wurde das System, wie es in drei Unternehmen eingesetzt ist analysiert. Auf Basis dieser Analyse wurde eine Lösung für den Einsatz von Workflow-Engines entwickelt, die bedingt durch das besondere Einsatzgebiet von üblichen Einsatzszenarien für Workflow-Engines abweicht. Anschließend wurden einige Prozessbeschreibungssprachen auf ihre Eignung für die Definition von Prozessen in VAS analysiert. Um Prozessmodellierer durch möglichst viele automatische Überprüfungen zu unterstützen, wurden die Abhängigkeiten zwischen Anwendungsfunktionen analysiert und eine Methodik zu ihrer Beschreibung entworfen, auf deren Basis ihre Einhaltung automatisch überprüft werden kann. Insgesamt wurde gezeigt, wie sich der Aufwand für die Anpassung von VAS an neue Unternehmen bzw. Werke durch den Einsatz von Workflow-Engines reduzieren lässt. Dadurch wird es außerdem ermöglicht, dass nicht nur Programmierer Abläufe anpassen können, sondern auch z.B. entsprechend geschulte Mitarbeiter von Unternehmen, in denen VAS eingesetzt wird. Durch den Einsatz von Workflow-Technologie, die auch moderne Konzepte wie Ad-hoc-Änderungen von Prozessinstanzen und die Validierung von Prozessdefinitionen auch bezüglich Abhängigkeiten von Anwendungsfunktionen unterstützt, können die Anforderungen an Prozessmodellierer weiter gesenkt werden. Dadurch wird die Behandlung einiger Hardware-Ausfälle stark erleichtert. Es entsteht allerdings ein relativ großer Aufwand, wenn VAS auf die Steuerung durch Workflow-Engines umgestellt werden soll. Alle Funktionalität, die in einem Prozess verwendet werden soll, muss in Anwendungsfunktionen gekapselt werden. Zudem müssen auch einige Funktionen implementiert werden, die für die Anbindung der Workflow-Engine und insbesondere die Kommunikation zwischen Prozessen erster und zweiter Ebene benötigt werden. Außerdem müssen die von einer Workflow-Engine zusätzlich benötigten Ressourcen ebenso berücksichtigt werden wie die Tatsache, dass eine Workflow-Engine deutlich mehr Zeit zum Weiterschalten eines Prozesses braucht als bei einem entsprechenden, fest programmierten Programm benötigt würde. Dadurch können eventuell Performance-Probleme entstehen, insbesondere bei der Steuerung der Prozesse zweiter Ebene.

Im letzten Kapitel wurden die Prozesse in den unterschiedlichen Unternehmen eingehend miteinander verglichen und Gemeinsamkeiten und Unterschiede analysiert. Daraus wurde abgeleitet, dass sich die Dokumentation der Gemeinsamkeiten in einigen Fällen lohnen kann, um den Aufwand für die Wartung von Prozessen zu verringern.

Es gibt zahlreiche Arbeiten, die sich mit ähnlichen Problemen wie den hier behandelten beschäftigen. Einige dieser Arbeiten sollen im Folgenden diskutiert werden.

6.1 Komponenten-Technologie und Workflow-Management

Im AristaFlow-Projekt [AAD+04, Ari] wird die Idee verfolgt, die Komposition von Komponenten in Abläufen zu erleichtern bzw. sogar eine (teilweise) Automatisierung der Komposition zu ermöglichen. Dies soll auf Basis einer genauen Beschreibung der Komponenten einschließlich der Abhängigkeiten der einzelnen Operationen erreicht werden. Ziel des Projekts ist, die Entwicklung adaptiver, prozessorientierter Informationssysteme aus Anwendungs-Komponenten zu ermöglichen. In diesem Rahmen wurden Konzepte zur Beschreibung und Validierung von Abhängigkeiten zwischen den Operationen einer Komponente entwickelt.

[Krot03] beschäftigt sich mit Abhängigkeiten zwischen den Operationen einer Komponente in der Art, dass eine Operation zwingend ausgeführt werden muss, bevor eine andere aufgerufen werden kann bzw. nachdem eine andere aufgerufen wurde. Grund für diese Beziehungen sind oft Daten, die eine Operation innerhalb der Komponente, d.h. von außen nicht sichtbar, der anderen Operation zur Verfügung stellt. Die in der Arbeit betrachteten Abhängigkeiten entsprechen den in Abschnitt 4.2.2 eingeführten externen Datenabhängigkeiten. Allerdings unterscheiden sich die Beschreibungsmittel. In [Krot03] werden die Beziehungen zwischen Paaren von Operationen betrachtet, während in dieser Arbeit die Vor- und Nachbedingungen einzelner Operationen spezifiziert werden.

In [Göse05] wird eine Klassifikation von Abhängigkeiten vorgeschlagen. Außerdem werden Abhängigkeiten von *zustandsbehafteten Daten* genauer betrachtet. Dabei handelt es sich darum, dass manche Operationen vom Zustand der ihnen übergebenen Daten abhängig sind, z.B. müssen diese in einem bestimmten Format vorliegen. Der Zustand einer Komponente kann als Datum betrachtet werden, das den Operationen der Komponente als zusätzlicher Eingabeparameter übergeben wird. Damit können die in Abschnitt 4.2.3 behandelten Abhängigkeiten ebenfalls als Abhängigkeiten von zustandsbehafteten Daten betrachtet werden.

[Blas96] betrachtet allgemein Probleme, die gelöst werden müssen, wenn aus vorgefertigten Programmbausteinen (d.h. Komponenten) Anwendungen zusammengesetzt werden sollen. Unter anderem wird ein Ansatz vorgestellt, der dazu dienen soll, den Datenfluss in einem vormodellierten Prozess (halb-)automatisch definieren zu können. Dazu wird unter anderem vorgeschlagen, ein kontrolliertes Vokabular für die Beschreibung der Parameter von Anwendungsfunktionen zu verwenden.

Im Bereich von Web Services wurden Standards entwickelt, die sich mit der über die reine API-Beschreibung (Beschreibung der zur Verfügung stehenden Operationen mit ihren Ein- und Ausgabeparametern) hinausgehenden Spezifikation von Software-Komponenten befassen. Ziel ist die vollautomatische Verknüpfung von Web Services. Zentraler Begriff ist hierbei die Choreographie von Web Services. Diese beschreibt zulässige Nachrichtenabfolgen zwischen einem oder mehreren Partnern. Anstatt konkreter Teilnehmer werden nur abstrakte Rollen definiert. Die Beschreibung der Menge zulässiger Nachrichtenabfolgen wird Konversations- oder Koordinationsprotokoll (*coordination pro-*

tocol) genannt [ReSt04]. Die Choreographie stellt eine Art öffentliche Schnittstelle dar. Dazu definiert ein Web Service die unterstützten Konversationsprotokolle sowie die Rolle, die er darin einnimmt.

Ein Beispiel einer Spezifikationsmethode für die Choreographie von Web Services ist Spezifikation Web Service Choreography Interface (WSCI) [AAF+02], die auch in BPML (Business Process Modeling Language, [BPML02]) enthalten ist. Sie und auch BPML bieten die Möglichkeit, Aspekte wie Kontrollfluss, Datenfluss, das Verhalten im Fehlerfall und transaktionale Aspekte zu spezifizieren. Ähnliche Möglichkeiten bietet BPEL4WS, das nicht nur zur Definition von ausführbaren Prozessen verwendet werden kann, sondern auch zur Beschreibung so genannter Business Protocols, formaler Beschreibungen der Konversationsprotokolle miteinander interagierender Prozesse [ACD+03].

6.2 Analyse von Prozessbeschreibungssprachen

Van der Aalst und andere haben eine Reihe von Workflow Patterns [WfP] definiert, angelehnt an die Idee der Design Patterns aus der objektorientierten Programmierung. Dazu wurden zwanzig Patterns für den Kontrollfluss definiert [AHKB03, Kiep02] und viele Produkte, Standards und Spezifikationen auf die Unterstützung der Patterns hin untersucht [WfP], unter anderem wurde auch ADEPT betrachtet [AaHo03]. Außerdem wurden Patterns für Datenfluss [RHEA04] und die Verwaltung von Ressourcen [RHEA04b] definiert. Auch für diese Patterns wurden einige Produkte und Spezifikationen untersucht. Für Kontrollfluss-Patterns existiert eine Analyse aller in Abschnitt 4.1 betrachteten Prozessbeschreibungssprachen, BPEL4WS und XPDL wurden auch auf Datenfluss-Patterns hin untersucht. Die Workflow Patterns stellen eine Möglichkeit dar, die unterschiedlichen Workflow-Engines und Prozessbeschreibungssprachen miteinander zu vergleichen. Dabei wird allerdings nur die Ausdrucksmächtigkeit bzw. Unterstützung für möglichst viele Konstrukte betrachtet, Aspekte wie Verständlichkeit, Analysierbarkeit oder Visualisierbarkeit werden nicht betrachtet. Diese Aspekte sind aber ähnlich wichtig und können nicht vernachlässigt werden. In [AaHo03] wird darauf hingewiesen, dass eines der Kontrollfluß-Patterns („Implicit Termination“) Entwurfsfehler verdeckt, weil es die Erkennung von Deadlocks verhindert. Die Unterstützung dieses Patterns hat demzufolge sogar direkte Nachteile.

6.3 Sichten auf Prozesse

In vielen Fällen haben unterschiedliche Beteiligte unterschiedliche Sichten auf Prozesse. Auch im Umfeld von VAS kann dies beobachtet werden. So sind Prozesse aus Sicht der Unternehmensleitung an Lieferungen, aus Sicht eines Lkw-Fahrers und der Workflow-Engine an Werksbesuche geknüpft. Während eines Werksbesuchs eines Lkw können mehrere Lieferungen abgewickelt werden, es kann aber auch sein, dass eine Lieferung bei mehreren Besuchen eines Lkw behandelt wird (z.B. zunächst Waren abholen und beim nächsten Werksbesuch den nicht benötigten Anteil zurückbringen). Die Unternehmensleitung hat relativ geringes Interesse daran, wie viele Lkw wie oft das Werk besuchen, ihr Interesse gilt den Warenströmen, die durch einzelne Aufträge oder Lieferungen repräsentiert werden.

Um die Informationen zu einer Lieferung zu erhalten, müssen eventuell die Ausführungsprotokolle mehrerer Prozesse ausgewertet werden. Dies zeigt Abbildung 6-1. Ein Lkw besucht ein Werk zweimal, wobei insgesamt drei Lieferungen von drei unterschiedlichen Produkten abgearbeitet werden. Dass die Schritte „Beladen Produkt 1“ und „Entladen Produkt 1“ zur gleichen Lieferung gehören, lässt sich nur anhand des Datenkontexts, der die Liefernummer enthalten muss, erkennen. Ebenso ist es für den Lkw-Fahrer irrelevant, ob „Entladen Produkt 1“ zu einer Anlieferung oder einer Rücklieferung gehört, was aber aus Unternehmenssicht zwei völlig unterschiedliche Vorgänge sind.

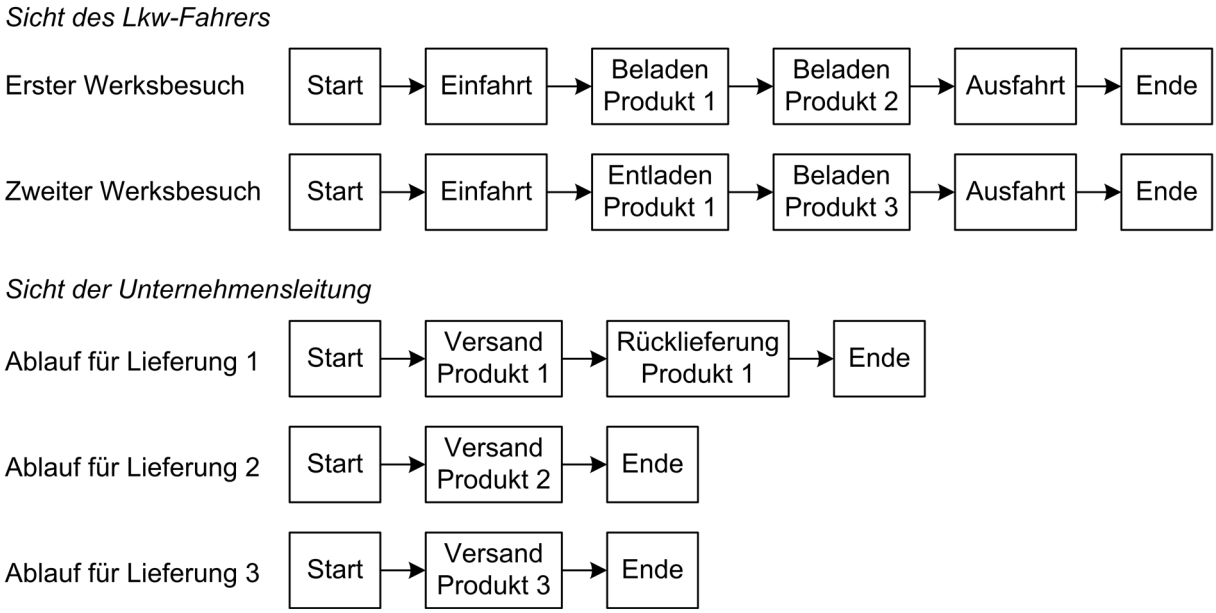

Abbildung 6-1: Versandprozesse aus Sicht des Lkw-Fahrers (entspricht den von einer Workflow-Engine verwalteten Prozessen) und der Unternehmensleitung.

Ein anderes Beispiel ist die Behandlung von Beladevorgängen in Unternehmen zwei. In manchen Fällen muss in zwei Schritten beladen werden, und dazwischen muss der Lkw bewegt werden, um seine Ladung gleichmäßig im Tank zu verteilen. Aus Sicht des Lkw-Fahrers ist dies ein Beladevorgang, der unterbrochen wird, aus Sicht der Workflow-Engine, die die Prozesse an dem Beladeterminal steuert, handelt es sich um zwei Vorgänge, jeweils von der Identifikation des Lkw-Fahrers bis zum Verlassen der Ladestraße. [BRB05] stellt Anforderungen an die Visualisierung von Prozessen zusammen. Die in diesem Beitrag skizzierte Komponente zur Visualisierung von Prozessinstanzen und Prozessschemata könnte dazu eingesetzt werden, die Sicht der Unternehmensleitung basierend auf den Ausführungsprotokollen der Workflow-Engines zur Verfügung zu stellen. In [Klot04] wird ein View-Konzept für Prozesse vorgestellt, das vom Konzept von Datenbank-Views inspiriert wurde. Dieses erlaubt es, nur Ausschnitte eines Prozesses darzustellen, Teile eines Prozesses zusammenzufassen, um die Komplexität der Ansicht zu verringern oder eine aggregierte Sicht mehrerer Prozessinstanzen zu erzeugen. Die-

se Mechanismen können nicht dazu verwendet werden, die an einzelnen Lieferungen orientierte Sicht aus den Daten der die Werksbesuche von Lkw steuernden Prozesse zu erzeugen. Allerdings könnten sie z.B. dazu verwendet werden, einem Werksleiter anzuzeigen, wie lange ein Lkw durchschnittlich im Werk ist oder wie häufig ein bestimmtes Terminal verwendet wird.

Anhang

Literaturverzeichnis

[AAD+04] Acker, H.; Atkinson, C.; Dadam, P.; Reichert, M.; Rinderle, S.: *Aspekte der komponentenorientierten Entwicklung adaptiver prozessorientierter Unternehmenssoftware.* Erschienen in: Turowski, K. (Hrsg.): Proceedings zur ersten Verbundtagung Architekturen, Komponenten, Anwendungen (AKA 2004), Augsburg, 2004, S. 7-24.

[AAF+02] Arkin, A.; Askary, S.; Fordin, S.; Jekeli, W.; Kawaguchi, K.; Orchard, D.; Pogliani, S.; Riemer, K.; Struble, S.; Takacsi-Nagy, P.; Trickovic, I.; Zimek, S.: *Web Service Choreography Interface (WSCI) 1.0.* 2002
http://www.w3.org/TR/2002/NOTE-wsci-20020808 (abgerufen: 05.10.2005)

[AaHo03] van der Aalst, W.; ter Hofstede, A.: *YAWL: Yet Another Workflow Language (Revised Version).* QUT Technical report, FIT-TR-2003-04, Queensland University of Technology, Brisbane, Australien, 2003.

[ACD+03] Andrews, T.; Curbera, F.; Dholakia, H.; Goland, Y.; Klein, J.; Leymann, F.; Liu, K.; Roller, D.; Smith, D.; Thatte, S.; Trickovic, I.; Weerawarana, S.: *Business Process Execution Language for Web Services, Version 1.1.* 2003.
http://www-106.ibm.com/developerworks/webservices/library/ws-bpel
http://dev2dev.bea.com/technologies/webservices/BPEL4WS.jsp
http://ifr.sap.com/bpel4ws (abgerufen: 11.10.2005)

[AHKB03] van der Aalst, W.; ter Hofstede, A.; Kiepuszewski, B.; Barros, A.: *Workflow Patterns.* Distributed and Parallel Databases, 14(3), Juli 2003, S. 5-51.

[Blas96] Blaser, R.: *Konfiguration verteilter Anwendungen aus vorgefertigten Programmbausteinen.* Diplomarbeit, Universität Ulm, 1996.

[BPML02] Arkin, A.: *Business Process Modeling Language.*
http://www.bpmi.org/bpml-spec.htm, 13.11.2002 (abgerufen: 06.10.2005)

[BRB05] Bobrik, R.; Reichert, M.; Bauer, T.: *Requirements for the Visualization of System-Spanning Business Processes.* Int. Workshop on Business Process Monitoring & Performance Management (BPMPM 2005), Kopenhagen, Dänemark, August 2005.

[Buss99] Bussler, C.: *Workflow Class Inheritance and Dynamic Workflow Class Binding.* Proceedings SABPM ‘99, Software Architectures for Business Process Management, Workshop auf der CAiSE ‘99, Heidelberg, 1999.

[Göse05] Göser, K.: *Plug&Play-Aspekte und Integration zustandsbehafteter Daten in Prozess-Management-Systeme.* Diplomarbeit, Universität Ulm, 2005.

[Jabl95] Jablonski, S.: *Workflow-Management-Systeme: Modellierung und Architektur.* International Thomson Publishing, Bonn, 1995.

[Kiep02] Kiepuszewski, B.: *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows.* Dissertation, Queensland University of Technology, Brisbane, Australien, 2002.

[Klot04] Klotz, A.: *View-Unterstützung in Prozess-Management-Systemen.* Diplomarbeit, Universität Ulm, 2004.

[Krot03] Kroter, D.: *Abbildung virtueller Kontroll- und Datenflüsse in Workflow Management Systemen.* Diplomarbeit, Universität Ulm, 2003.

[LeRo00] Leymann, F.; Roller, D.: *Production workflow: concepts and techniques.* Prentice Hall PTR, Upper Saddle River, New Jersey, USA, 2000.

[OMG02] Object Management Group: *Workflow Management Facility Specification, V1.2.* 2000.

[RBFD01] Reichert, M.; Bauer, T.; Fries, T.; Dadam, P.: *Realisierung flexibler, unternehmensweiter Workflow-Anwendungen mit ADEPT.* Erschienen in: P. Horster (Hrsg.): Proceedings Elektronische Geschäftsprozesse – Grundlagen, Sicherheitsaspekte, Realisierungen, Anwendungen, Klagenfurt, Österreich, September 2001, S. 217-228.

[ReDa98] Reichert, M.; Dadam, P.: *ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control.* Journal of Intelligent Information Systems, 10(2), 1998, S. 93-129.

[Reic00] Reichert, M.: *Dynamische Ablaufänderungen in Workflow-Management-Systemen.* Dissertation, Universität Ulm, 2000

[ReSt04] Reichert, M., Stoll, D.: *Komposition, Choreographie und Orchestrierung von Web Services - Ein Überblick.* EMISA FORUM, Mitteilungen der GI-FG "Entwicklungsmethoden für Informationssysteme u. deren Anwendung", Band 24, Heft 2, 2004, S. 21-32

[RHEA04] Russell, N.; ter Hofstede, A.; Edmond, D.; van der Aalst, W.: *Workflow Data Patterns.* QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, Australien, 2004

[RHEA04b] Russell, N.; ter Hofstede, A.; Edmond, D.; van der Aalst, W.: *Workflow Resource Patterns.* BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, Niederlande, 2004

[RRD04] Rinderle, S.; Reichert, M.; Dadam, P.: *Disjoint and Overlapping Process Changes: Challenges, Solutions, Applications.* Lecture Notes in Computer Science, Band 3290, Januar 2004, Seiten 101 – 120

[Sun03] Sun Microsystems: *Enterprise JavaBeans™ Specification, Version 2.1.* http://java.sun.com/products/ejb/docs.html (abgerufen: 18.10.2005)

[WfMC95] Workflow Management Coalition: *The Workflow Reference Model.* Document Number TC00-1003, 1995.

[WfMC98] Workflow Management Coalition: *Audit Data Specification.* Document Number WFMC-TC-1015, Version 1.1, 1998.

[WfMC98b] Workflow Management Coalition: *Workflow Management Application Programming Interface (Interface 2&3) Specification.* Document Number WFMC-TC-1009, Version 2.0, 1998.

[WfMC99] Workflow Management Coalition: *Terminology & Glossary.* Document Number WFMC-TC-1011, 1999.

[WfMC02] Workflow Management Coalition: *Workflow Process Definition Interface – XML Process Definition Language.* Document Number WFMC-TC-1025, Version 1.0, 2002.

Linksammlung

Alle aufgeführten Seiten wurden zuletzt am 17.10.2005 abgerufen.

[Ade] ADEPT Homepage, http://www.informatik.uni-ulm.de/dbis/index01.htm?01/forschung/projects/adept/adept.htm

[Ari] Projekt AristaFlow, http://www.aristaflow.de

[FMC] Fundamental Modeling Concepts, http://www.f-m-c.org

[Sha] Enhydra Shark Workflow-Engine, http://shark.objectweb.org

[WfP] Workflow Patterns, http://www.workflowpatterns.com

[WMW] IBM WebSphere MQ Workflow, http://www-306.ibm.com/software/integration/wmqwf

Dokumentation zum System: http://www-306.ibm.com/
software/integration/wmqwf/library/manuals/wmqwf34.html