Fachgebiet IT in Produktion und Logistik
Fakultät Maschinenbau, Technische Universität Dortmund

Methodik zur Erzeugung von Webformularen in betrieblichen Informationssystemen am Beispiel eines Bestellprozesses

Zur Erlangung des akademischen Grades eines
Bachelor of Science (Maschinenbau)
vorgelegt von

Cand.-Ing. Claudia Köster

Matrikelnummer: 139857
Studiengang: Bachelor of Science im Maschinenbau

1. Gutachter: Prof. Dr. Markus Rabe
2. Gutachterin: Dipl.-Inf. Anne-Antonia Scheidler

Ausgegeben am: 26.03.2013
Eingereicht am: 18.06.2013

Inhaltsverzeichnis

1 Einleitung

Ein Blick in das Curriculum des Maschinenbaus zeigt, dass dieser innerhalb der Ingenieurwissenschaften eine interdisziplinäre Stellung einnimmt. Es enthält neben seinen Kernelementen wie Produktions- und Maschinentechnik auch Anteile aus der Elektrotechnik, Informatik, Physik, Chemie und Verfahrenstechnik. Konkrete Schnittstellen zur Informatik findet man bei Simulationen an verschiedenen Systemen, Software für konstruierte Maschinen und Anlagen, sowie in der Unterstützung von Betriebsorganisation und -abläufen. Der letztgenannte Bereich beinhaltet das Feld der betrieblichen Informationssysteme, innerhalb dessen MaschinenbauingenieurInnen oft reine AnwenderInnen sind. Je nach persönlicher Ausrichtung der Ausbildung sind sie jedoch auch befähigt, an der Konzeption und Programmierung der darin enthaltenen Anwendungen mitzuwirken. In jedem Fall gehört der Umgang mit den Informationssystemen insbesondere im Arbeitsleben zur täglichen Routine dazu, wie man während der Praktika im Rahmen des ingenieurwissenschaftlichen Studiums lernen kann. Aus all diesen Berührungspunkten zwischen Maschinenbau und Informatik heraus motiviert sich die vorliegende Arbeit. Sie zeigt auf, dass IngenieurInnen unmittelbar in die Gestaltung ihrer informatischen Arbeitsumgebung mit einbezogen werden können und sollten.

Ausgehend von der Anwenderansicht wird in der Arbeit ein Teilaspekt eines bestehenden betrieblichen Informationssystems untersucht. Ziel der Untersuchung ist die Identifikation eines Schemas für den Aufbau und die inhaltlichen und formalen Zusammenhänge von Formularen, die innerhalb der Firmensoftware verwendet werden. Um die Handhabbarkeit des Themas zu gewährleisten, wird für die Untersuchung ein einziger Beispielprozess aus dem betrachteten System ausgewählt und beispielhaft untersucht. Auf dieser Grundlage wird die angestrebte Methodik entwickelt, vollständig beschrieben und zusätzlich mithilfe von XML an ausgewählten Beispielen in maschinenlesbarer Form dargestellt. Diese Kombination gewährleistet Nachvollziehbarkeit für Personen, die später auf die Ergebnisse zurückgreifen wollen und der Maschinenverarbeitbarkeit, um einen leichte Weiterverwendung zu ermöglichen.

In Kapitel 2 werden grundlegende Begriffe, die im Laufe der Arbeit immer wieder benutzt werden, geklärt und das Thema in Beziehung zu bisherigen Arbeiten im Sachgebiet gesetzt. Kapitel 3 stellt den ausgewählten Beispielprozess vor und analysiert den Einsatz von Formularen darin. Diese Analyse betrifft nur die inhaltlichen Bestandteile der Formulare und wird durch eine detaillierte Untersuchung des formalen Zusammenbaus eines Formulars ergänzt. Beides zusammen wird dann genutzt, um Funktions- und Aufbauregeln für die Methodik abzuleiten. Im Folgekapitel 4 wird auf Basis dieser Anforderungen gezeigt, welche Eingabeinformationen und Mechanismen benötigt werden, um ein funktionsfähiges Formular automatisch erzeugen zu lassen. Kapitel 5 zeigt Teile der Methodik an einigen Beispielen in XML beschrieben.

2 Aspekte der automatisierten Webformularentwicklung

Das folgende Kapitel führt den Leser in das in der Arbeit behandelte Thema ein. Die zentralen Begriffe der Arbeit werden zunächst erläutert und gegen die Literatur abgegrenzt. Darauf aufbauend wird das Thema als Ganzes eingeführt und in den Kontext anderer Arbeiten auf dem Gebiet eingebettet.

2.1 Zentrale Begriffe der Arbeit

Bereits 1975 wurde der Begriff des betrieblichen Informationssystems von Grochla verwendet. Gemäß seiner Definition ist es ein zweigeteiltes System, dessen erster Teil sich mit der informationellen Erfassung des Betriebszustandes und seiner relevanten Umweltsegmente befasst. Zweitens enthält es hauptsächlich Informationsverarbeitungsprozesse, die einer realistischen Planung der betrieblichen Zukunft dienen. Die von ihm im Laufe seines Buches als „automatisierte Datenverarbeitung“ bezeichnete Technik zur Unterstützung des Systems durch Computer und ähnliche Geräte taucht in der Definition nicht auf [Gro75, S.13, S.24]. Heutzutage wird ein betriebliches Informationssystem dagegen als „Gesamtheit der Betriebssysteme, Applikationen, Datenbanken und Kommunikation ermöglichenden Ressourcen sowie der sie operativ haltenden technischen Einrichtungen, organisatorischen Regeln und betrieblichen Einheiten“ bezeichnet [Fis11, S.432]. Hier werden hauptsächlich Elemente der elektronischen Datenverarbeitung (EDV) als Bausteine genannt, aus denen sich das Informationssystem im Ganzen zusammensetzt. Sie stehen im Vordergrund vor den nicht EDV-abhängigen Bestandteilen. Die Definition des betrieblichen Informationssystems nach [Fis11] wird aufgrund ihrer Aktualität in dieser Arbeit zugrunde gelegt.

Auf die Bedeutung des betrieblichen Informationssystems für das erfolgreiche Abwickeln der Vorgänge im Unternehmen weist Bullinger hin. Es sei für viele produzierende Unternehmen das Rückgrat ihrer betrieblichen Informationsverarbeitung [BF97, S.1]. Draheim spezifiziert bezüglich der beteiligten Software, sie sei kritisch für den Unternehmenserfolg, da sie extrem gut mit vielen Benutzern umgehen könne, hohe Lasten toleriere und eng mit den Geschäftsprozessen verbunden sei [Dra05, S.3]. Im Hinblick auf die Anwender kann man seine Aufzählung um zwei Punkte erweitern: Die verschiedenen Benutzer der Software sind nicht alle auf dem gleichen Bildungsstand bezüglich Entwicklung und Benutzung von Software und es gibt unter Umständen Prozessschritte in der Firma, die nicht vollautomatisch eingebunden sind, sondern manuell eingepflegt werden müssen. Der erste Punkt kann Auswirkungen auf die Gestaltung einer Software haben, der zweite Punkt sollte als mögliche Fehlerquelle beachtet werden.

Im Ingenieurwesen stößt man im Zusammenhang mit betrieblichen Informationssystemen oft auf den Begriff des „Enterprise Resource Planning“ (ERP) in Verbindung mit

einer einzelnen umfangreichen Software, die scheinbar alle in einem Betrieb vorhandenen Prozesse unterstützt. Hierbei handelt es sich um einen Sammelbegriff für Software, die zur Lösung von Aufgaben bei der Planung betrieblicher Ressourcen und Prozesse verwendet wird [Fis11, S.301]. Schuh schreibt von einer „Mengen- und Kapazitätsplanung in der Fertigung und Einbeziehung der vor- und nachgelagerten Bereiche wie Beschaffung und Vertrieb bis hin zur Darstellung und Unterstützung der kompletten Auftragsabwicklung“ [Sch12, S.4]. Beide Definitionen zeigen die deutliche Abgrenzung zum weiter gefassten Begriff des betrieblichen Informationssystems: Während das ERP der Unterstützung ausgewählter operativer Abläufe dient und oft als ein einzelnes Softwarepaket ausgeführt wird, soll das betriebliche Informationssystem Planungen und Entscheidungen innerhalb der Unternehmensausrichtung und -führung unterstützen und bezeichnet die Gesamtheit aller vorkommender Software. Außerdem beinhaltet es auch nicht an die EDV gebundene Lösungen. Die vorliegende Arbeit kann durch die Ausrichtung auf das betriebliche Informationssystem eine dementsprechend größere Zahl an Anwendungsgebieten erschließen, als dies bei der Beschränkung auf ERP-Software möglich wäre.

Viele Softwareelemente des betrieblichen Informationssystems besitzen ein in Formularen aufgebautes „Graphical User Interface“ (GUI). Dieser Begriff bezeichnet gemäß Stahlknecht die „bildhafte Darstellung der für den Benutzer erforderlichen Informationen“ [SH02, S.83]. Wird das GUI ausschließlich „zur Eingabe oder Ausgabe von Daten auf dem Bildschirm“ verwendet, spricht man von einer „Maske“. Die Bezeichnung „Formular“ wird hiermit gleich gesetzt. Die englische Begriffsentsprechung „Form“ birgt dagegen in der vorhandenen Definition eine Verwechslungsgefahr mit Fenstern der .NET-Programmierumgebung [Fis11]. Der Vorteil der Fomulare als Mittel zur Dateneinund -ausgabe besteht dabei insbesondere darin, dass auch Laien der fehlerarme Umgang mit den Daten ermöglicht wird, da die Manipulationsmöglichkeiten dem Anwendungsumfang und der zu erwartenden EDV-Kenntnisse der Nutzer gemäß beschränkt werden können [Cla06]. Wenn in der vorliegenden Arbeit von dem GUI oder der GUI-Ebene gesprochen wird, so bezieht sich das auf die Definition von [SH02]. Die Begriffe „Formular“ und „Maske“ werden im Laufe der Arbeit synonym verwendet, während der Begriff „Form“ vermieden wird.

Quelle der Ausgabe eines Formulares und Ziel der Eingabe ist oft eine Datenbank. Als Konstrukt, dessen Inhalte durch die GUI-Elemente repräsentiert und manipulierbar gemacht werden, ist sie der Kern vieler Informationssysteme, dessen Verlust oder Beschädigung das gesamte System unbrauchbar machen kann. Eine Datenbank bezeichnet gemäß Stahlknecht „mehrere Dateien, zwischen denen logische Abhängigkeiten bestehen“ [SH02, S.139]. Fischer ergänzt, dass „dieses Konstrukt auf Dauer und für den Mehrbenutzerzugriff angelegt“ ist [Fis11, S.208] während [Zeh98] und [Eng03] herausstellen, dass zu einer Datenbank immer ein Programm zur Speicherung und Abfrage der strukturierten Daten in Form eines Datenbankverwaltungssystems gehört. In der vorliegenden Arbeit wird bei der Verwendung des Begriffes von der Organisationsart einer „relationalen Datenbank“ ausgegangen, in der die enthaltenen „Daten als Datensätze in zweidimensionalen Tabellen organisiert“ sind [Fis11, S.210]. Im weiteren Verlauf der Arbeit bezieht sich der Begriff „Da-

tenbank“ auf relationale Datenbanken im Sinne einer Kombination der obigen Angaben von [SH02], [Fis11] und [Zeh98].

Den einzelnen Daten eines Datensatzes in einer Datenbank ist jeweils genau ein Datentyp zugeordnet. Datentyp bezeichnet hierbei die zulässige Wertemenge, deren Elemente ein Eintrag enthalten darf, sowie die auf ihn zulässig anwendbaren Operanden [SH02, ABC+98]. Der Datenaustausch werden zwischen den Formularen und der Datenbank findet mithilfe der sogenannten „Prozessebene“ statt. Ein Prozess bezeichnet einen Vorgang, der sich über einen gewissen Zeitraum erstreckt und mit einem Ergebnis endet [Dud13]. In der vorliegenden Arbeit werden die Prozesse durch Programme angestoßen und ausgeführt. Ein Programm ist die „zur Lösung einer Aufgabe vollständige Anweisung oder Verarbeitungsregel an einem Rechner“ [WHB12, S.24]. Aufgrund ihrer Verbindung werden die Begriffe „Programm“ und „Programmprozess“ in der vorliegenden Arbeit synonym für alle Vorgänge verwendet, die in der Prozessebene ablaufen.

Im Gegensatz zum Programmprozess steht der Geschäftsprozess. Dieser stellt eine Folge von Wertschöpfungsaktivitäten dar, die durch den Einsatz bestimmter Faktoren eine Leistung erzeugen, aus welcher der Kunde einen Nutzen für sich ziehen kann [Spr13, S.177]. Diese Aktivitäten besitzen eine zeitlich-logische Abfolge und sind in ihrer zeitlichen Ausdehnung durch ein Beginn- und ein Endereignis begrenzt. Eine Aktivität wird dabei von einem entsprechenden Vorgänger ausgelöst und bedingt wiederum weitere Aktivitäten [Mer12, S.65 ff.][Web12]. Zur optischen Darstellung und Abgrenzung einzelner Geschäftsprozesse untereinander ist eine Vielzahl von Verfahren entwickelt worden. In der vorliegenden Arbeit wird mit der integrierten Unternehmensmodellierung (IUM) gearbeitet. Diese beschreibt ein „Konzept für die Abbildung der verschiedenen Aspekte von Produktionsunternehmen“ [SMJ93, S.60]. Sie bietet eine gute Möglichkeit zur übersichtlichen Darstellung und erleichtert damit die Erarbeitung einer „festgelegten Art und Weise des Vorgehens“ („Methodik“ gem. [Dud13]) für das Erstellen der Formulare im betrieblichen Informationssystem. Diese Methodik, die im Laufe der Arbeit entwickelt wird, muss in einer geeigneten Art und Weise dargestellt werden, um sie für den Anwender möglichst unmittelbar anwendbar zu machen. Hierzu wird die Auszeichnungssprache XML (Extensible Markup Language) benutzt. Sie erlaubt, die entwickelten Inhalte so zu strukturieren und zu beschreiben, dass sie unabhängig von Anwendung und Plattform angewendet werden können [Von07, S.29 f.].

2.2 Einordnung des Themas in den Stand der Wissenschaft

[BF97] stellt eine Reihe von Werkzeugen zum Erstellen von Software innerhalb eines betrieblichen Informationssystems vor. Er stellt fest, dass die Sichten auf ein System mit der Begriffswelt und den Erfahrungen des Benutzers in größtmögliche Übereinstimmung gebracht werden sollten, um eine effektive Nutzung des Systems zu erreichen [BF97, S.54]. Die „Sicht“ ist dabei die Darstellung von mehreren inhaltlich zusammenhängenden Datenelementen. Er nutzt für die Untersuchung der Formulare in seinem System ein Tabellenschema, in das die Eckdaten einer Sicht, wie die sichtbaren Datenelemente, die damit durchführbaren Operationen, die aus der Sicht erreichbaren anderen Sichten sowie

notwendige Bedingungen für die Operationen und Übergänge eingetragen werden. Außerdem erstellt er ein Entity Relationship Modell zu den inhaltlichen Zusammenhängen und macht es zur Grundlage für die optische Gestaltung des GUI.

In [Bic97] werden konkrete Hinweise darauf gegeben, wie ein Informationssystem innerhalb eines Unternehmens aufgebaut werden kann. Zur Entwicklung des GUI wird dort herausgestellt, dass die einheitliche optische Gestaltung der Einzelelemente des Informationssystems besonders stark dazu beiträgt, dass der Anwender sie als zusammenhängend empfindet. Noch wichtiger ist, dass ein einheitliches Design die Orientierung im System entscheidend erleichtert. Da der Fokus seiner Arbeit jedoch nicht auf der Entwicklung der Einzelkomponenten, sondern auf der Einführung eines gesamten Informationssystems liegt, wird auf Details zur Verbindung des GUI mit der Prozess- und Datenbankebene verzichtet.

[Neu03] geht in seiner Arbeit detailliert auf die Entwicklung einer einheitlichen Oberflächengestaltung und einer Anbindung an die hinterliegende Datenbank im Rahmen des Entwurfs eines kompletten Informationssystems ein. Für die Beschreibung seiner Oberflächendokumente verwendet er XML und stellt die Möglichkeiten zur Regelung der Abläufe zwischen GUI, Datenbanksystem und externen Systemen durch die mit XML beschriebenen Dokumente dar.

[Dra05] stellt in seinem Werk eine komplette Modellierungsmethode für jede Art formularbasierter Firmensoftware vor. Hierzu nimmt er bereits bekannt Modellierungsverfahren und -sprachen als Grundlage und passt sie entsprechend an. Die Verbindungen zwischen den im Formular repräsentierten Inhalten und den ausgewählten Repräsentationsmitteln werden in verschieden stark detaillierten Graphen dargestellt. Eine entsprechende Beschreibungssprache wird ebenfalls verwendet.

Ausgangspunkt der Betrachtungen in der vorliegenden Arbeit ist ein betriebliches Informationssystem. Die Tatsache, dass auch [BF97], [Bic97] und [Neu03] mit diesem System arbeiten, zeigt, dass es ein etablierter und immer noch aktueller Betrachtungsgegenstand ist. Gemäß der von [BF97] geforderten Orientierung der Systementwicklung an der Erfahrungs- und Begriffswelt des Benutzers, ist die Sicht, die der Benutzer auf das System hat, der Ausgangspunkt für alle Überlegungen in dieser Arbeit.

Bedenkt man die bereits in 2.1 bereits erwähnte Heterogenität der Benutzerkenntnisse in einem betrieblichen Informationssystem und den in [Cla06] erwähnten Vorteil, den Zugang zu den unteren Ebenen dieses Systems über Formulare passgenau abstufen zu können, bietet sich die Erstellung eines formularbasierten GUI an. Die Benutzer erhalten nur die Zugriffe auf die darunter liegenden Ebenen, die im jeweils zugänglichen GUI vorgesehen und in den entsprechenden Programmroutinen hinterlegt sind. Diese Zugriffe wiederum sind dann entsprechend ihrer Kenntnisse und Aufgaben gestaltet. Sowohl [Dra05], als auch [Neu03] fokussieren ihre Ausführungen auf formularbasierte Darstellungen. [Neu03] begrenzt sich in seiner Arbeit dabei auf das betriebliche Informationssystem, während [Dra05] grundsätzlich jede Art von Anwendung, die mittels formularbasierter GUIs gestaltet werden kann, betrachtet.

Der Wunsch nach Maschinenlesbarkeit der Struktur wird verständlich, wenn man die Änderungseinflüsse betrachtet, die auf ein betriebliches Informationssystem wirken.

[Sch12] kategorisiert die Änderungseinflüsse in drei Einflussgruppen: Mensch, Technik und Organisation. Diese Gruppen sind von Schuh zwar auf PPS-Systeme zugeschnitten, lassen sich jedoch leicht auf ein betriebliches Informationssystem übertragen. Zusätzlich zu den drei Gruppen von Schuh kann man eine Unterscheidung zwischen internen und externen Einflüssen treffen. Tabelle 2.1 zeigt Beispiele dafür, welche Einflussnahmen im Beispielsystem auftreten können.

	Intern	**Extern**
Mensch	Vertrieb benötigt neues Feld in Formular	Kunde wünscht Anpassung in Kundenportal
Technik	neues Programm in Programmlandschaft benötigt veränderte Schnittstelle	Standard eines verwendeten Programmierkomponenten ändert sich
Organisation	Prozessorganisation wird restrukturiert	Gesetzgebung macht neue Vorgaben zum Datenschutz

Tab. 2.1.: Beispiele für Einflüsse auf ein betriebliches Informationssystem

Eine Software, die im betrieblichen Informationssystem verwendet wird, besitzt wiederum drei Ebenen auf welche die Veränderungen einwirken können: das GUI, die Prozessebene und die hinterlegten Datenbanken. Eine Datenbank nach Inbetriebnahme der Software zu ändern, ist in der Regel mit hohem Aufwand verbunden. Änderungen an den Programmroutinen können ebenfalls Änderungen an den Formularen nach sich ziehen, wenn Überprüfungsregeln geändert werden, die den Feldern zugewiesen werden müssen. Zuletzt können die Formulare selbst natürlich in ihrem Aufbau verändert werden. Je systematischer die gesamte Grundstruktur ist, desto besser kann die Formularerstellung automatisiert werden und desto geringer ist der Aufwand zum Einarbeiten von Änderungen in allen genannten Bereichen. Diesem Gedanken folgend, erlauben sowohl die Darstellungen von [Dra05] und [BF97], als auch der Ansatz von [Neu03] die für diese Arbeitserleichterung nötige Automatisierbarkeit. [Dra05] und [BF97] nutzen eigens entwickelte Beschreibungsarten dafür, während [Neu03] das offene XML-Format heranzieht. Zusätzlich zur Automatisierbarkeit erlaubt die Beschreibung der Formularstruktur in XML die von [Bic97] geforderte Einheitlichkeit der optischen Gestaltung, da eine strikte Trennung von Struktur, Inhalt und Darstellung bei XML von vorneherein vorgesehen ist [Von07].

	Bullinger1997	Bichler1997	Neuschwinger2003	Draheim2005	Köster2013
betriebliches Informationssystem	o	o	o		o
auf andere Systeme übertragbar				o	o
GUI formularbasiert			o	o	o
Beschreibung in XML			o		o
Formularerstellung automatisiert	o		o	o	o

Tab. 2.2.: Einordnung der Arbeit

Tabelle 2.2 zeigt eine schematische Übersicht der angesprochenen Aspekte dieser Arbeit. Die fünf Aspekte bilden mit den vier herangezogenen Literaturquellen und der vorliegenden Arbeit eine Matrix. Es ist markiert, welches Werk welche Aspekte behandelt. Dadurch tritt klar hervor, dass die vorliegende Arbeit eine Kombination von Aspekten behandelt, die so bisher noch nicht abgedeckt wurde.

3 Bildung und Untersuchung des Beispielprozesses

Im folgenden Kapitel wird ein Beispielprozess aus einem produzierenden Industriebetrieb exemplarisch auf die mögliche Verwendung von Formularen untersucht. Die potenziell einsetzbaren Dokumente werden in ihre inhaltlichen Bestandteile zerlegt. Um zu klären, welche weiteren Einflüsse die inhaltlichen Bestandteile und andere Aspekte von Formularen aufeinander haben, wird außerdem ein leeres Formular betrachtet und zerlegt. Das Zusammenwirken aller in diesen zwei Untersuchungen identifizierten Aspekte wird anschließend untersucht und dargestellt. Am Ende des Kapitels werden aus den gewonnenen Erkenntnissen dann die Anforderungen an die zu entwickelnde Methodik abgeleitet.

3.1 Beschreibung des Prozesses

Betrachtet wird das betriebliche Informationssystem eines produzierenden Unternehmens. Dabei wird davon ausgegangen, dass das System Schnittstellen für externe Kunden und Lieferanten bietet und alle vorhandenen Abteilungen des Betriebes in das System eingebunden sind. Der vorliegende produktionsauslösende Bestellprozess, dessen IUM-Graph in Anhang A.1 und A.2 dargestellt ist, wurde aus verschiedenen Gründen unter den vielen möglichen Prozessen in einem solchen System ausgewählt: Er involviert mehrere Abteilungen, die in jedem produzierenden Industriebetrieb unabhängig von dessen Größe vorkommen. Vertrieb, Arbeitsvorbereitung, Produktion, Qualitätswesen und Logistik können zwar je nach Unternehmen unterschiedliche Bezeichnungen tragen, führen die im Prozess beschriebenen Vorgänge jedoch immer so wie beschrieben oder in abgewandelter Form aus. Der Prozess gewährleistet außerdem eine gute Nachvollziehbarkeit, da er in jedem produzierenden Unternehmen vorkommende Kernabläufe umfasst. Aus diesen Gründen ist er auf jede entsprechende Firma übertragbar und von besonderem Interesse für MaschinenbauingenieurInnen, da sie mit hoher Wahrscheinlichkeit in solch einem Prozess involviert sind.

Die vorliegende Darstellung in IUM erlaubt eine übersichtliche Trennung zwischen solchen Prozessen, die das physische Produkt betreffen und solchen, die die Ablaufsteuerung übernehmen. Die zur Umsetzung des Prozesses benötigten Formulare lassen sich anhand der Darstellungsform einfach den einzelnen Prozessschritten zuordnen, was die Lösung der Aufgabenstellung unterstützt.

Zur Orientierung in der Zeichnung (Anhang A.1/A.2) wird im Folgenden kurz ihre Gestaltung und ihr Inhalt erklärt: Zu sehen sind zwei parallele Geschäftsprozessstränge, von denen der obere in blauen und gelben Kästen enthaltene den Steuerungsprozess beschreibt. Hier werden ausschließlich Informationen in Form von Aufträgen manipuliert. Dabei wandeln die gelben Funktionen die blauen Aufträge von ihrem Ausgangs- in ihren Endzustand um. Dem Verlauf ist von links nach rechts in Pfeilrichtung zu folgen. Verzweigun-

gen, die mit nicht ausgefüllten Rhomben gekennzeichnet sind, stellen eine entweder/oder-Entscheidungsstelle dar. Ausgefüllte Rechtecke zeigen eine Aufspaltung des Stranges in zwei parallel weiterlaufende Stränge („logisches UND“). Die gleiche Lesart ist beim in der Mitte liegenden rotgelben Strang anzuwenden. Im Unterschied zum oberen Strang verändern die Funktionen hier jedoch physische Materialien. Die grünen Kästen ganz unten weisen die Abteilung zu, welche die jeweilige Funktion ausführt. Ist einer Funktion keine Abteilung zugeordnet, ist die Abteilung für den Schritt zuständig, die zuletzt entlang des Stranges zugeordnet worden ist. In diesem Punkt wird zugunsten der Übersichtlichkeit des Graphen vom IUM-Schema abgewichen.

Der vorliegende Prozess zeigt das Vorgehen, gemäß dem eine Bestellung von einem Kunden abzuwickeln ist, die nicht aus bestehenden Lagervorkommen bedient werden kann, sondern für die eigens produziert werden muss. Der erste Teil des Auftragsstranges ist allerdings noch unabhängig davon, ob im weiteren Verlauf des Prozesses die Produktion beteiligt sein wird. Die Bestellung des Kunden wird vom Vertrieb entgegengenommen und geprüft. Aus dieser Prüfung ergibt sich die Entscheidung, ob eine Produktion notwendig ist und welches Lieferdatum dementsprechend zugesagt werden kann. Der sich bei der Entscheidung bildende Zweig „reiner Kommissionierauftrag“ würde weiter verfolgt, wenn genug Waren im Lager wären, um die Bestellung ohne zusätzliche Produktion zu bedienen. Diesen Zweig vollständig darzustellen würde das Diagramm unnötig vergrößern, böte aber für unsere Zwecke keinen Mehrwert, da er in weiten Teilen identisch ist mit dem Zweig, der auch Produktions- und Qualitätsprüfungsschritte enthält. Nur den ersten Kasten des Zweiges einzuzeichnen, gewährleistet dagegen einen zweckmäßigen Grad an Vollständigkeit des Diagramms, da die Notwendigkeit, an dieser Stelle eine Entscheidung zu treffen, verdeutlicht wird. Auf die vollständige Darstellung kann aber ohne Informationsverlust verzichtet werden.

Nachdem die Produktion durch den entsprechenden Auftrag angestoßen wurde, läuft sie gemäß den Angaben im mittleren Strang ab. Sowohl die Produktion als auch die Qualitätsprüfung können mehrschrittig ausfallen, sodass an den entsprechenden Stellen Schleifen im Prozess eingebaut sind. Abschließend wird die Ware versendet, womit sowohl der Produktionsstrang als auch der Auftragsstrang, durch den Produktion, Qualitätsprüfung und Versand weiterhin überwacht wurden, enden.

3.2 Untersuchung der Formularbenutzung im Beispielprozess

Die Weitergabe von prozessrelevanten Daten erfolgt entlang des dargestellten Prozesses über die Eingabe und das Abrufen der Daten mithilfe einer gemeinsamen Datenbank. Die Darstellung des Prozesses mit IUM erlaubt eine einfache Zuordnung zwischen den einzelnen Prozessschritten und den Formularen, die dazu beim jeweiligen Schritt verwendet werden können. Die Quadrate, die im Diagramm an einigen Stellen unmittelbar an den Elementen eingezeichnet sind, ordnen die verwendeten Formulare über eine Kennnummer fest dem Schritt zu, in dem sie verwendet werden. Ein orange eingefärbtes Quadrat steht dabei für die Eingabe von Werten während des Prozessabschnittes in die zugeordnete Maske, während violett eine reine Ausgabe der für den Schritt relevanten Daten markiert.

Sollte beides relevant sein, sind beide Farben in einem Rechteck zu sehen. Die Zuordnung zwischen den Formularnummern und den Namen bzw. dem Inhalt der entsprechenden Masken lässt sich dem Anhang A.3 entnehmen. Er zeigt die Zerlegung der identifizierten Formulare in ihre inhaltlichen Bestandteile. Das Diagramm ist von links nach rechts zu lesen und zeigt in der ersten Ebene die einzelnen Formulare mit ihrer Benennung gemäß der Zuordnung zwischen Anhang A.1/A.2 und Anhang A.3 durch die Nummerierung. Die zweite Ebene gibt die Möglichkeit bestimmte wiederkehrende inhaltliche Gruppen, wie z.B. Adressblöcke zu benennen und damit inhaltliche Atomare gleichen Namens anhand dieser Gruppen zu unterscheiden. In der dritten Ebene sind die inhaltlichen Atomare selbst zu sehen. Der Begriff kennzeichnet Einheiten, die inhaltlich insofern nicht weiter zerlegbar sind, als dass jede von ihnen mit genau einem Dateneintrag in der Datenbank verbunden ist. Daten, die einmal im Laufe des Prozesses eingegeben wurden oder aus den in der Datenbank hinterlegten Stammdaten abgerufen wurden, werden in der Regel wiederverwendet. Jedes Atomar in jeder Maske, in der es vorkommt, einzeln in der dritten Ebene aufzuführen, würde das Diagramm enorm vergrößern, ohne einen inhaltlichen Mehrwert zu bieten. Daher werden Felder, die bereits in einem Vorgängerformular vorgekommen sind, weggelassen und durch den Sammelplatzhalter „...“ ersetzt.

Jedem inhaltlichen Atomare kann man eine Reihe bestimmter Eigenschaften zuweisen. Als Beispiel für so eine Eigenschaft wird rechts von der dritten Ebene jedem Atomar seine Zugriffsart zugewiesen. Hiervon gibt es insgesamt drei Arten im vorgestellten Beispiel: Der Lesezugriff erlaubt dem Zugreifenden zwar die Betrachtung des Datenfeldinhaltes aber keine Manipulation daran. Da es der häufigste Fall in unserem Beispiel ist, wird er als Standardfall gewählt und nicht gesondert aufgeführt, um die Darstellung möglichst übersichtlich zu halten. Die Zugriffsart „Neueingabe“ erlaubt, neue Datensätze in der Datenbank zu erzeugen, während „Änderung“ nur eine Manipulation bereits bestehender Daten gewährt. Jedes inhaltliche Atomar hat in jedem Formular, in dem es auftaucht, genau eine Zugriffsart. Die grafische Darstellung dieser Zuordnung ist in Anhang A.4 oben links zu finden.

3.3 Zerlegung des Formulars in seine nicht inhaltlichen Bestandteile

Betrachtet man ein inhaltsloses Formular, kann man seine Bestandteile in eine Kategorienhierarchie gliedern, wie sie in Abbildung 3.1 zu sehen ist. Rein optische Bestandteile dienen der räumlichen Gliederung des sich ergebenden Formulars und repräsentieren keine Informationen aus der Prozess- oder Datenbankebene. Sie können durch inhaltliche, technische oder gestalterische Aspekte beeinflusst werden. So kann es z.B. sinnvoll sein, inhaltlich verwandte Daten als Block anzuordnen, eine Überlappung mehrerer Elemente zu vermeiden oder ein Corporate Design bei der Erstellung des GUI zu beachten. Selbst Einfluss ausüben auf den Inhalt oder die Technik können sie aber nicht.

Funktionale Bestandteile können Informationen repräsentieren oder Programme auf der Prozessebene anstoßen. Sie lassen sich wiederum in drei Gruppen aufteilen: Reine Eingabeelemente wie z.B. Buttons lösen vorher hinterlegte Prozesse aus, die Daten mani-

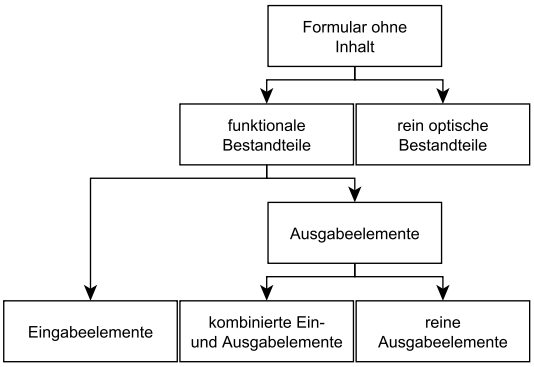

Abb. 3.1.: formale Elementehierarchie eines Formulars

pulieren können oder andere Formulare aufrufen. Will man einzelne Dateninhalte aus der Datenbank gezielt manipulieren, ist es nötig, den bisherigen Inhalt vor der Veränderung sichtbar zu machen. Kombinierte Ein- und Ausgabeelemente zeigen die vorhandenen Dateninhalte an und lassen zu, dass man auf sie zugreift und sie ändert. Durch das Entfernen der Fähigkeit, das angezeigte Datenelement zu manipulieren, kann jedes kombinierte Ein- und Ausgabeelement in ein reines Ausgabeelement umgewandelt werden. Andersherum kann man jedes Ausgabeelement, das eingesetzt wird, mit der Fähigkeit ausstatten, den gezeigten Inhalt manipulierbar zugänglich zu machen. Im Folgenden wird das reine Ausgabeelement als Sonderfall des kombinierten Ein- und Ausgabeelementes behandelt und beides zusammen mit dem Oberbegriff „Ausgabeelement“ bezeichnet.

Zerlegt in Grundbestandteile kann man für die Ausgabeelemente vier mögliche Typen identifizieren:

Inhaltsfelder, Comboboxen machen den Inhalt des Datenelements in der Regel als Text zugänglich und erlauben eine beliebige Eingabe von Werten ohne vorherige Einschränkungen

Listen lassen eine beliebige Menge von Optionen aus einer vorher festgelegten Anzahl von Möglichkeiten wählen

Dropdownmenüs, Radiobuttons, Spinner, Slider lassen genau eine Option aus einer vorher festgelegten Anzahl von Möglichkeiten wählen

Checkboxen lassen genau eine Option von insgesamt zwei vorher festgelegten Möglichkeiten wählen

Wie in der Aufzählung bereits andeutet, gibt es zu jedem der vier Typen verschiedene Möglichkeiten der optischen Umsetzung, die wiederum verschieden gut geeignet sind, um unterschiedliche Inhalte zu repräsentieren. Die dargestellte Auswahl von Repräsentationselementen ist nicht vollständig, sondern zeigt eine subjektive Menge häufig vorkommender Ausprägungen. Sie kann beliebig eingeschränkt oder erweitert werden, solange möglichst für jeden Typ mindestens eine Möglichkeit der Darstellung erhalten bleibt.

3.4 Gegenseitige Beeinflussung der Formularaspekte

In Anhang A.4 wird neben der Zuordnung von Zugriffsarten, die bereits in Abschnitt 3.2 erläutert wurde, auch gezeigt, welche anderen Aspekte von formularbasierten Systemen sich gegenseitig beeinflussen und wie die im vorherigen Abschnitt erarbeiteten Repräsentationselemente davon tangiert werden. Im Folgenden wird diese Einflusshierarchie detailliert beschrieben.

Das Bild ist von oben nach unten zu lesen und gliedert sich in mehrere Verzweigungen, die von links beginnend nacheinander beschrieben werden. Weiß hinterlegte mit Volllinien umgebene Rechtecke zeigen die einzelnen Aspekte wie z.B. das Formular selbst, Inhalte und Eigenschaften von Inhalten oder Repräsentationselemente. Die Kästen mit Rahmen aus Strichlinien zeigen die Art der Beziehung zwischen den Aspekten, die mit ihnen verbunden sind. Oft ist diese Beziehung durch eine feste Zuordnung zweier Aspekte zueinander gekennzeichnet. In diesem Fall kann das Diagramm als ausführliche Darstellung eines Klassendiagramms gelesen werden, wie das folgende Beispiel verdeutlicht. Die Beziehung zwischen dem Spezialdatentyp und dem Basisdatentyp, die beide mittig in der Darstellung zu finden sind, ist eine n:1-Beziehung. Jeder Spezialdatentyp ist genau einem Basisdatentypen zugeordnet, aber ein Basisdatentyp kann mehrere Spezialdatentypen besitzen. Da es in den Beziehungen zwischen Aspekten aber auch Zusammenhänge gibt, die nicht bilateral sind oder keine feste Zuordnung erlauben, sind die Beziehungen zur Verdeutlichung ausführlicher dargestellt, als im klassischen ER-Diagramm.

Der erste Zweig in A.4, der bereits in Abschnitt 3.2 betrachtet wurde, legt die Zuordnung der Zugriffsart zu jedem einzelnen inhaltlichen Atomar in jedem Formular fest. Die Darstellung macht deutlich, dass eine beliebige Anzahl von inhaltlichen Atomaren n jeweils genau einmal in einer beliebigen Anzahl von Formularen m vorkommen kann und jeder dieser Kombinationen genau eine Zugriffsart zugewiesen werden muss. In einem Formular nicht vorkommende Atomare sind mit einem Minuszeichen versehen. Die Zugriffsart wiederum regelt, ob bei der Benutzung des Feldes eine Plausibilitätsprüfung vorgenommen werden muss. Diese Plausibilitätsprüfung ist nur nötig, wenn Informationen in das System eingegeben werden. Wird diese Vorgehensweise im gesamten System verfolgt, sind alle Daten im System zum Zeitpunkt ihrer Eingabe und jeder Änderung überprüft worden und damit immer konform zu ihrer jeweiligen Prüfungsvorschrift. Daten, auf die im betrachteten Fall nur Lesezugriff herrscht, werden sich im Verlauf einer Formularbenutzung nicht ändern, also müssen sie auch nicht noch einmal überprüft werden. Daher ergibt sich die folgende Zuordnung: Lautet die Zugriffsart „Neueingabe“ oder „Änderung“, so muss eine Plausibilitätsprüfung nach jeder Manipulation geschehen. Ist die Zugriffsart „Lesen“ angegeben, muss keine Überprüfung stattfinden. Die Zuordnung ist damit vollständig zwischen allen möglichen Ausprägungen der beiden Aspekte erfolgt.

Der nächste Ast beginnt wiederum beim inhaltlichen Atomar ganz oben und wird in der Mitte der Zeichnung weiter nach unten fortgeführt. Er zeigt, dass jedes Atomar genau einen Spezialdatentyp besitzt und jeder Spezialdatentyp jeweils genau einem, keinem oder mehreren inhaltlichen Atomaren zugewiesen werden kann. Aufgrund der Fülle von möglichen Ausprägungen der beiden Aspekte ist die Zuordnung nicht vollständig,

sondern wird durch Platzhalter symbolisch gezeigt. Die beiden folgenden Einflussebenen sind am besten parallel zu betrachten. Die linke Fortsetzung zeigt, dass jeder Spezialdatentyp auf genau einem Basisdatentypen aufbaut. Im vorliegenden Fall wurden als Basisdatentypen „String“, „Integer“, „Real“, „Boolean“ und „Datum“ gewählt. Diese Auswahl ist exemplarisch zum vorliegenden Beispielprozess gewählt worden und kann in anderen Systemen anders zusammengesetzt sein. Die Einflussbeziehungen zu anderen Formularaspekten verändern sich dadurch aber nicht. Die Spezialdatentypen beruhen auf jeweils einem Basisdatentyp und tragen deren Eigenschaften weiter in sich. Sie bestimmen, welche Zeichen in einem Dateneintrag verwendet werden dürfen und bilden damit die Grundlage für die Vorschrift der Plausibilitätsprüfung, die im rechten Ast unten gezeigt wird. Die aus dieser Kombination entstehenden Datentypen werden im Folgenden mit der Notation „*Basisdatentyp.Spezialdatentyp*“ beschrieben. Die in kursiver Schrift erscheinenden Begriffe „Basisdatentyp“ und „Spezialdatentyp“ stehen dabei als Platzhalter für mögliche Ausprägungen der beiden Typen. Der Spezialdatentyp besitzt die Möglichkeit, das Regelwerk, dass sein jeweiliger Basisdatentyp vorschreibt, um weitere Regeln zu erweitern. Dabei sind zwei Arten von Vorschriften denkbar: Zum einen kann der erlaubte Zeichenraum weiter eingeschränkt werden. Zum anderen kann der Spezialdatentyp eine bestimmte Syntax des Inhalts vorschreiben. Eine begrenzte Länge, zwingend notwendige Zeichen und eine bestimmte Zeichenreihenfolge sind hier denkbar. Diese Regeln machen deutlich, dass sich das Konstrukt „*Basisdatentyp.Spezialdatentyp*“ am Prinzip der Vererbungsregeln der objektorientierten Programmierung orientiert. Zu jedem Basisdatentyp gibt es immer genau einen Spezialdatentyp, der keine Regeln zu denen des Basisdatentyps hinzufügt. Dieser Spezialdatentyp heißt „*Basisdatentyp.*basis“ und muss vorhanden sein. Außerdem ist es wichtig zu beachten, dass ein Spezialdatentyp die Regeln seines Basisdatentyps niemals aufheben kann, sondern immer nur weiter einschränken wird. Unmittelbar aus den Regeln, die Basis- und Spezialdatentyp zusammen aufstellen, ergibt sich dann die Vorschrift der Plausibilitätsprüfung. Jeder Spezialdatentyp erzeugt genau eine solche Vorschrift und jede Vorschrift gehört zu genau einem Spezialdatentyp. Mit der nicht vollständigen Darstellung dieser Beziehung ist dieser Zweig beendet und der weitere Einfluss des Basisdatentyps im links aus dem Spezialdatentyp hervorgehenden Zweig kann betrachtet werden.

Der Basisdatentyp gibt einen Hinweis darauf, welche Repräsentationsart seines Inhaltes im Formular sinnvoll ist. Hierbei handelt es sich nicht um eine feste Zuordnung, wie in den bisher betrachteten Zweigen, sondern um eine subjektive Bewertung der jeweiligen Kombination aus Basisdatentyp und Repräsentationselement, da es theoretisch immer mehrere Möglichkeiten gibt, einen Inhalt mehr oder weniger sinnvoll zu repräsentieren. Die Entscheidung, welches Repräsentationselement im Einzelfall das am besten passendste ist, hängt von einer Vielzahl subjektiver Faktoren ab. Die beispielhafte Bewertung in Anhang A.4 findet mittels einer dreistufigen Skala statt:

+ bedeutet, dass das gewählte Repräsentationselement gut geeignet ist, um den Basisdatentyp darzustellen

o bedeutet, dass das gewählte Repräsentationselement geeignet ist, um den Basisdatentyp darzustellen, die Darstellung aber u.U. kompliziert oder mit Einschränkungen verbunden sein kann

- bedeutet, dass das gewählte Repräsentationselement eher schlecht geeignet ist, um den Basisdatentyp darzustellen, da bei dieser Kombination komplizierte, unübersichtliche oder schlecht verständliche Darstellungen entstehen

Die oben für den Beispielprozess gewählten Basisdatentypen sind als Spaltenüberschriften zu finden, während die Zeilentitel Beispiele aus den vier Grundtypen von Repräsentationselementen zeigen, die in Abschnitt 3.3 identifiziert wurden. Die eindeutige Zuordnung eines eines Repräsentationselementes findet erst durch das inhaltliche Atomar selbst statt, wie im ganz rechten Zweig der Darstellung A.4 zu sehen ist. Auch sie ist aufgrund der hohen Anzahl möglicher inhaltlicher Atomare unvollständig.

4 Bildung der Methodik

Basierend auf den Einflüssen der einzelnen Formularaspekte aufeinander, die in Kapitel 3 erarbeitet worden sind, wird im nun folgenden Kapitel eine Methodik erarbeitet, die diese Erkenntnisse nutzt und ein funktionsfähiges Formular erzeugt. Dazu werden zunächst die Anforderungen an die Methodik explizit herausgearbeitet und formuliert. Dann wird eine Möglichkeit zu ihrer Erfüllung aufgezeigt, die sich an einem Blackboxmodell orientiert. Zunächst die benötigten Startinformationen aufgezählt und mit knappen Erläuterungen versehen. Die zwei Abschnitte danach gehen darauf ein, welche Operationen mit den zur Verfügung gestellten Informationen durchgeführt werden müssen, um das gewünschte Ergebnis zu erhalten.

4.1 Anforderungen an die zu bildende Methodik

Die Zerlegung des Formulars in Abschnitt 3.3 gibt uns Auskunft darüber, welche Bestandteile von der Methodik erzeugt werden müssen und welche davon mit entsprechenden Funktionen verbunden werden müssen. Die Zuordnungen, die in A.4 dargestellt sind und in Abschnitt 3.4 beschrieben wurden, folgen inhaltlich auseinander. Das bedeutet jedoch nicht, dass eine Maschine z.B. aus dem Namen eines inhaltlichen Atomars den zugehörigen Datentyp erkennen kann. Deswegen müssen die einzelnen Aspekte aus der Darstellung als geordnete Startinformation zur Verfügung gestellt werden. Die Speicherstruktur dieser Startinformationen muss zweckmäßig gewählt werden.

Nach der vollständigen Beschreibung der Startinformationen muss die Blackbox, die zwischen ihnen und dem voll funktionsfähigen Formular als Ergebnis steht, mit Inhalt in Form von Verarbeitungsregeln und -systematiken gefüllt werden. Diese müssen zwei Schritte durchführen, nach deren Erledigung das sichtbare und benutzbare Formular fertiggestellt ist.

- Die für die Erstellung des GUI relevanten Startinformationen müssen so ausgewertet werden, dass eine benutzbare und ergonomische Maske erzeugt wird
- Alle Funktionen, die während der Datenmanipulation durch den Benutzer benötigt werden, müssen verfügbar gemacht werden.

4.2 Startinformationen

Damit die unten vorgestellten Verarbeitungsregeln ohne Unterbrechung ablaufen können, muss geklärt werden, welche Startinformationen dafür benötigt werden und welche Form diese haben müssen. Die folgenden Angaben müssen für alle Ausgabeelemente gemacht werden. Muss der beschriebene Teil der Startinformationen auch für reine Eingabeelemente gemacht werden, so ist dies explizit angegeben.

Allen Elementen im Formular muss ein **Inhalt** zugewiesen werden. Reine Eingabeelemente müssen mit einer Funktion verbunden werden, die bei Betätigung des Elementes ausgelöst werden soll. Von jedem Ausgabeelement wird ein inhaltliches Atomar aus der Datenbank repräsentiert. Zu diesem Atomar muss der entsprechende Pfad in einer Art und Weise angegeben werden, die von dem später die Startinformationen verarbeitenden Programm verstanden werden kann.

Eine zum Inhalt passende Darstellung wird über eine adäquate Auswahl des Repräsentationselementes (kurz: **Repräsentation**) gewährleistet. Es wird entweder gemäß eines vorher festgelegten Kriterienkatalog automatisch gewählt oder als gezielte Angabe beim einzelnen inhaltlichen Atomar erwähnt.

Das **Label** eines Elementes dient dem kenntlich machen des Inhalts im fertigen Formular. Dies ist wichtig, damit der Anwender den ihm angezeigten Inhalt in einen sinnvollen Kontext stellen kann. Denkbar ist die Unterscheidung, ob z.B. der Inhalt „2“, der in einem Feld angezeigt wird, der lagernden Menge eines Produktes entspricht oder der geforderten Liefermenge des Kunden.

Bei Ausgabeelementen muss angegeben werden, welche **Zugriffsart** im aktuellen Formular zugeteilt wird. Sie wird aus einer vorher definierten Menge von Möglichkeiten gewählt und legt fest, ob ein Atomar auf Plausibilität geprüft werden muss oder nicht.

Der **Datentyp** wird ebenfalls nur bei Ausgabeelementen benötigt. Erfolgt eine Plausibilitätsprüfung, richtet sich die anzuwendende Prüfvorschrift nach ihm. Die Auswahl an zulässigen Datentypen ist beschränkt und muss mit den in der hinterlegten Datenbank verfügbaren Datentypen abgestimmt werden.

Dependenzen regeln den Einfluss der inhaltlichen Atomare bzw. deren Änderungen aufeinander. Oft können sie nicht explizit einem inhaltlichen Atomar zugewiesen werden. Deswegen bilden sie für jedes Formular einen eigenen Katalog.

4.3 Erstellung der Formularmaske

Die Startinformationen zum Inhalt, der Repräsentation und dem Label werden im ersten Schritt, der in den Anforderungen an die Methodik genannt wird, benötigt. Während der Inhaltspfad und das Label einfach ausgelesen werden können, um die Formularmaske zu erzeugen, muss der Auswahl der Repräsentation mehr Aufmerksamkeit geschenkt werden. In Abschnitt 3.4 wurde erläutert, dass die Repräsentation im Endeffekt von jedem einzelnen inhaltlichen Atomar abhängt, der Basisdatentyp jedoch auch Hinweise darauf geben kann, welche Repräsentationen für das jeweils behandelte Atomar zweckmäßig sein können. Aus diesen beiden Einflüssen ergibt sich die folgende Überlegung.

Gibt man die gewünschte Repräsentation bei jedem Atomar gesondert an, so bedeutet das ein hohes Maß an repetitiver Arbeit. Auf Grundlage der Bewertungen zur Eignung der Repräsentationen zur Darstellung der einzelnen Datentypen ist erkennbar, dass zur Zuordnung der Repräsentation sinnvolle Gruppen von Atomaren gebildet werden können, denen durch automatisiert nutzbare Regeln die gleiche Repräsentation zugewiesen werden kann. Trotzdem sollte es immer möglich sein, im Einzelfall Ausnahmen von diesen Regeln machen zu können. Um beiden Fällen gerecht zu werden, wird ein Standardregelwerk

angelegt, das die Vorgaben für die Gruppen enthält. Dieses Regelwerk soll eine beliebige Komplexität besitzen können und wird um die Möglichkeit der Abweichung von der Standardregel im Einzelfall ergänzt, indem bei jedem inhaltlichen Atomar eine gesonderte Angabe gemacht werden kann, falls es nötig ist. Das Regelwerk kann aus einer Kette von if-else-Anweisungen oder anderen ähnlichen Bausteinen in dem Programm abgelegt werden, welches die Startinformationen verarbeitet. Am Ablageort der eigentlichen Startinformationen wird angegeben, welche Repräsentationsarten überhaupt möglich sind und dort ist auch der Platz, an dem eine eventuell nötige Einzelfallentscheidung eingetragen wird. Dieser zweiteilige Mechanismus der Repräsentationszuordnung erlaubt eine weitgehende Automatisierung, die vermeidet bei jedem inhaltlichen Atomar eine Einzelangabe machen zu müssen und erlaubt gleichzeitig bei Bedarf die flexible Anpassung der Repräsentationsart. Auch eine zeitliche Flexibilität des Regelwerks ist gegeben. So kann man mit einem einzelnen Standardwert beginnen und mit zunehmender Erfahrung den Katalog mehr und mehr verfeinern, bis ein sinnvolles Mittel zwischen dem Detaillierungsgrad des Regelwerks und dem Aufwand seiner Pflege und der Eingabe von abweichenden Einzelfällen gefunden ist. Es liegt innerhalb der Verantwortung der Ersteller des Katalogs, die Feinmaschigkeit des Kataloges im Verhältnis zum Aufwand der Pflege zu halten. Wichtig ist bei der Ausarbeitung des Regelwerks die Vermeidung von unklaren Zuordnungen und Widersprüchen. Es muss also nach jeder Veränderung des Werkes geprüft werden, ob es inhaltliche Atomare geben könnte, auf die zwei Regeln ohne Vorzug der einen oder anderen angewendet werden.

Beispiel: Standardkatalog für Repräsentationselemente

Bei der Formulierung des Methodikalgorithmus wird festgelegt, dass ein Textfeld sehr gut dazu geeignet ist, alle vorkommenden Datentypen darzustellen. In den Standard für die Repräsentationselemente wird also eingetragen, dass immer wenn kein anderes Repräsentationselement angegeben wird, ein Textfeld zur Darstellung des jeweiligen inhaltlichen Atomars verwendet werden soll. Nach den ersten Testläufen zeigt sich jedoch, dass gerade bei boolschen Variablen die Angabe in einem frei befüllbaren Textfeld nachteilig ist und dort als Standardoption eine Checkbox gewünscht wird. Daraufhin wird der Standardkatalog um diese Option erweitert: Immer wenn keine andere Angabe gemacht wird, wird für Inhalte, die dem Datentyp „Bool.*Spezialdatentyp*“ angehören, eine Checkbox verwendet. Für alle inhaltlichen Atomare deren Datentyp nicht boolsch ist und für die keine Einzelangabe gemacht wurde, wird weiterhin eine Textbox genutzt. Zusätzlich haben mehrere Vertriebsmitarbeiter die Anmerkung gemacht, dass sich die begrenzte Zahl der Auswahlmöglichkeiten bei der Einteilung der Kunden in A-, B- und C-Kunden schneller erfassen ließe, wenn sie auf einer Skala angeordnet wären. Aus diesem Grund wird beim entsprechenden Feld im Formular „Kundenstammdaten“ als Repräsentationselement der Radiobutton gewählt. Dieser Eintrag wird direkt beim einzelnen inhaltlichen Atomar „Kundenbewertung“ eingetragen, da es sich um eine Einzelfallentscheidung handelt.

Sind die in diesem Abschnitt beschriebenen Startinformationen für alle Felder gesetzt, kann die Methodik die einzelnen Elemente des Formulars optisch sinnvoll anordnen. Mögliche Einflussparameter hierfür sind gestalterische Grundsätze wie eine überlappungsfreie Darstellung aller Felder, die passende Anordnung von Labels beim jeweiligen Inhalt, sowie ein vorher festgelegter Standardaufbau der Formulare z.B. im Rahmen des Corporate Designs. Auch die im nächsten Absatz genauer beschriebenen Informationen „Zugriffsart“ und „Dependenzen“ können hier mit einfließen. Die Zugriffsart kann evtl. herangezogen werden, um eine optische Abgrenzung zwischen manipulierbaren und nicht manipulierbaren Zellen zu erzeugen und die Dependenzen könnten genutzt werden, um eine Gruppierung von Atomaren zu automatisieren. In jedem Fall können die Richtlinien zur Anordnung, Gruppierung und Kennzeichnung der einzelnen Atomare in einem separaten Stylesheet abgelegt werden, um sie sauber getrennt vom Inhalt zu lagern. Dies hat den Vorteil, dass bei einer Änderung der Angaben zur Gestaltung keine Fehler in den inhaltlichen und strukturellen Angaben entstehen können.

4.4 Anbindung der bei Benutzung des Formulars nötigen Bestandteile

Zugriffsart, Datentyp und Dependenzen sind für die Maskenerstellung nur bedingt notwendig. Unabdingbar ist jedoch, dass sie so vom verarbeitenden Programm zur Verfügung gestellt werden, dass sie während der Benutzung der Formulare unmittelbar nach jeder Manipulation abgerufen werden können. Hierfür muss zunächst über den Datentyp die entsprechende Prüfvorschrift ausgelesen und angewendet werden. Gemäß des in Abschnitt 3.4 dargestellten Aufbaus der Datentypen kann für sie ein eigener Katalog angelegt werden. In diesem Katalog wird jeder Spezialdatentyp mit einem Namen gekennzeichnet, seinem Basisdatentyp zugewiesen und kann noch zwei weitere Informationen enthalten. Der durch ihn erlaubte Zeichenraum kann entweder durch eine Positiv- oder eine Negativliste festgelegt werden. In der Positivliste werden alle Zeichen angegeben, die der Datentyp erlaubt. Die Negativliste führt all jene Zeichen auf, die im Basisdatentyp erlaubt sind, im jeweiligen Spezialdatentyp aber nicht mehr verwendet werden dürfen. In beiden Katalogarten dürfen nur Zeichen angegeben werden, die auch der Basisdatentyp erlaubt. Zusätzlich zu den Katalogen kann eine bestimmte Zeichenreihenfolge und -länge festgelegt werden. Hierbei ist besonders wichtig, die Platzhalterzeichen so auszuwählen, dass sie mit dem verarbeitenden Programm kompatibel sind. In der Angabe zur Reihenfolge dürfen durch die Angabe von eigentlich nicht erlaubten Zeichen als Festzeichen in der Reihenfolge auch explizite Ausnahmen von den Katalogen gemacht werden.

Beispiel: Spezialdatentyp E-Mail

Der Eintrag des Datentyp „string.email“ soll die Prüfvorschrift für alle inhaltlichen Atomare festlegen, die diesen Typ zugewiesen bekommen. Zunächst wird der oben vergebene Name mit dem Basisdatentyp „string“ verbunden. Grundsätzlich sind in einer Mailadresse alle Zeichen erlaubt, die der Basisdatentyp bietet. Das @ ist jedoch ein Funktionszeichen und darf nur genau einmal vor-

kommen. Es wird also zunächst auf eine Negativliste mit „@“ als einzigem Eintrag gesetzt. Bei der Festlegung von Reihenfolgen innerhalb von Datentypen wurde vereinbart, dass das Zeichen * entweder für kein, ein oder beliebig viele zulässige Zeichen steht. Das Zeichen ? ersetzt genau ein beliebiges Zeichen. Dementsprechend wird als Reihenfolge für E-Mail-Adresse die Zeichenkette ?*@?*.??* festgelegt. So wird dafür gesorgt, dass das @ genau einmal zwischen zwei beliebigen Zeichenketten vorkommen muss, aber niemals anstelle eines der Platzhalter vergeben werden darf. Vor ihm und nach ihm muss mindestens jeweils ein Zeichen auftauchen. Es dürfen aber beliebig viele werden. Hinter dem Punkt müssen mindestens zwei Zeichen stehen, es können aber auch mehr sein.

Dependenzen müssen während der Formularbenutzung immer wieder abgeprüft werden, da sie Veränderungen in allen Bereichen des Formulars aufgrund jüngster Manipulationen vornehmen müssen können. Da sie durch verschiedenste Ursachen ausgelöst werden können und daraufhin Einfluss auf alle Bereiche des Formulars nehmen können, muss ihrer Ablagestruktur einige Aufmerksamkeit geschenkt werden. Jede Dependenz hat mindestens eine Ursache und eine Wirkung. Es ist aber auch denkbar, dass die kombinierte Änderung mehrerer Atomare verschiedene Wirkungen auf eine Vielzahl von anderen Atomaren hat. Aus diesem Grund sind die Dependenzen in einem eigenen Katalog abgelegt, der zu jeweils einem Formular gehört. Hat eine Dependenz mehrere Ursachen oder Wirkungen besteht zwischen ihnen eine logische UND-Verknüpfung. Ein logisches ODER dagegen kann durch die Kombination von mehreren Dependenzen erzeugt werden.

Beispiel: Realisierung von UND und ODER-Verknüpfungen in Dependenzen

Im Kundenbestellformular eines Halbzeugehändlers ist angegeben worden, dass Stangenmaterial mit Kreisquerschnitt geliefert werden soll. Nachdem der Kunde diese Auswahl im Formular getroffen hat, sollen weitere Felder sichtbar gemacht werden. Im ersten neuen Feld ist anzugeben, welches Material das Halbzeug haben soll, im zweiten Feld wird nach dem Durchmesser gefragt. Möchte der Kunde hingegen Stangen von Flachmaterial erwerben, muss er die Materialart sowie Dicke und Breite angeben. Außerdem muss für beide Fälle ein Eingabefeld für die gewünschte Länge zur Verfügung stehen. Aus diesem Szenario leiten sich mehrere UND- und ODER-Dependenzen ab, die sich alle auf ein mögliches Attribut der einzelnen inhaltlichen Atomare namens „sichtbar“ beziehen.

Das Atomar *Länge* muss angezeigt werden, wenn beim Atomar *Querschnittsform* der Wert „rund“ ODER „rechteckig“ angegeben wurde. Bei der runden Querschnittsform müssen Eingabefelder für *Länge* UND *Durchmesser* eingeblendet werden. Bei rechteckigem Material werden dagegen Möglichkeiten zur Eingabe von *Länge* UND *Höhe* UND *Breite* gebraucht.

Für die Dependenzen, die sich alle auf das mögliche Attribut der Sichtbarkeit beziehen, gilt also: Für die Auswahl von rund oder rechteckig müssen auf jeden

Fall zwei verschiedene Regeln angelegt werden. Die Einblendung der Felder für die Längen- und Querschnittsmaße in den beiden unterschiedlichen Fällen passt jeweils in eine gemeinsame Regel. Eine mögliche Ausprägung der Dependenzen für diesen Fall wäre also:

Dependenz 1: URSACHE{*Querschnittsform*: Inhalt = „rund“}
WIRKUNG{*Länge*: Attribut.sichtbar = „ja“}
WIRKUNG{*Durchmesser*: Attribut.sichtbar = „ja“}

Dependenz 2: URSACHE{*Querschnittsform*: Inhalt = „rechteckig“}
WIRKUNG{*Länge*: Attribut.sichtbar = „ja“}
WIRKUNG{*Höhe*: Attribut.sichtbar = „ja“}
WIRKUNG{*Breite*: Attribut.sichtbar = „ja“}

Jede Ursache und jede Wirkung besteht aus jeweils zwei Teilen. Zuerst wird das Zielatomar angegeben, auf das sich die Ursache oder Wirkung bezieht. Um dies zu ermöglichen, muss jedes inhaltliche Atomar eine innerhalb des Formulars eineindeutige ID zugewiesen bekommen. Wählt man diese ID Sinn tragend wie z.B. „lieferstandortname“ statt „atomar2“ erhöht dies die Lesbarkeit des Dependenzenkatalogs durch den Menschen. Auch das Hinzufügen anderer Eigenschaften von inhaltlichen Atomaren kann sich aus dem fertigen Dependenzenkatalog ergeben, denn jeder Bestandteil eines inhaltlichen Atomars, das in der Bedingung für eine Ursache oder Wirkung genutzt wird, muss in deren Definition auch vorkommen. Im Beispiel oben muss beispielsweise ein Attribut „sichtbar“ für die Atomare vorhanden sein. Die in diesem Kapitel angegebenen Startinformationen stellen also einen erweiterbaren Pflichtkatalog dar. Für die Formulierung der Bedingung, welche den zweiten Teil jeder Ursache und Wirkung bildet, sollte eine feste Syntax gewählt werden, die mit der beim verarbeitenden Programm benutzten Programmiersprache kompatibel ist.

Wurden alle nötigen Startinformationen zur Verfügung gestellt und in der oben beschriebenen Art und Weise verarbeitet, steht ein funktionsfähiges Formular zur Verfügung. Da die Erarbeitung eines vollständigen Algorithmus zu diesem Zweck den Rahmen dieser Arbeit übersteigt, sind die oben genannten Überlegungen und Anweisungen als prosaische, nicht vollständige Skizze des Programmablaufs zu betrachten.

5 Darstellung in XML

Im Folgenden wird die in Kapitel 4 ausgearbeitete Methodik auf ein Formular aus dem Beispielprozess aus Kapitel 3 angewendet. Zunächst wird die DTD erläutert, die sich aus den nötigen Startinformationen ergibt und dann wird das Formular „01 - Kundenbestellung“ als XML-Beispiel dargestellt. Dieses zusammenhängende Beispiel dient der Verdeutlichung des bisher erarbeiteten Stoffes, sowie als Grundlage für auf diese Arbeit aufbauende Arbeiten.

5.1 DTD

Der Quelltext der DTD, welcher in Anhang B.1 zu sehen ist, beschreibt die Struktur des Systems, das die Formulare und alle für sie benötigten Informationen enthält. Zu diesem Zweck enthält das Wurzelelement „system“ immer mindestens ein Formular und einen Spezialdatentypen (Zeile 2). Jedes Formular besteht aus mindestens einem funktionalen Bestandteil in Form eines inhaltlichen Atomars oder einer Funktion und optional einer beliebigen Menge von Dependenzen. Des Weiteren benötigt jedes Formular einen innerhalb des XML-Dokumentes eindeutigen Namen als ID-Attribut. Der Attributsname „name“ wird in der gesamten DTD immer dann verwendet, wenn eine Sinn tragende ID zweckmäßig ist, um Bezüge leichter lesbar zu machen. Inhaltliche Atomare enthalten eine Reihe von Attributen und Elementen, die zum größten Teil die in Kapitel 4.2 beschriebenen nötigen Startinformationen darstellen. Das ID-Attribut ist erneut Sinn tragend gewählt. Das Attribut „zugriff“ erzwingt die Auswahl einer Zugriffsart aus einer vorher festgelegten Menge. Dies ist auch bei der Repräsentation der Fall. Ihre Angabe ist jedoch nicht obligatorisch. Fehlt sie, greift das andernorts angelegte Standardregelwerk für Repräsentationselemente, wie in Kapitel 4.3 beschrieben. Das Attribut „datentyp“ refenrenziert auf die Spezialdatentypen, die am Ende der DTD aufgelistet sind, und ist wieder ein zwingend benötigtes Attribut. Das Attribut „sichtbar“ ist ein Beispiel für ein Attribut, welches nicht von den Startinformationen vorgeschrieben wird. An dieser Stelle können beliebige weitere Attribute eingefügt werden, die z.B. aufgrund von gewünschten Einflussnahmemöglichkeiten durch die Dependenzen benötigt werden. Die beiden Startinformationen „Inhalt“ und „Label“ werden danach als Elemente aufgeführt, die beide gemäß der Angaben in der DTD einen beliebigen Inhalt haben können, der sich aber an den Angaben in Kapitel 4 richten sollte.

Soll eine Funktion in das Formular eingebaut werden, sind ähnliche Informationen wie beim inhaltlichen Atomar nötig. Die Attribute „name“ und „sichtbar“ sind identisch angelegt und die Repräsentation unterscheidet sich nur durch die Auswahlmöglichkeiten. Im vorliegenden Fall ist nur die Möglichkeit vorgesehen, einen Button zu verwenden. Trotzdem ist kein fixer Wert festgelegt, um einfache Erweiterbarkeit zu gewährleisten. Das Attribut „benutzbar“ entspricht in etwa der Zugriffsart bei inhaltlichen Atomaren. Hier gibt es jedoch nur zwei Möglichkeiten. Entweder das Element ist benutzbar oder nicht.

Als Elemente sind wie bei inhaltlichen Atomaren ebenfalls eine Angabe zum Inhalt nötig und ein Beschriftungslabel. Der Inhalt enthält dabei nicht zwingend den Verweis auf einen Pfad in der Datenbank, sondern den Ablageort eines passenden Funktionsalgorithmus.

Der Dependenzkatalog für das einzelne Formular ist ab Zeile 21 beschrieben. Jede Dependenz besteht aus mindestens einer Ursache und mindestens einer Wirkung. Es ist darüber hinaus aber auch jede Kombination von mehrfachen Ursachen und Wirkungen möglich. Jede Ursache und jede Wirkung hat jeweils ein Attribut namens „ziel“, auf das sie sich bezieht. Damit wird die ID einer Funktion oder eines inhaltlichen Atomars referenziert. Innerhalb des jeweiligen Elements wird die Bedingung, welche die Ursache oder Wirkung kennzeichnet, in einer wie in Kapitel 4 beschriebenen festgelegten Syntax formuliert. Die einzelnen Dependenzen werden mit einer ID versehen, die jedoch nicht Sinn tragend ist.

Außerhalb des Formulars wird der Katalog der Spezialdatentypen abgelegt. Jede ID wird hier in der in Kapitel 3 beschriebenen Form *Basisdatentyp.Spezialdatentyp* gebildet und zusätzlich seinem Basisdatentyp noch mal explizit zugewiesen. Die möglichen Basisdatentypen sind in einer eigenen Liste festgelegt. Als Elemente können eine Syntaxregel und entweder eine Positiv- oder Negativliste abgelegt werden. Wird keines der drei Elemente eingetragen, handelt es sich um den Basisdatentyp. Wenn einer der beiden Listentypen angelegt wird, muss dieser mindestens einen Eintrag enthalten. Pro Eintrag sollte genau ein Zeichen angegeben werden.

Mit dieser DTD ist sind alle Formulare im System beschrieben und die für die Methodik benötigten Starinformationen so abgelegt, dass sie verarbeitet werden können. Einige Stellen, an denen sie erweitert werden kann, wurden innerhalb der Beschreibung aufgezeigt. Die Menge der möglichen Anknüpfungspunkte ist damit aber nicht erschöpft.

5.2 XML-Datei Kundenbestellung

In Anhang B.2 ist der Quelltext einer Instanz der oben beschriebenen DTD abgebildet. Sie stellt das aus dem Beispielprozess in Kapitel 3 bereits bekannte Formular „01 - Kundenbestellung“ dar, dessen inhaltliche Atomare in Anhang A.3 aufgelistet sind. Beim Erstellen des XML-Dokumentes wurde die Annahme getroffen, dass der Standardkatalog von Repräsentationselementen für alle Elemente des Datentyps „bool.*Spezialdatentyp*“ eine Checkbox und für alle anderen Elemente ein Textfeld vorsieht. Die Reihenfolge, in der die Elemente im Dokument aufgeführt werden, orientiert sich an einem möglichen Benutzungsablauf, ist für die Verarbeitung der Informationen durch das ausführende Programm jedoch von untergeordneter Bedeutung.

In das erste inhaltliche Atomar wird vom Benutzer die Kundennummer eingetragen, da sie in den entsprechenden Stammdaten als Primärschlüssel verwendet wird. Es wird angenommen, dass es sich hierbei um eine reine Ziffernkombination handelt, weswegen der Datentyp „int.basis“ zugewiesen wird. Ihre Neueingabe in die Auftragsdaten durch die Pfadangabe im Element <inhalt> identifiziert den Kunden, zu dem der Auftrag gehört damit eindeutig und ermöglicht den Abruf kundenspezifischer Daten, wie dem Kundennamen im nächsten Atomar. Dieser Name wird zwar im Formular zur Überprüfung der Kundennummer angezeigt, muss jedoch nicht in den Auftragsdaten abgelegt werden. Dementsprechend

ist bei <inhalt> der Pfad angegeben, aus dem heraus die Anzeige erfolgt und die Zugriffsart „lesen“ wird festgelegt. Einschränkungen der zulässigen Zeichenauswahl ergeben sich nicht, weswegen der Datentyp „string.basis“ gewählt wird.

Im zweiten Codeblock sind die Atomare beschrieben, die zur vollständigen Eingabe und Anzeige der gewünschten Lieferadresse benötigt werden. Es wird hierbei davon ausgegangen, dass für jeden Kunden eine Liste von Firmenstandorten hinterlegt ist, die jeweils über einen eindeutigen Namen identifiziert werden können. Dementsprechend wird auch in diesem Block wieder so verfahren, dass ein Identifizierer in die Auftragsdaten geschrieben wird und die weiteren Details der Adresse nur mit Lesezugriff in das Formular gespeist werden, nicht jedoch beim Auftrag abgelegt werden müssen. Die Repräsentation des Atomars „iAlieferstandortname“ weicht vom Standard ab, da hier eine einzige Möglichkeit aus einer vorher festgelegten und vermutlich nicht sehr großen Menge gewählt werden muss. Deswegen wurde „dropdown“ als Repräsentation separat eingetragen. Für die Zugriffsart des Atomars ist als Startwert „lesen“ festgelegt, da kein Standort ausgewählt werden kann, ohne, dass eine Kundennummer eingegeben worden ist. An diesem Punkt greift zum ersten Mal im Dokument eine Dependenz. Der Dependenzenkatalog ist ab Zeile 119 zu finden und das erste und zweite Element dort regeln die Änderung der Zugriffsart auf die Standortauswahl. Wenn der Inhalt des Elementes zur Kundennummer leer ist, besteht nur Lesezugriff auf die Standortauswahl. Ist er nicht leer, ist eine Neueingabe in die Auftragsdaten erlaubt. Der Begriff „nil“ steht dabei für einen leeren Eintrag und die Zeichen „=“ und „!=“ für die Vergleichsoperanden „gleich“ und „nicht gleich“. Da die linke Seite der Operanden leer gelassen wurde, bezieht sich der Vergleich auf den Inhalt des angegebenen Atomars. Andernfalls wird dort eine Bezeichnung angegeben, wie bei der jeweiligen Wirkung der beiden Regeln zu sehen ist.

Der nächste Block zeigt eine sehr ähnliche Struktur wie der vorhergehende, da es sich auch bei diesen Atomaren um Adressdaten handelt. Der Unterschied zwischen den beiden Blöcken liegt im ersten Atomar. Hier kann mittels einer boolschen Variable angegeben werden, ob die Rechnungsadresse gleich der Lieferadresse sein soll. Ist das Häkchen in der für diesen Fall vorgesehenen Checkbox gesetzt (Wert der boolschen Variable = 1), ist die erneute Eingabe einer Adresse unnötig und die Standortauswahl verbleibt im Lesezugriff, kann vom verarbeitenden Programm identisch mit der Lieferadresse in die Auftragsdaten geschrieben werden und weitere Adressdetails können ausgeblendet bleiben. Andernfalls (Wert der boolschen Variable = 0) muss die abweichende Standortauswahl durch die Änderung des Zugriffs erlaubt werden und die Detailfelder der Adresse werden angezeigt. Diese beiden Mechanismen sind in den Dependenzen 3 und 4 hinterlegt. Durch die Ein- und Ausblendung der Detailfelder zur Rechnungsadresse wird die Nutzung des nicht durch die Startinformationen vorgeschriebenen Attributs „sichtbar“ demonstriert.

Der als nächstes folgende Quelltextblock dient der Eingabe von Kontaktdaten des Ansprechpartners, der beim Kunden für den Auftrag zuständig ist. Auch hier wird wieder vorausgesetzt, dass für jeden Kunden eine feste Liste möglicher Ansprechpartner hinterlegt ist, auf die nun zurückgegriffen werden kann. Es greift das gleiche Prinzip wie bei der Lieferadresse: Nach Neueingabe eines eindeutig identifizierenden Atomars aus einem Dropdownmenü in den Auftrag werden die Details zur ausgewählten Person zur Überprüfung

der Eingabe mit Lesezugriff zur Verfügung gestellt. Besondere Aufmerksamkeit verdienen die in diesem Block verwendeten Datentypen „string.telefonnr“ und „string.email“, die am Ende der XML-Datei ab Zeile 189 zusammen mit den verwendeten Basisdatentypen definiert werden. Der Datentyp „string.telefonnr“ zeigt dort die Anwendung einer Positivliste, da in ihm nur wenige Zeichen werden dürfen, die vom Basisdatentyp string zur Verfügung gestellt werden. Die ausschließliche Verwendung der Ziffern 0 bis 9, sowie des Leerzeichens und der Sonderzeichen +, - und / erlaubt alle gängigen Notationsmöglichkeiten für Telefonnummern. Diese Vielfalt könnte auf Wunsch durch weitere Reduktion der erlaubten Zeichen und eine Syntaxregel eingeschränkt werden, um beispielsweise eine einheitliche Notation von Telefonnummern im gesamten System zu erzwingen. Ein Beispiel für eine Syntaxregel sieht man beim Datentyp „string.email“ angegeben. Sie wird ergänzt durch eine Negativliste. Die Definition entspricht der Umsetzung des Beispiels „Spezialdatentyp E-Mail“ aus Kapitel 4.4.

Im fünften Block des Quellcodes wird die Eingabe der gewünschten Lieferpositionen ermöglicht. Die jeweils zum Artikel passende Einheit kann mithilfe der Sachnummer ausgelesen werden, sodass diese nicht im Auftrag abgelegt werden muss. Der zu liefernden Menge ist der Datentyp „real.basis“ zugeordnet, um auch nicht diskrete Angaben, wie z.B. 2,3 Liter, zu ermöglichen. Erstmals kommen in diesem Block nicht nur inhaltliche Atomare, sondern auch Funktionen vor. Die erste Funktion dient dem Hinzufügen der gerade eingegebenen Daten zur Gesamtbestellung. Der dazugehhörige Button ist zunächst deaktiviert und wird erst dann durch die Dependenz 5 benutzbar, wenn in den drei Feldern „Sachnummer“, „Menge“ und „gewünschtes Lieferdatum“ ein Eintrag vorgenommen wurde. Die Dependenzen 6 bis 8 sorgen für die erneute Deaktivierung des Buttons, wenn eines der drei Felder leer ist. Die zweite Funktion erlaubt das Löschen einer bereits eingetragenen Position.

Der letzte Block beinhaltet die abschließende Funktion, welche die vollständige Bestellung verschickt. Sie wird ähnlich wie die Funktion zur Positionseingabe erst freigegeben, wenn bestimmte Felder ausgefüllt sind, und wieder deaktiviert, sobald eines der Felder unausgefüllt bleibt. Die Dependenzen 9 bis 13 regeln diese Aktivierungen je nach Zustand der Felder „Kundennummer“, „Lieferstandort“, „Rechnungsstandort“ und „Ansprechpartner“.

Die Instanz „Kundenbestellung“ der DTD ist damit vollständig beschrieben. Das Hinzufügen weiterer Instanzen ist innerhalb der gleichen Datei innerhalb des Wurzelelementes „System“ möglich. Im Rahmen dieser Arbeit wird davon abgesehen, eine weitere Instanz detailliert darzustellen, da mit dem ersten Beispiel alle Besonderheiten gezeigt und erläutert werden konnten. Eine weitere Instanz würde somit keinen inhaltlichen Mehrwert für die Arbeit bieten.

6 Fazit und Ausblick

Im Rahmen der vorliegenden Arbeit wurden aus einem Fallbeispiel im Umfeld des produzierenden Gewerbes Voraussetzungen und Regeln für eine Methodik entwickelt, die unabhängig vom Umgebungssystem zur automatisierten Erstellung von Formularen verwendet werden kann. Nach der Einführung ins Thema und der Vorstellung themenverwandter Arbeiten wurde ein ausgewählter Beispielprozess in geeigneter Form dargestellt und dann analysiert. Die Nutzungsmöglichkeiten von Formularen im Prozess wurden geprüft und die Zerlegung in ihre inhaltlichen Bestandteile wurde vorgenommen. Anschließend ergänzte die Untersuchung eines beliebigen inhaltsfreien Formulars die Menge der vorhandenen Einflussfaktoren. Anhand der in den beiden Teilen erarbeiteten Aspekte wurde dann geklärt, in welcher Art und Weise sich diese gegenseitig beeinflussen. Diese Beeinflussungsanalyse erlaubte es dann, die Anforderungen zu formulieren, welche die Methodik erfüllen muss. Die Methodik selbst wurde dann als Blackbox modelliert, an deren Ende als Ergebnis das funktionsfähige Formular stand. Nach der Aufzählung der benötigten Startinformationen gaben die Anweisungen zu ihrer Verarbeitung detaillierte Auskunft über die Regeln und Verarbeitungsanweisungen, welche die Methodik bilden und die Blackbox im Modell füllten. Um dem Anspruch der Maschinenverarbeitbarkeit der Ergebnisse nachzukommen, wurden die zuvor prosaisch beschriebenen Startinformationen mithilfe einer DTD strukturiert dargestellt. Die Ausarbeitung einer Instanz dieser DTD in Form einer XML-Datei, die ein Formular aus dem zu Anfang verwendeten Beispielprozess abbildet, verdeutlichte dann die Umsetzung eines festgelegten Parametersatzes und erlaubte eine Veranschaulichung der zuvor erarbeiteten Inhalte.

Da die Arbeit darauf ausgerichtet ist, grundlegende Überlegungen zur automatisierten Erstellung von Webformularen auszuarbeiten, bietet sich an ihrem Ende eine ganze Reihe von Anknüpfungspunkten an. Man kann drei Gebiete identifizieren, in denen die vorgestellten Ergebnisse als Grundlage für weitere Arbeiten genutzt werden können:

1. Annahme eines komplexeren Umgebungssystems: Hier ist es denkbar, die vorliegende Methodik um Komponenten zu erweitern, die ein Mehrbenutzersystem miteinbeziehen. Eine größere Menge an abzubildenden Prozessen und damit eine höhere Zahl zu erzeugender Formulare, die sich evtl. wiederholen, bietet interessante Ausarbeitungsmöglichkeiten. Außerdem ist die Nutzung der Ergebnisse in einem anderen Kontext als dem des betrieblichen Informationssystems im produzierenden Gewerbe gewünscht.

2. Gestaltungsaspekte der Formulare genauer betrachten: Der Entwurf von passenden Stylesheets und die grundsätzlichen Überlegungen, die vor seiner Erstellung anzustellen sind, können vorgenommen werden. Außerdem ist die Umsetzung von kumulierten, sich wiederholenden Atomaren, wie z.B. in einer Tabellenansicht nötig, noch nicht bedacht worden. Interessante Aspekte können sich hier insbesondere dann ergeben, wenn die zusammengesetzten Elemente in ihrer Größe variabel sind.

3. Details, die bei der Umsetzung der Methodik an einem realen System auffallen, ausarbeiten: Bisher existiert noch kein vollständiger Algorithmus, welcher die Methodik durchführt. Zu ihm würde z.B. auch die konkrete Ausformulierung eines Standardkataloges für die Repräsentationselemente gehören.

Insgesamt kann man feststellen, dass die Arbeit über ausreichend Anknüpfungspunkte verfügt, um zu gewährleisten, dass die erarbeiteten Inhalte weiter verwendet werden. Die verschiedenen Ausrichtungen der Weiterverarbeitungsmöglichkeiten erlauben, eine Vielzahl von Anschlussarbeiten mit individuell angepasster Ausrichtung anzufertigen.

Durch die Bearbeitung der vorliegenden Aufgabenstellung konnte eine Reihe von Erkenntnissen gewonnen werden, die nur teilweise im Vorfeld antizipiert worden waren. Das spezielle Ausgangssystem der Arbeit hat nicht von Beginn an vermuten lassen, dass die Ergebnisse der Arbeit so allgemeiner Natur sein würden, dass sie auch in anderen inhaltlichen Zusammenhängen genutzt werden können. Dieses Ergebnis überraschte positiv, da es dazu beiträgt, dass auf den Ergebnissen weiter aufgebaut werden kann, da es eine Vielzahl potenzieller Nutzer aus unterschiedlichen Bereichen gibt. Die Tatsache, dass das in der naturwissenschaftlichen und technischen Forschung immer wieder herangezogene Blackboxmodell auch in der vorliegenden Arbeit zur Lösung der Aufgabenstellung herangezogen werden konnte, war ebenfalls unerwartet. Die Einordnung der Aufgabenstellung in den Kontext themenverwandter Arbeiten zeigte sich schwierig, da eine große Zahl von Literaturquellen zu kommerziellen Lösungen für die Formularerzeugung die Identifikation von relevanten Quellen verkomplizierte. Die Beschäftigungsmöglichkeit mit der Auszeichnungssprache XML und ihren möglichen Anwendungen machte deutlich, wie wertvoll die breit angelegte Themenauswahl in der Anfangsphase des Maschinenbaustudiums ist, um als ausgebildeter Ingenieur an interdisziplinären Aufgabenstellungen mitzuarbeiten. Abschließend kann resümiert werden, dass die Bearbeitung der Aufgabenstellung eine spannende Gelegenheit war, Systemmodellierung und -analyse vor einem informatischen Hintergrund durchzuführen. Die Erarbeitung und verständliche Darstellung von grundsätzlichen Überlegungen zu diesem Thema, bei denen von vorneherein klar war, dass sie nicht vollständig sein würden und zwingend von anderen Personen weiter bearbeitet werden müssen, ruft das angenehme Gefühl hervor, einer sinnvollen und lohnenden Tätigkeit nachgegangen zu sein. Gleichzeitig war es möglich eine Vielzahl der Fertigkeiten zu nutzen, die im Laufe des Studiums vermittelt worden sind.

Literaturverzeichnis

[ABC+98] Appelrath, Hans-Jürgen; Boles, Dietrich; Claus, Volker; Wegener, Ingo: *Starthilfe Informatik*. Stuttgart, Leipzig: B.G. Teubner, 1998

[BF97] Bullinger, Hans-Jörg; Fähnrich, Klaus-Peter: *Betriebliche Informationssysteme*. Berlin, Heidelberg: Springer-Verlag, 1997

[Bic97] Bichler, Martin: *Aufbau unternehmensweiter WWW-Informationssysteme*. Braunschweig, Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaft mbH, 1997

[Cla06] Claus, Volker (Hrsg.): *Duden Informatik A-Z*. Mannheim, Wien, Zürich: Dudenverlag, 2006

[Dra05] Draheim, Dirk: *Form-Oriented Analysis*. Berlin, Heidelberg: Springer-Verlag, 2005

[Dud13] Dudenverlag, Bibliographisches Institut G.: *Duden*. Online im Internet. `http://www.duden.de/`. Version: 28.05.2013

[Eng03] Engelmann, Lutz (Hrsg.): *Basiswissen Schule Informatik Abitur*. Berlin: paetec Gesellschaft für Bildung und Technik mbH, 2003

[Fis11] Fischer, Peter; Hofer, Peter (Hrsg.): *Lexikon der Informatik*. Berlin, Heidelberg: Springer-Verlag, 2011

[Gro75] Grochla, Erwin: *Betriebliche Planung und Informationssysteme Entwicklung und aktuelle Aspekte*. Reinbek bei Hamburg: Rowohlt, 1975

[Mer12] Mertens, Peter; Bodendorf, Freimut (Hrsg.); König, Wolfgang (Hrsg.): *Grundzüge der Wirtschaftsinformatik*. 11. Auflage. Berlin, Heidelberg: Springer-Verlag, 2012

[Neu03] Neuschwinger, Andreas: *Multimediales, Informationsmodellbasiertes Arbeitsplatzkommunikationssystem*, Institut für Automatisierungstechnik, Lehrstuhl für Produktionssysteme, Ruhr-Universität Bochum, Diss., 2003

[Sch12] Schuh, Günther; Stich, Volker (Hrsg.): *Produktionsplanung und -steuerung 1*. 4. Auflage. Berlin, Heidelberg: Springer-Verlag, 2012

[SH02] Stahlknecht, Peter; Hasenkamp, Ulrich: *Einführung in die Wirtschaftsinformatik*. 10. Auflage. Berlin, Heidelberg: Springer-Verlag, 2002

[SMJ93] Spur, Günter; Mertins, Kai; Jochem, Roland; Warnecke, Hans-Jürgen (Hrsg.); Schuster, Richard (Hrsg.): *Integrierte Unternehmensmodellierung*. Berlin, Wien, Zürich: Beuth Verlag GmbH, 1993

[Spr13] SpringerGabler (Hrsg.): *Gabler Kompakt-Lexikon Wirtschaft.* 11. Auflage. Wiesbaden: Springer Fachmedien Wiesbaden, 2013

[Von07] Vonhoegen, Helmut: *Einstieg in XML.* 4. Auflage. Bonn: Galileo Press, 2007

[Web12] Weber, Rainer: *Technologie von Unternehmenssoftware.* Berlin, Heidelberg: Springer-Verlag, 2012

[WHB12] Wagner, Klaus-P.; Hüttl, Thomas; Backin, Dieter; Vieweg, Iris (Hrsg.); Werner, Christian (Hrsg.): *Einführung Wirtschaftsinformatik.* Gabler Verlag, 2012

[Zeh98] Zehnder, Carl A.: *Informationssysteme und Datenbanken.* 6. Auflage. Stuttgart: B.G. Teubner, 1998

Abkürzungsverzeichnis

EDV	Elektronische Datenverarbeitung
ERP	Enterprise Ressource Planning, Planung von Unternehmensressourcen
GUI	Graphical User Interface, Grafische Benutzeroberfläche
IUM	integrierte Unternehmensmodellierung
XML	Extensible Markup Language, Erweiterbare Auszeichnungssprache

A Abbildungen

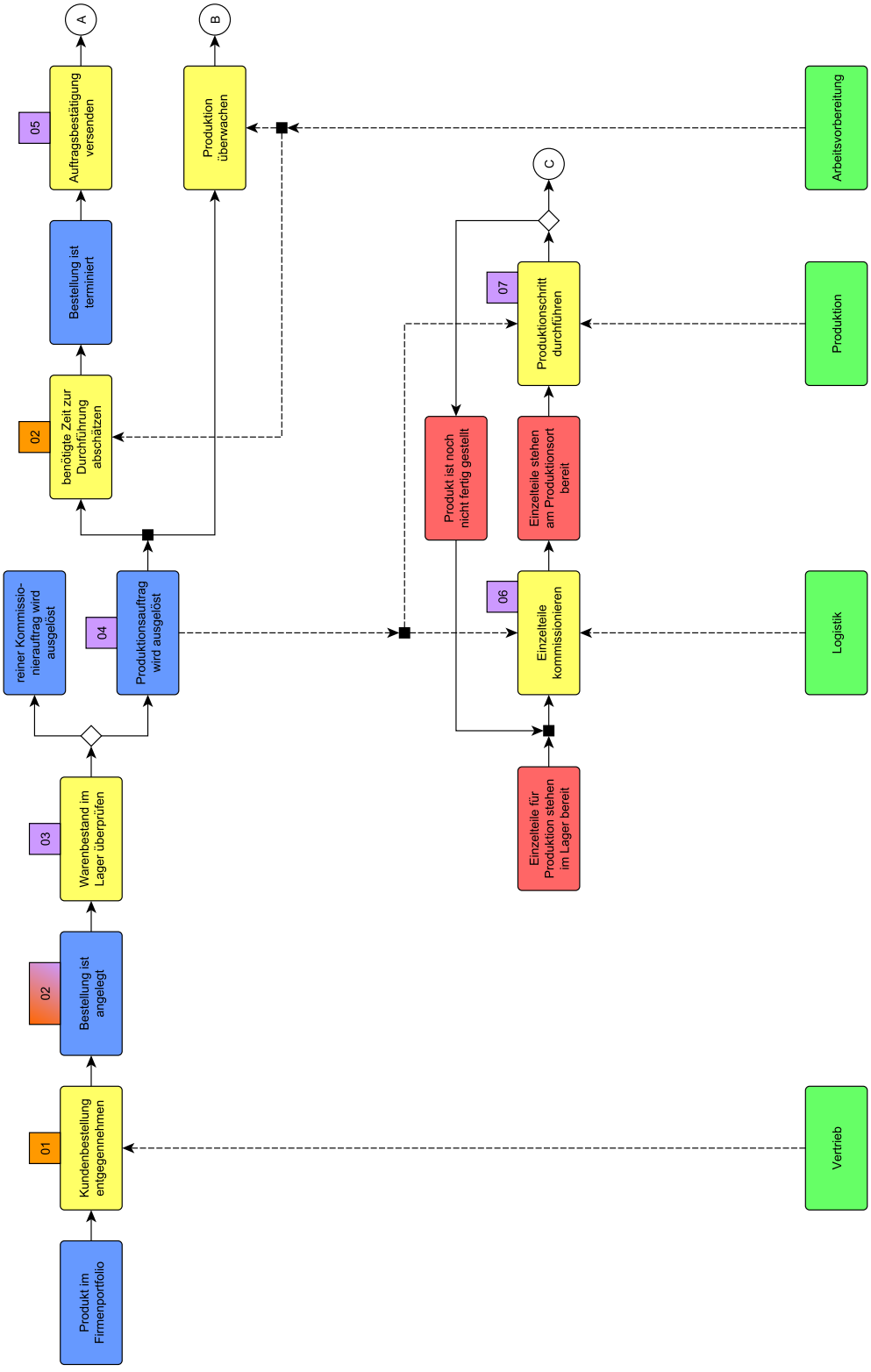

Abb. A.1.: Produktionsauslösender Bestellprozess, Teil1

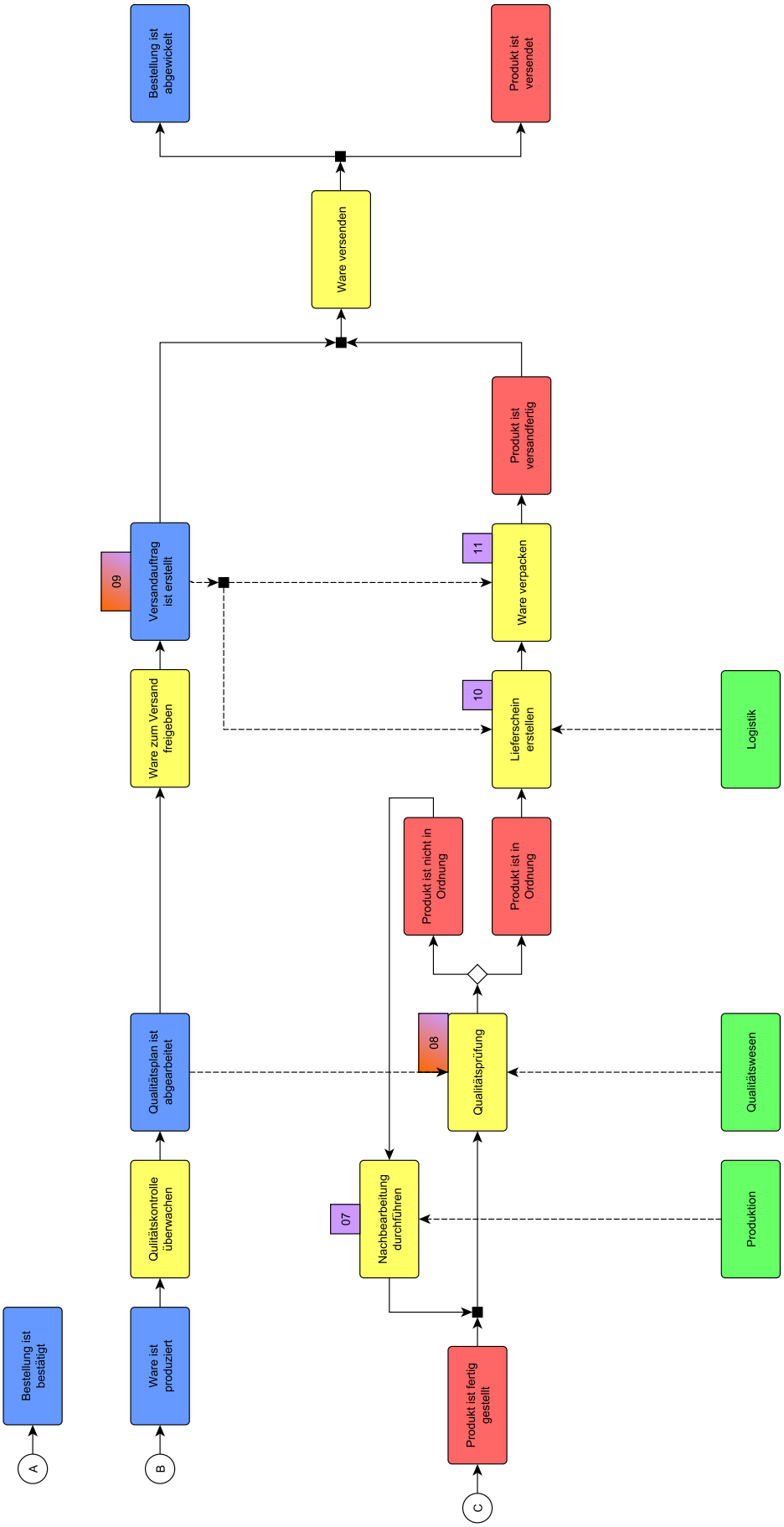

Abb. A.2.: Produktionsauslösender Bestellprozess, Teil2

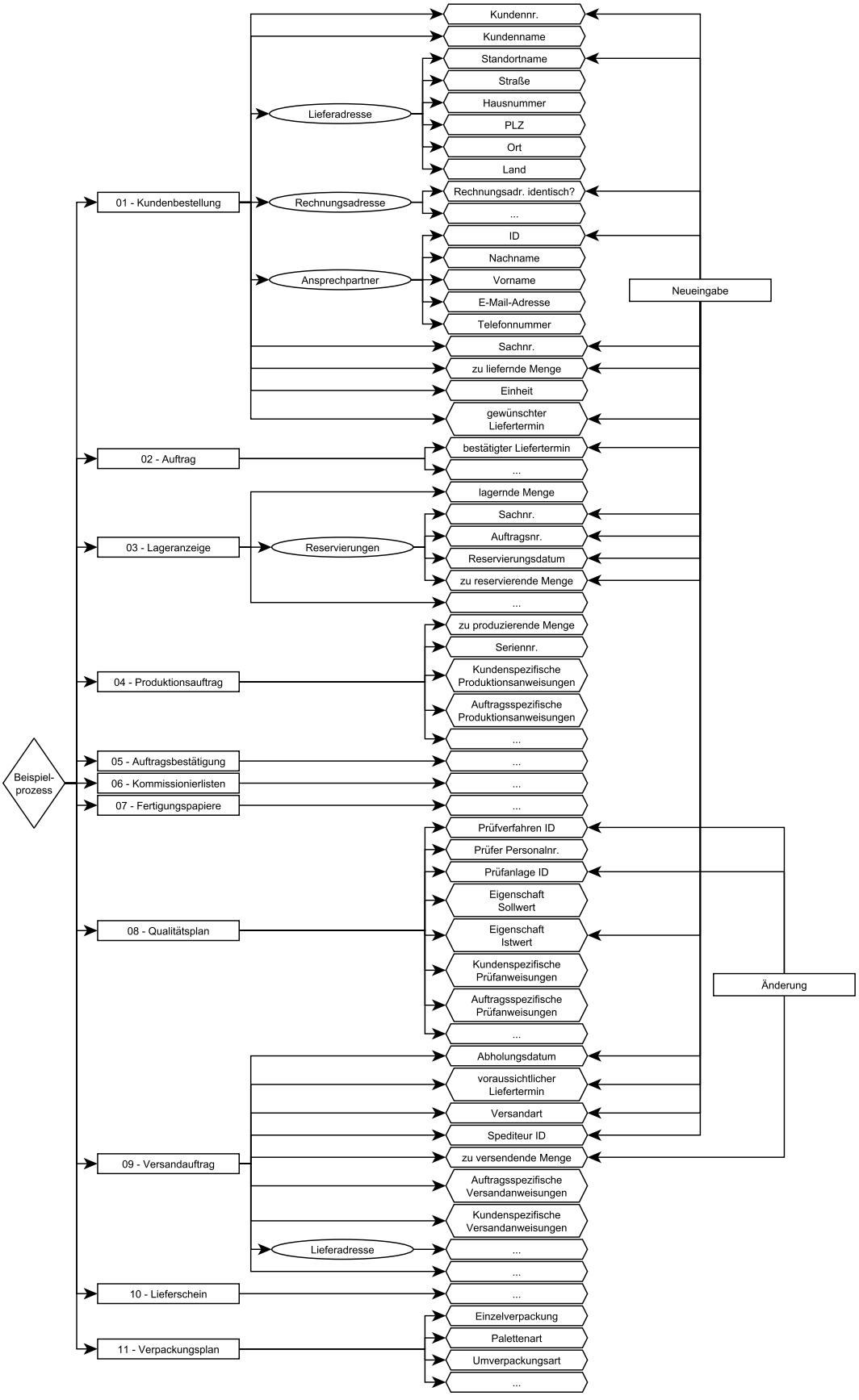

Abb. A.3.: Zuordnung der Formularnummern zu den Formularnamen und -inhalten

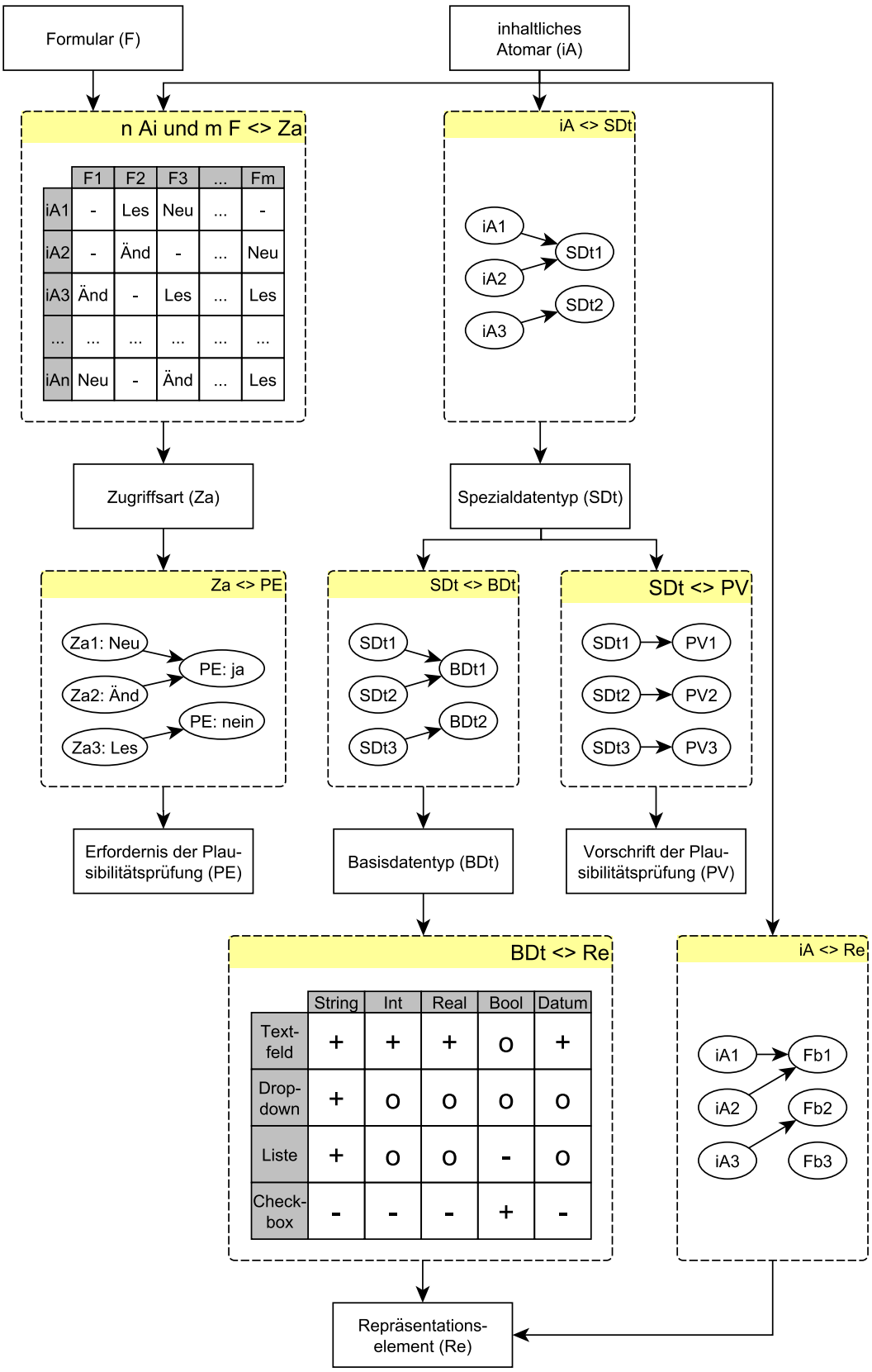

Abb. A.4.: Funktionale Einflüsse der verschiedenen Formularinhalte und -bestandteile aufeinander

B Quelltext

B.1 DTD

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <!ELEMENT system                ((formular)+, (spezialdatentyp)+)>
 3  <!ELEMENT formular             ((inhaltlichesAtomar|funktion)+,
                                                    (dependenz)*)>
 4    <!ATTLIST formular
 5    name                        ID                          #REQUIRED>
 6    <!ELEMENT inhaltlichesAtomar (inhalt, label)>
 7      <!ATTLIST inhaltlichesAtomar
 8      name                      ID                          #REQUIRED
 9      sichtbar                  (ja|nein)                   #REQUIRED
10      zugriff                   (lesen|änderung|neueingabe) #REQUIRED
11      repraesentation           (textbox|dropdown|liste|checkbox)
                                                            #IMPLIED
12      datentyp                  IDREF                       #REQUIRED>
13      <!ELEMENT inhalt          (#PCDATA)>
14      <!ELEMENT label           (#PCDATA)>
15    <!ELEMENT funktion          (inhalt, label)>
16      <!ATTLIST funktion
17      name                      ID                          #REQUIRED
18      sichtbar                  (ja|nein)                   #REQUIRED
19      repraesentation           (button)                    #IMPLIED
20      benutzbar                 (ja|nein)                   #REQUIRED>
21      <!ELEMENT inhalt          (#PCDATA)>
22      <!ELEMENT label           (#PCDATA)>
23    <!ELEMENT dependenz         ((ursache)+, (wirkung)+)>
24      <!ATTLIST dependenz
25      id                        ID                          #REQUIRED>
26      <!ELEMENT ursache         (#PCDATA)>
27        <!ATTLIST ursache
28        ziel                    IDREF                       #REQUIRED>
29      <!ELEMENT wirkung         (#PCDATA)>
30        <!ATTLIST wirkung
31        ziel                    IDREF                       #REQUIRED>
32  <!ELEMENT spezialdatentyp     ((syntaxregel)?,(positivliste|
                                                  negativliste)?)>
33    <!ATTLIST spezialdatentyp
34    name                        ID                          #REQUIRED
```

```
35    basistyp                    (string|int|real|boolean|datum)
                                                          #REQUIRED>
36    <!ELEMENT syntaxregel       (#PCDATA)>
37    <!ELEMENT positivliste      ((eintrag)+)>
38      <!ELEMENT eintrag         (#PCDATA)>
39    <!ELEMENT negativliste      ((eintrag)+)>
40      <!ELEMENT eintrag         (#PCDATA)>
```

B.2 Formular Kundebestellung

```
 1 <?xml version="1.0" encoding="ISO-8859-1"?>
 2 <!DOCTYPE System SYSTEM "Formular.dtd" >
 3 <system>
 4   <formular name="Kundenbestellung">
 5     <inhaltlichesAtomar name="iAkundennr" sichtbar="ja" zugriff=
                                    "neueingabe" datentyp="int.basis">
 6       <inhalt>auftrag.kundennr</inhalt>
 7       <label>Kundennr.</label>
 8     </inhaltlichesAtomar>
 9   <inhaltlichesAtomar name="iAkundenname" sichtbar="ja" zugriff=
                                      "lesen" datentyp="string.basis">
10     <inhalt>kundenstammdaten.name</inhalt>
11     <label>Kunde</label>
12   </inhaltlichesAtomar>
13
14   <inhaltlichesAtomar name="iAlieferstandortname" sichtbar="ja"
      zugriff="lesen" repraesentation="dropdown" datentyp="string.basis">
15     <inhalt>auftrag.lieferstandortname</inhalt>
16     <label>Lieferstandort</label>
17   </inhaltlichesAtomar>
18   <inhaltlichesAtomar name="iAlieferstandortstrasse" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
19     <inhalt>standorte.strasse</inhalt>
20     <label>Straße</label>
21   </inhaltlichesAtomar>
22   <inhaltlichesAtomar name="iAlieferstandorthausnr" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
23     <inhalt>standorte.hausnr</inhalt>
24     <label>Hausnr.</label>
25   </inhaltlichesAtomar>
26   <inhaltlichesAtomar name="iAlieferstandortplz" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
27     <inhalt>standorte.plz</inhalt>
```

```
28      <label>PLZ</label>
29    </inhaltlichesAtomar>
30    <inhaltlichesAtomar name="iAlieferstandortort" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
31      <inhalt>standorte.ort</inhalt>
32      <label>Ort</label>
33    </inhaltlichesAtomar>
34    <inhaltlichesAtomar name="iAlieferstandortland" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
35      <inhalt>standorte.land</inhalt>
36      <label>Land</label>
37    </inhaltlichesAtomar>
38
39    <inhaltlichesAtomar name="iArechnungsadresseidentisch" sichtbar="ja"
                            zugriff="neueingabe" datentyp="boolean.basis">
40      <inhalt>auftrag.rechnungsadresseidentisch</inhalt>
41      <label>Rechnungsadresse und Lieferadresse identisch?</label>
42    </inhaltlichesAtomar>
43    <inhaltlichesAtomar name="iArechnungsstandortname" sichtbar="ja"
                           zugriff="neueingabe" repraesentation="dropdown"
                                                  datentyp="string.basis">
44      <inhalt>auftrag.rechnungsstandortname</inhalt>
45      <label>Rechnungsstandort</label>
46    </inhaltlichesAtomar>
47    <inhaltlichesAtomar name="iArechnungsstandortstrasse" sichtbar=
                           "nein" zugriff="lesen" datentyp="string.basis">
48      <inhalt>standorte.strasse</inhalt>
49      <label>Straße</label>
50    </inhaltlichesAtomar>
51    <inhaltlichesAtomar name="iArechnungsstandorthausnr" sichtbar="nein"
                                zugriff="lesen" datentyp="string.basis">
52      <inhalt>standorte.hausnr</inhalt>
53      <label>Hausnr.</label>
54    </inhaltlichesAtomar>
55    <inhaltlichesAtomar name="iArechnungsstandortplz" sichtbar="nein"
                                zugriff="lesen" datentyp="string.basis">
56      <inhalt>standorte.plz</inhalt>
57      <label>PLZ</label>
58    </inhaltlichesAtomar>
59    <inhaltlichesAtomar name="iArechnungsstandortort" sichtbar="nein"
                                zugriff="lesen" datentyp="string.basis">
60      <inhalt>standorte.ort</inhalt>
61      <label>Ort</label>
```

```
62   </inhaltlichesAtomar>
63   <inhaltlichesAtomar name="iArechnungsstandortland" sichtbar="nein"
                                zugriff="lesen" datentyp="string.basis">
64     <inhalt>standorte.land</inhalt>
65     <label>Land</label>
66   </inhaltlichesAtomar>
67
68   <inhaltlichesAtomar name="iAansprechpartnerid" sichtbar="ja"
                         zugriff="neueingabe" repraesentation="dropdown"
                                              datentyp="string.basis">
69     <inhalt>auftrag.ansprechpartner</inhalt>
70     <label>Ansprechpartner</label>
71   </inhaltlichesAtomar>
72   <inhaltlichesAtomar name="iAansprechpartnernachname" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
73     <inhalt>ansprechpartner.nachname</inhalt>
74     <label>Nachname</label>
75   </inhaltlichesAtomar>
76   <inhaltlichesAtomar name="iAansprechpartnervorname" sichtbar="ja"
                                zugriff="lesen" datentyp="string.basis">
77     <inhalt>ansprechpartner.vorname</inhalt>
78     <label>Vorname</label>
79   </inhaltlichesAtomar>
80   <inhaltlichesAtomar name="iAansprechpartnertelefon" sichtbar="ja"
                            zugriff="lesen" datentyp="string.telefonnr">
81     <inhalt>ansprechpartner.telefon</inhalt>
82     <label>Telefonnr.</label>
83   </inhaltlichesAtomar>
84   <inhaltlichesAtomar name="iAansprechpartneremail" sichtbar="ja"
                                zugriff="lesen" datentyp="string.email">
85     <inhalt>ansprechpartner.email</inhalt>
86     <label>Land</label>
87   </inhaltlichesAtomar>
88
89   <inhaltlichesAtomar name="iAneusachnr" sichtbar="ja" zugriff=
                                    "neueingabe" datentyp="string.basis">
90     <inhalt>lieferwaren.sachnr</inhalt>
91     <label>Sachnr.</label>
92   </inhaltlichesAtomar>
93   <inhaltlichesAtomar name="iAneuliefermenge" sichtbar="ja" zugriff=
                                      "neueingabe" datentyp="real.basis">
94     <inhalt>lieferwaren.liefermenge</inhalt>
95     <label>zu liefernde Menge</label>
```

```
 96   </inhaltlichesAtomar>
 97   <inhaltlichesAtomar name="iAneuliefereinheit" sichtbar="ja" zugriff=
                                          "lesen" datentyp="string.basis">
 98     <inhalt>lieferwaren.einheit</inhalt>
 99     <label>Einheit</label>
100   </inhaltlichesAtomar>
101   <inhaltlichesAtomar name="iAneulieferterminwunsch" sichtbar="ja"
                            zugriff="neueingabe" datentyp="datum.basis">
102     <inhalt>lieferwaren.lieferterminwunsch</inhalt>
103     <label>gewünschter Liefertermin</label>
104   </inhaltlichesAtomar>
105   <funktion name="fktPositionhinzufügen" sichtbar="ja"
                             repraesentation="button" benutzbar="nein" >
106     <inhalt>fkt.Positionhinzufügen</inhalt>
107      <label>diese Lieferposition hinzufügen</label>
108   </funktion>
109   <funktion name="fktPositionlöschen" sichtbar="ja" repraesentation=
                                               "button" benutzbar="ja" >
110     <inhalt>fkt.Positionlöschen</inhalt>
111     <label>bereits eingegebene Lieferposition löschen</label>
112   </funktion>
113
114   <funktion name="fktbestellungversenden" sichtbar="ja"
                             repraesentation="button" benutzbar="nein" >
115     <inhalt>fkt.bestellungversenden</inhalt>
116     <label>Bestellung versenden</label>
117   </funktion>
118
119   <dependenz id="regel1">
120     <ursache ziel="iAkundennr">=nil</ursache>
121     <wirkung ziel="iAlieferstandortname">attr.zugriff="lesen"
                                                          </wirkung>
122   </dependenz>
123   <dependenz id="regel2">
124     <ursache ziel="iAkundennr">!=nil</ursache>
125     <wirkung ziel="iAlieferstandortname">attr.zugriff="neueingabe"
                                                          </wirkung>
126   </dependenz>
127   <dependenz id="regel3">
128     <ursache ziel="iArechnungsadresseidentisch">=1</ursache>
129     <wirkung ziel="iArechnungsstandortname">attr.zugriff="lesen"
                                                          </wirkung>
130     <wirkung ziel="iArechnungsstandortstrasse">attr.sichtbar="nein"
```

```
                                                                    </wirkung>
131     <wirkung ziel="iArechnungsstandorthausnr">attr.sichtbar="nein"
                                                                    </wirkung>
132     <wirkung ziel="iArechnungsstandortplz">attr.sichtbar="nein"
                                                                    </wirkung>
133     <wirkung ziel="iArechnungsstandortort">attr.sichtbar="nein"
                                                                    </wirkung>
134     <wirkung ziel="iArechnungsstandortland">attr.sichtbar="nein"
                                                                    </wirkung>
135   </dependenz>
136   <dependenz id="regel4">
137     <ursache ziel="iArechnungsadresseidentisch">=0</ursache>
138     <wirkung ziel="iArechnungsstandortname">attr.zugriff="neueingabe"
                                                                    </wirkung>
139     <wirkung ziel="iArechnungsstandortstrasse">attr.sichtbar="ja"
                                                                    </wirkung>
140     <wirkung ziel="iArechnungsstandorthausnr">attr.sichtbar="ja"
                                                                    </wirkung>
141     <wirkung ziel="iArechnungsstandortplz">attr.sichtbar="ja"
                                                                    </wirkung>
142     <wirkung ziel="iArechnungsstandortort">attr.sichtbar="ja"
                                                                    </wirkung>
143     <wirkung ziel="iArechnungsstandortland">attr.sichtbar="ja"
                                                                    </wirkung>
144   </dependenz>
145   <dependenz id="regel5">
146     <ursache ziel="iAneusachnr">!=nil</ursache>
147     <ursache ziel="iAneuliefermenge">!=nil</ursache>
148     <ursache ziel="iAneulieferterminwunsch">!=nil</ursache>
149     <wirkung ziel="fktPositionhinzufügen">attr.benutzbar="ja"
                                                                    </wirkung>
150   </dependenz>
151   <dependenz id="regel6">
152     <ursache ziel="iAneusachnr">=nil</ursache>
153     <wirkung ziel="fktPositionhinzufügen">attr.benutzbar="nein"
                                                                    </wirkung>
154   </dependenz>
155   <dependenz id="regel7">
156     <ursache ziel="iAneuliefermenge">=nil</ursache>
157     <wirkung ziel="fktPositionhinzufügen">attr.benutzbar="nein"
                                                                    </wirkung>
158   </dependenz>
159   <dependenz id="regel8">
```

```
160     <ursache ziel="iAneulieferterminwunsch">=nil</ursache>
161     <wirkung ziel="fktPositionhinzufügen">attr.benutzbar="nein"
                                                              </wirkung>
162   </dependenz>
163   <dependenz id="regel9">
164     <ursache ziel="iAkundennr">!=nil</ursache>
165     <ursache ziel="iAlieferstandortname">!=nil</ursache>
166     <ursache ziel="iArechnungsstandortname">!=nil</ursache>
167     <ursache ziel="iAansprechpartnerid">!=nil</ursache>
168     <wirkung ziel="fktbestellungversenden">attr.benutzbar="ja"
                                                              </wirkung>
169   </dependenz>
170   <dependenz id="regel10">
171     <ursache ziel="iAkundennr">=nil</ursache>
172     <wirkung ziel="fktbestellungversenden">attr.benutzbar="nein"
                                                              </wirkung>
173   </dependenz>
174   <dependenz id="regel11">
175     <ursache ziel="iAlieferstandortname">=nil</ursache>
176     <wirkung ziel="fktbestellungversenden">attr.benutzbar="nein"
                                                              </wirkung>
177   </dependenz>
178   <dependenz id="regel12">
179     <ursache ziel="iArechnungsstandortname">=nil</ursache>
180     <wirkung ziel="fktbestellungversenden">attr.benutzbar="nein"
                                                              </wirkung>
181   </dependenz>
182   <dependenz id="regel13">
183     <ursache ziel="iAansprechpartnerid">=nil</ursache>
184     <wirkung ziel="fktbestellungversenden">attr.benutzbar="nein"
                                                              </wirkung>
185   </dependenz>
186 </formular>
187
188 <spezialdatentyp name="string.basis" basistyp="string">
                                                    </spezialdatentyp>
189 <spezialdatentyp name="string.telefonnr" basistyp="string">
190   <positivliste>
191     <eintrag>0</eintrag>
192     <eintrag>1</eintrag>
193     <eintrag>2</eintrag>
194     <eintrag>3</eintrag>
195     <eintrag>4</eintrag>
```

```
196     <eintrag>5</eintrag>
197     <eintrag>6</eintrag>
198     <eintrag>7</eintrag>
199     <eintrag>8</eintrag>
200     <eintrag>9</eintrag>
201     <eintrag>+</eintrag>
202     <eintrag>-</eintrag>
203     <eintrag>/</eintrag>
204     <eintrag> </eintrag>
205   </positivliste>
206 </spezialdatentyp>
207 <spezialdatentyp name="string.email" basistyp="string">
208   <syntaxregel>?*@?*.??*</syntaxregel>
209   <negativliste>
210     <eintrag>@</eintrag>
211   </negativliste>
212 </spezialdatentyp>
213 <spezialdatentyp name="int.basis" basistyp="int"></spezialdatentyp>
214 <spezialdatentyp name="real.basis" basistyp="real"></spezialdatentyp>
215 <spezialdatentyp name="datum.basis" basistyp="datum">
                                                    </spezialdatentyp>
216 <spezialdatentyp name="boolean.basis" basistyp="boolean">
                                                    </spezialdatentyp>
217 </system>
```

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum	Unterschrift