

# Workflow-Engineering in der Praxis

Prüfung derzeitiger Technologien und Standards für den Einsatz in  
Kleinunternehmen

Technische Universität Graz, Austria



## **MASTERARBEIT**

Vedran Bratić, Bakk.rer.soc.oec.

27. April 2010

Betreuer:

Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Denis Helić

(IICM) Institut für Informationssysteme und  
Computer Medien

## **Kurzfassung**

Die Fahrzeugdiagnose während der Entwicklung ist für das Unternehmen BMW immer wichtiger. Die schnelle und fehlerfreie Kommunikation mit dem Motorsteuergerät, welches als das Gehirn des Fahrzeugs fungiert, ist von essentieller Bedeutung. Diese Arbeit soll mögliche Lösungsansätze prüfen, welche dazu verhelfen sollen, die derzeit recht einfachen Anfragen an das Steuergerät zu komplexerer Diagnoseprozessen zu verbinden. Das Augenmerk wird hierbei an die Standards und den technischen Fortschritt im Bereich der Workflow-Management-Systeme geworfen.

## **Abstract**

The diagnostic of vehicles during their development gains more and more in importance in the company BMW. The communication with the electronic control unit, which acts like the brain of the vehicle, has to be fast and precise. This work takes a look on approaches, which should help by combining the rather simple request-response communication to more complex diagnostic workflows. The attention hereby will be on standards and technology improvements in the field of Workflow Management Systems.

## Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am .....

## **Danksagung**

Diese Masterarbeit ist mit Hilfe einiger Personen erstellt worden und ich nutze hier die Gelegenheit, um ihnen zu danken. Meinem Betreuer danke ich für die Möglichkeit an diesem Thema zu arbeiten und für die tatkräftige Unterstützung zu jeder Zeit. Die Zusammenarbeit mit der Firma ASE war in den letzten Monaten außerordentlich interessant und hat ausgezeichnet funktioniert, wofür ich sehr dankbar bin.

Allen voran gebührt ein großer Dank meinen Eltern und Brüdern, sowie meiner Frau. Während meiner Ausbildungsjahre haben sie kreative Wege gefunden mich zu motivieren, wenn es mal nicht so gut ging.

# Inhaltsverzeichnis

<b>ABBILDUNGSVERZEICHNIS .....</b>	<b>7</b>
<b>TABELLENVERZEICHNIS .....</b>	<b>9</b>
<b>1 EINFÜHRUNG .....</b>	<b>10</b>
1.1 MOTIVATION .....	10
1.2 AUFBAU DER ARBEIT .....	11
<b>2 SYSTEMANALYSE UND ANFORDERUNG .....</b>	<b>12</b>
2.1 DIE FAHRZEUGDIAGNOSE IM ALLGEMEINEN .....	12
2.2 BISHERIGE ERFAHRUNGEN UND LÖSUNGSANSÄTZE .....	14
2.2.1 <i>Labeldatenbank</i> .....	15
2.2.2 <i>Diagnose Parameter Assistent</i> .....	15
2.3 DIAGNOSE MIT DIAACCESS .....	20
2.3.1 <i>Funktionsweise und Anwendung</i> .....	20
2.3.2 <i>Typische Diagnoseprozesse</i> .....	21
2.3.3 <i>Die Analyse der gesammelten Erfahrungen</i> .....	26
2.4 ZUSAMMENFASSUNG .....	26
<b>3 DAS ENTWICKELN VON ABLÄUFEN .....</b>	<b>28</b>
3.1 WORKFLOW MANAGEMENT SYSTEME .....	28
3.1.1 <i>Elemente, Begriffe und Definitionen</i> .....	28
3.1.2 <i>Organisationen zur Standardisierung und Entwicklung der WfMS-Technologien</i> .....	31
3.2 ABLAUFMODELLIERUNG .....	34
3.2.1 <i>Event-driven Process Chain (EPC)</i> .....	34
3.2.2 <i>Extended Event-driven Process Chain (eEPC)</i> .....	38
3.2.3 <i>Business Process Modeling Notation (BPMN)</i> .....	40
3.3 ABLAUFBESCHREIBUNG BZW. -DEFINITION .....	48
3.3.1 <i>Electronic Business using XML (ebXML)</i> .....	49
3.3.2 <i>XML Process Definition Language (XPDL)</i> .....	53
3.3.3 <i>Business Process Execution Language (BPEL)</i> .....	59
3.3.4 <i>Event-driven Process Chain Markup Language (EPML)</i> .....	66
3.4 ZUSAMMENFASSUNG .....	72
<b>4 WORKFLOW MANAGEMENT SYSTEME IN DER PRAXIS .....</b>	<b>73</b>
4.1 IMPLEMENTIERUNGEN UND PROJEKTE .....	73
4.1.1 <i>EPC und eEPC Projekte</i> .....	73
4.1.2 <i>BPMN Projekte</i> .....	74
4.1.3 <i>EPML Projekte</i> .....	75
4.1.4 <i>BPEL Projekte</i> .....	75

4.1.5	<i>XPDL Projekte</i> .....	76
4.1.6	<i>Übersetzungs- und Transformationstools</i> .....	78
4.2	FAZIT DER RECHERCHE .....	78
4.2.1	<i>Komponenten einer möglichen Lösung</i> .....	79
4.2.2	<i>Gegenüberstellung der Lösungen</i> .....	79
<b>5</b>	<b>WASAKA – DIE NEUE DIAGNOSESOFTWARE</b> .....	<b>81</b>
5.1	KOMPONENTEN DER DIAGNOSE.....	81
5.2	DIE FUNKTIONALITÄT .....	83
5.2.1	<i>Das Lesen des Fehlerspeichers</i> .....	84
5.2.2	<i>Die Durchführen von Messungen</i> .....	85
5.2.3	<i>Weitere Diagnosemodule</i> .....	86
5.3	ARCHITEKTUR UND AUFBAU .....	88
5.4	DIE ENTWICKLUNGSPHASE DER ABLÄUFE.....	89
5.5	ABLÄUFE IM WASAKA.....	89
5.5.1	<i>Erweiterung durch Skriptsprachen</i> .....	90
5.5.2	<i>Erweiterung durch Konfiguration und Codegenerierung</i> .....	90
5.5.3	<i>Erweiterung durch Plug-In Codierung</i> .....	92
5.6	ZUSAMMENFASSUNG UND AUSBLICK .....	94
<b>6</b>	<b>LITERATURVERZEICHNIS</b> .....	<b>96</b>

## Abbildungsverzeichnis

---

Abbildung 1: SGBD Explorer - IDENT Job .....	13
Abbildung 2: SGBD Explorer – Tabelle der vom Steuergerät unterstützten Messwerte.....	14
Abbildung 3: Diagnose Parameter Assistent – Liste aller Konfigurationen .....	16
Abbildung 4: Diagnose Parameter Assistent - Parametrierung (Ein EDIABAS und ein MCD-3 Projekt).17	
Abbildung 5: Diagnose Parameter Assistent - Tabellen einer Steuergerätedatei.....	19
Abbildung 6: DiaAccess - Einstiegsformular .....	21
Abbildung 7: DiaAccess - Darstellung des Fehlerspeichers .....	23
Abbildung 8: DiaAccess – Dialog zum Messen .....	24
Abbildung 9: DiaAccess - NMK Schnelllernen .....	25
Abbildung 10: WfMS – Beziehungen der Basisterminologie (Workflow Management Coalition, 1999) 29	
Abbildung 11: WfMS – Zeitliche Entwicklung der Standards (Bartonitz, BPEL - Business Process Execution Language, 2007) .....	32
Abbildung 12: EPC – Anwendung der Notation an einem einfachen Beispiel (Day, Bonati, Büttiker, & Lee, 2006).....	35
Abbildung 13: EPC – Anwendungsbeispiele für Operatoren (Day, Bonati, Büttiker, & Lee, 2006).....	37
Abbildung 14: eEPC – Das einfache Beispiel der EPC aus Abbildung 12 erweitert um Elemente der eEPC (Day, Bonati, Büttiker, & Lee, 2006) .....	38
Abbildung 15: BPMN – Ablaufobjekte.....	41
Abbildung 16: BPMN – Ein einfaches BPMN Beispiel (Owen & Raj, 2003) .....	45
Abbildung 17: BPMN – Verantwortlichkeitsbereiche .....	46
Abbildung 18: ebXML – Komponentenarchitektur (Europäisches ebXML-Informationszentrum, 2009)51	
Abbildung 19: ebXML – Anwendungsszenario (Europäisches ebXML-Informationszentrum, 2009) .....	52
Abbildung 20: XPDL – Package Meta-Model (Workflow Management Coalition, 2008) .....	54
Abbildung 21: XPDL – Prozess Meta-Modell.....	56
Abbildung 22: WS-BPEL – Ebenen der Internet-Geschäftsprozesse (Nawrot, 2007) .....	60
Abbildung 23: WS-BPEL – Prozessdefinition (Stapf, 2005) .....	61
Abbildung 24: WS-BPEL – PartnerLinks (Bachmann, 2009) .....	62

**Abbildung 25: WS-BPEL – Variablen (Bachmann, 2009)..... 62**

**Abbildung 26: WS-BPEL – Prozessdefinition (Bachmann, 2009) ..... 65**

**Abbildung 27: EPML – Design Prinzipien (Mendling & Nüttgens, 2004)..... 67**

**Abbildung 28: EPML – Elemente (Mendling & Nüttgens, 2005)..... 68**

**Abbildung 29: EPML – Flache EPK mit EPML Repräsentation (Mendling & Nüttgens, EPC Markup  
Language (EPML), 2005) ..... 70**

**Abbildung 30: EPML – Hierarchische EPK mit EPML Repräsentation ..... 71**

**Abbildung 31: Wasaka – Zusammenspiel der Diagnosekomponenten ..... 82**

**Abbildung 32: Wasaka – Nach erfolgreicher Initialisierung des Steuergerätes ..... 84**

**Abbildung 33: Wasaka – Fehlerspeicher Modul ..... 85**

**Abbildung 34: Wasaka - Modul zum Lesen der Messwerte ..... 86**

**Abbildung 35: Wasaka - Steller stellen ..... 87**

**Abbildung 36: Wasaka - Modul EEPROM Abgleich ..... 87**

**Abbildung 37: Wasaka – Architektur der Lösung..... 88**

**Abbildung 38: Wasaka – Beispielkonfiguration eines Diagnoseablaufs ..... 91**

**Abbildung 39: Wasaka – XML Definition des Dialogs Fehlerspeicher ..... 92**

**Abbildung 40: Wasaka – Minimalles Skelett eines Plug-Ins ..... 93**



## Tabellenverzeichnis

---

Tabelle 1: Fahrzeugdiagnose – Beispieltelegramme zur Kommunikation mit dem Steuergerät.....	12
Tabelle 2: BPMN – Ereignisauslöser (Owen & Raj, 2003) .....	43
Tabelle 3: BPMN – Gateways (Owen & Raj, 2003).....	44
Tabelle 4: BPMN – Verbindungselemente .....	45
Tabelle 5: BPMN – Artefakte (White, Introduction to BPMN, 2004) .....	47
Tabelle 6: XPDL – Übersicht der Attribute der wichtigsten XPDL-Entitäten (Bartonitz, XPDL - XML Process Definition Language, 2007) .....	59
Tabelle 7: WS-BPEL – Elementare Aktivitäten (WSBPEL Technical Committee, 2007).....	64
Tabelle 8: WS-BPEL – Strukturierte Aktivitäten (WSBPEL Technical Committee, 2007) .....	64
Tabelle 9: ASE Kriterien vs. Workflow Management Systeme .....	80

# 1 Einführung

---

Diese Arbeit wurde in Kooperation mit der Firma ASE GmbH durchgeführt. Die Firma ASE ist ein langjähriger Softwarelieferant für das Unternehmen BMW. Die letzten beiden Softwareprodukte beziehen sich auf die Fahrzeugdiagnose während der Entwicklung neuer Motoren, welche im Hause BMW immer mehr an Bedeutung gewinnt.

Es handelt sich dabei um Daten, welche vom Steuergerät des Fahrzeugs gelesen bzw. in das Steuergerät geschrieben werden. Die Kommunikation mit dem Steuergerät geschieht dabei über ein Diagnosesystem. Das Diagnosesystem, welches vom Unternehmen BMW genutzt wird, heißt EDIABAS. Um die Arbeit mit dem Steuergerät und dem Diagnosesystem zu verbessern, wird ein neues Diagnosetool von der Firma ASE entwickelt. Einer der wesentlichen Merkmale dieses Tools ist die Zusammenfassung einzelner Kommunikationsvorgänge mit dem Steuergerät zu komplexeren Diagnoseprozessen.

Das Augenmerk dieser Arbeit wird also an die Standards und den technischen Fortschritt im Bereich der Workflow-Management-Systeme geworfen.

## 1.1 Motivation

Um sich die Arbeit zu erleichtern, begannen einige Mitarbeiter der Abteilung für Fahrzeugdiagnose des Unternehmens BMW kleinere Visual Basic Funktionen zu schreiben. Es begann mit sehr einfachen Diagnosefunktionen, wie dem Auslesen des Fehlerspeichers oder der Steuergeräteidentifikation.

Im Laufe der letzten sieben bis acht Jahre entstand eine Microsoft Access Applikation mit über fünfzig Formularen und mindestens genauso vielen Tabellen und Abfragen. An der Entwicklung dieses Tools, welches den Namen DiaAccess trägt, waren insgesamt fünf bis zehn Personen zu verschiedenen Zeiten tätig. Mittlerweile kann der Quellcode der Applikation nur mehr schwer gewartet und weiterentwickelt werden. Trotz vieler Fehler, der Unübersichtlichkeit und ohne klare Struktur ist die Applikation für viele Mitarbeiter eine sehr wichtige Unterstützung bei ihrer täglichen Arbeit.

Die Firma ASE wurde daher beauftragt eine neue Applikation für die Diagnose zu erstellen, welche zwar die Funktionalität von DiaAccess unterstützen soll, aber auch die neuesten Anforderungen in der Softwareentwicklung erfüllen muss. Dazu zählen

vor allem die Wiederverwendbarkeit und die Wartung der einzelnen Diagnosemodule, aber auch die einfache Erweiterung mit neuen Modulen.

## **1.2 Aufbau der Arbeit**

Im Kapitel 2 dieser Arbeit sind die Aufbereitung der bisherigen Erfahrungen in der Fahrzeugdiagnose und die Analyse der Anforderungen enthalten. Dabei werden DiaAccess und bestehende Softwareprodukte der Firma ASE zum Thema Diagnose genauer betrachtet.

Das Kapitel 3 gibt einen Überblick über derzeitige Standards sowie die bisherigen Entwicklungen im Bereich der Ablaufprogrammierung (Workflow Modelle und Sprachen). Es wird genauer auf die Ablaufmodellierung, -definition und -ausführung eingegangen.

Die Aufstellung von bestehenden Projekten und die Bewertung der Technologien im Bezug auf die Anforderungen werden im Kapitel 4 verarbeitet. Derzeitige Softwarelösungen zum Thema Workflow-Management werden beschrieben.

Schließlich gibt das Kapitel 5 einen Einblick in die neue Diagnosesoftware der Firma ASE und die dahinter liegenden Lösungskonzepte. Die zukünftigen Entwicklungen werden angesprochen.

## 2 Systemanalyse und Anforderung

Die Neuentwicklung der Diagnosesoftware wird unter dem Namen Wasaka geführt. Dieses Kapitel erläutert die Thematik der Diagnose, analysiert den Ist-Zustand und stellt die grundlegenden Anforderungen an die Applikation fest.

Nach dem die einzelnen Diagnosekomponenten geklärt werden, zeigt das Unterkapitel 2.2 die Erkenntnisse und Ergebnisse der Firma ASE zum Thema Diagnose im Hause BMW. Die relevanten Softwareprodukte werden etwas näher gebracht, um die Ziele für das Projekt Wasaka besser definieren zu können.

Die bisherige Softwarelösung von BMW wird im Unterkapitel 2.3 vorgestellt. Sie dient als Basis für den Funktionsumfang von Wasaka.

### 2.1 Die Fahrzeugdiagnose im Allgemeinen

Die Fahrzeugdiagnose funktioniert nach bestimmten Protokollen. Zu den Steuergeräten der einzelnen Steuergeräteproduzenten gibt es entsprechende Dokumentationen. Darin sind die Telegramme beschrieben, welche das Steuergerät für die Kommunikation unterstützt. Das Diagnoseprotokoll EDIABAS setzt darauf auf.

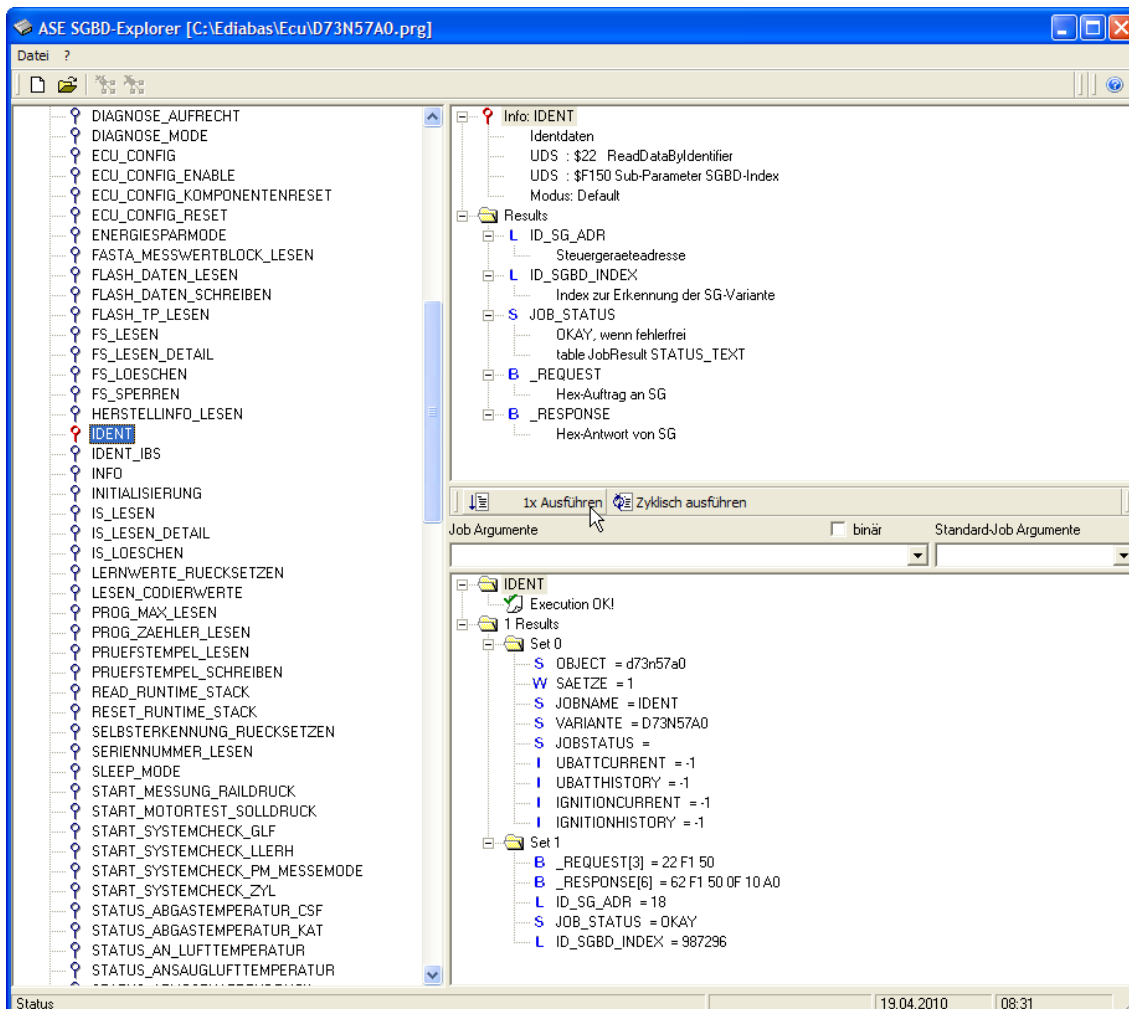
<i>Aufgabe</i>	<i>Anfrage</i>	<i>Antwort</i>
Lesen der IDENT-Daten	82, 12, F1, 1A, 80	83, F1, 12, 7F, 1A, 11, 30
Lesen der Fahrgestellnummer	83, 12, F1, 22, F1, 50	86, F1, 12, 62, F1, 50, 0F, 17, 40, 92
Lesen der Abgleichwerte	84, 12, F1, 2C, 03, F3, 03	84, F1, 12, 6C, 03, F3, 03, EC
	83, 12, F1, 22, F3, 03	87, F1, 12, 62, F3, 03, 00, 00, 00, 00, E2

**Tabelle 1: Fahrzeugdiagnose – Beispieltelegramme zur Kommunikation mit dem Steuergerät**

Um sich aber nicht ständig Telegrammcodes merken zu müssen, erstellt und wartet die Firma BMW sogenannte SGBDen (Steuergerätebeschreibungsdateien). Darin

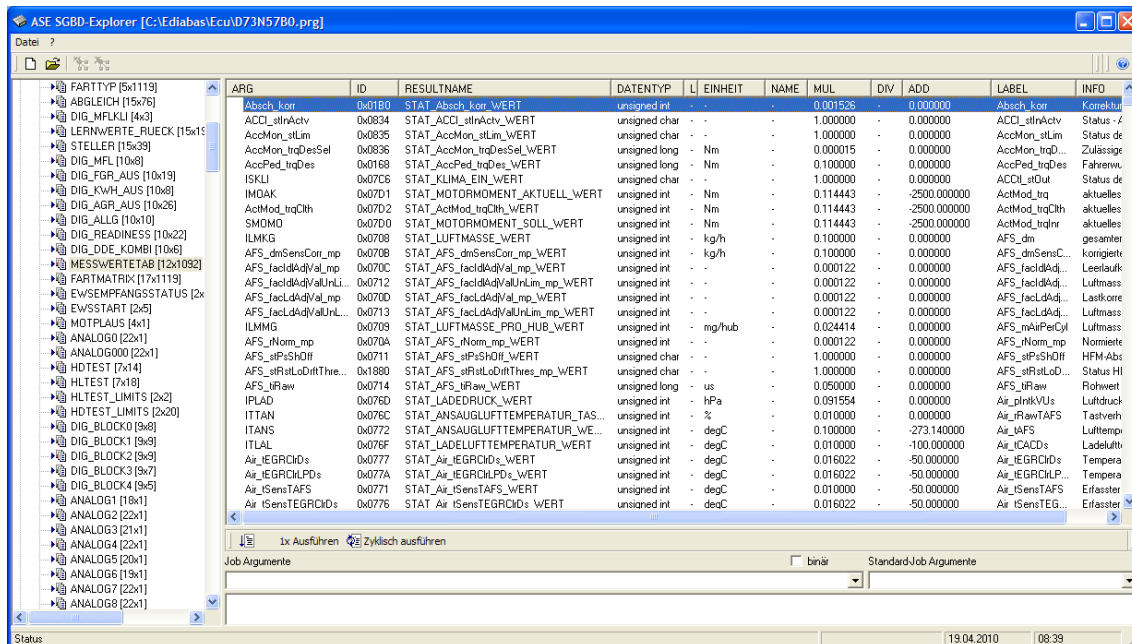
werden in der unternehmenseigenen Programmiersprache, welche sehr der Programmiersprache C ähnlich ist, die Diagnosefunktionen definiert. Diese Diagnosefunktionen werden Jobs oder Services genannt.

Das zu verschickende Telegramm bekommt einen sprechenden Namen (z.B.: FS\_LESEN für Fehlerspeicherlesen) und so auch die Ergebnisse, die diese Funktion vom Steuergerät ermittelt. Mit Hilfe vordefinierter Jobs kann die Liste der unterstützten Jobs oder deren Ein- bzw. Ausgabeparameter samt Beschreibung abgefragt werden. Solche Identifizierungsjobs werden standardmäßig von jeder SGBD unterstützt.



**Abbildung 1: SGBD Explorer - IDENT Job**

Die Ergebnisse aus EDIABAS kommen immer in Schlüssel-Wert Paaren. Diese Schlüssel-Werte Paare werden wiederum in Sets gruppiert. Das Set 0 ist immer gleich und enthält einige Standardinformationen. Alle weiteren Sets sind jobspezifische Ergebnisse.



ARG	ID	RESULTNAME	DATENTYP	L	EINHEIT	NAME	MUL	DIV	ADD	LABEL	INFO
Absch_korr	0x0180	STAT_Absch_korr_WERT	unsigned int	-	-	-	0.001526	-	0.000000	Absch_korr	Korrektur
ACCI_stnActv	0x0834	STAT_ACCI_stnActv_WERT	unsigned char	-	-	-	1.000000	-	0.000000	ACCI_stnActv	Status - /
AccMon_stLim	0x0835	STAT_AccMon_stLim_WERT	unsigned char	-	-	-	1.000000	-	0.000000	AccMon_stLim	Status de
AccMon_trqDesSel	0x0836	STAT_AccMon_trqDesSel_WERT	unsigned long	-	Nm	-	0.000015	-	0.000000	AccMon_trqDes	Zulässige
AccPed_trqDes	0x0168	STAT_AccPed_trqDes_WERT	unsigned long	-	Nm	-	0.100000	-	0.000000	AccPed_trqDes	Fahrerw.
ISKLI	0x07C6	STAT_KLIMA_EIN_WERT	unsigned char	-	-	-	1.000000	-	0.000000	ACCI_stOut	Status de
IMQAK	0x07D1	STAT_MOTORMOMENT_AKTUELL_WERT	unsigned int	-	Nm	-	0.114443	-	-2500.000000	ActMod_trq	aktuelles
ActMod_trqClth	0x07D2	STAT_ActMod_trqClth_WERT	unsigned int	-	Nm	-	0.114443	-	-2500.000000	ActMod_trqClth	aktuelles
SMDMD	0x07D0	STAT_MOTORMOMENT_SOLL_WERT	unsigned int	-	Nm	-	0.114443	-	-2500.000000	ActMod_trqInr	aktuelles
ILMKG	0x0708	STAT_LUFTMASSE_WERT	unsigned int	-	kg/h	-	0.100000	-	0.000000	AFS_dm	gesamter
AFS_dinSensCorr_mp	0x0708	STAT_AFS_dinSensCorr_mp_WERT	unsigned int	-	kg/h	-	0.100000	-	0.000000	AFS_dinSensC...	Korigiert
AFS_facld4d4val_mp	0x070C	STAT_AFS_facld4d4val_mp_WERT	unsigned int	-	-	-	0.000122	-	0.000000	AFS_facld4d4...	Leserlaufk
AFS_facld4d4valLim	0x0712	STAT_AFS_facld4d4valLim_mp_WERT	unsigned int	-	-	-	0.000122	-	0.000000	AFS_facld4d4...	Luftmass
AFS_facld4d4valLim_mp	0x0700	STAT_AFS_facld4d4valLim_mp_WERT	unsigned int	-	-	-	0.000122	-	0.000000	AFS_facld4d4...	Lastkore
AFS_facld4d4valLim...	0x0713	STAT_AFS_facld4d4valLim...	unsigned int	-	-	-	0.000122	-	0.000000	AFS_facld4d4...	Luftmass
ILMMG	0x0709	STAT_LUFTMASSE_PRO_HUB_WERT	unsigned int	-	mg/hub	-	0.024414	-	0.000000	AFS_mAirPerCyl	Luftmass
AFS_rNorm_mp	0x070A	STAT_AFS_rNorm_mp_WERT	unsigned int	-	-	-	0.000122	-	0.000000	AFS_rNorm_mp	Normierte
AFS_stPsShOff	0x0711	STAT_AFS_stPsShOff_WERT	unsigned char	-	-	-	1.000000	-	0.000000	AFS_stPsShOff	HFM-Abs
AFS_stRatLoD...	0x1880	STAT_AFS_stRatLoD...	unsigned char	-	-	-	1.000000	-	0.000000	AFS_stRatLoD...	Status HI
AFS_rRaw	0x0714	STAT_AFS_rRaw_WERT	unsigned long	-	us	-	0.050000	-	0.000000	AFS_rRaw	Rawwert
IPLAD	0x0760	STAT_LADEDRUCK_WERT	unsigned int	-	hPa	-	0.091554	-	0.000000	Air_pInkVUs	Luftdruck
ITTAN	0x076C	STAT_ANSAUGLUFTTEMPERATUR_TAS...	unsigned int	-	%	-	0.010000	-	0.000000	Air_rRawTAFS	Tastverh
ITRANS	0x0772	STAT_ANSAUGLUFTTEMPERATUR_WE...	unsigned int	-	degC	-	0.100000	-	-273.140000	Air_rAFS	Lufttemp
ITLAL	0x076F	STAT_LADELUFTTEMPERATUR_WERT	unsigned int	-	degC	-	0.010000	-	-100.000000	Air_rCADs	Ladeklüth
Air_IeGRChDs	0x0777	STAT_Air_IeGRChDs_WERT	unsigned int	-	degC	-	0.016022	-	50.000000	Air_IeGRChDs	Tempera
Air_IeGRChLPDs	0x077A	STAT_Air_IeGRChLPDs_WERT	unsigned int	-	degC	-	0.016022	-	50.000000	Air_IeGRChLP...	Tempera
Air_I5ensTAFS	0x0771	STAT_Air_I5ensTAFS_WERT	unsigned int	-	degC	-	0.010000	-	50.000000	Air_I5ensTAFS	Erfasster
Air_I5ensTEGRChDs	0x0776	STAT_Air_I5ensTEGRChDs_WERT	unsigned int	-	degC	-	0.016022	-	50.000000	Air_I5ensTEG...	Erfasster

**Abbildung 2: SGBD Explorer – Tabelle der vom Steuergerät unterstützten Messwerte**

Neben den Jobs sind in den SGBDn auch verschiedene Wertetabellen vorhanden, welche zum Beispiel die möglichen Messgrößen des Steuergerätes enthalten. Jede Zeile in der Abbildung 2 stellt eine Messgröße im Steuergerät dar. Um eine bestimmte Messgröße abzufragen wird der Messwertjob mit entsprechenden Parameter, welcher in der Tabelle steht, aufgerufen und anschließend die Ergebnisse nach dem Ergebnisnamen (welcher auch in der Tabelle aufgeführt ist) durchsucht.

## 2.2 Bisherige Erfahrungen und Lösungsansätze

Die Firma ASE hat mit der Fahrzeugdiagnose bei BMW schon einiger Zeit zu tun, wie bereits in der Einführung erwähnt. Unter anderem existiert die sogenannte Labeldatebank für die Verwaltung von Steuergerätedaten der BMW-Fahrzeuge.

Des Weiteren wurde ein Softwareprodukt zur leichteren Parametrierung von Diagnosejobs erstellt. Der sogenannte Diagnose Parameter Assistent war ein wichtiger Schritt zur Beschreibung der typischen Diagnoseprozesse und dem Verständnis über den grundsätzlichen Aufbau der Diagnosekomponenten.

### **2.2.1 Labeldatenbank**

Die wichtigste Aufgabe der Labeldatenbank ist die Verwaltung von Fahrzeugprojekten im Kontext der Steuergerätegrößen. Es handelt sich um ein verteiltes System, welches sowohl BMW intern als auch von Partnerunternehmen verwendet wird.

Die Fahrzeugprojekte sind durch diverse Attribute gekennzeichnet, wie:

- Getriebeart,
- Ausstattung (wie Abstandsensoren, Navigation, automatische Sitzverstellung, etc.),
- Motorleistung,
- länderspezifische Attribute (gesetzliche, klimabezogene, etc.),
- ... und viele mehr.

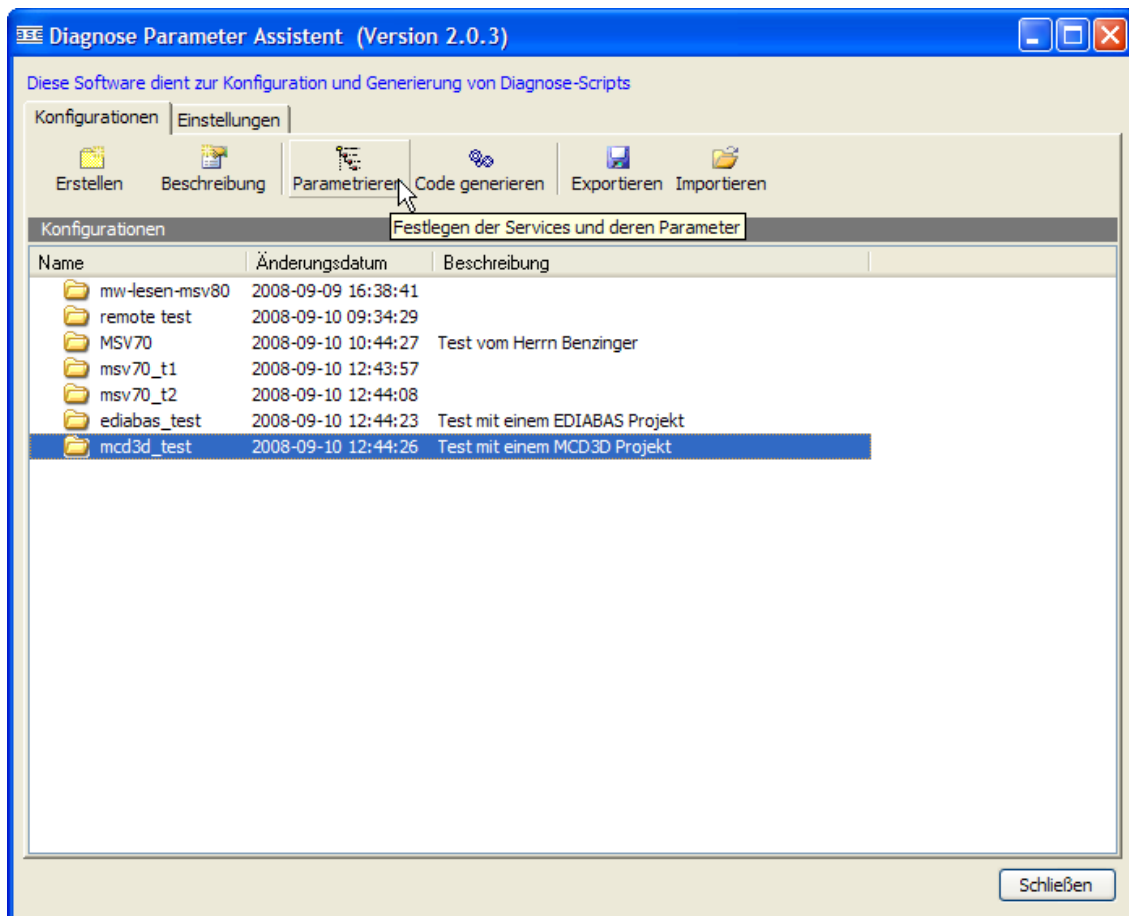
Innerhalb eines Projektes wird die zeitliche Entwicklung der Steuergerätesoftware und der Steuergerätedaten gewartet. So wird mit Hilfe der Labeldatenbank genau verfolgt wer, wann und welche Größe geändert hat. Ein Datensatz eines Steuergerätes hat über 20.000 einstellbare Größen, daher spielt die Datenmenge eine große Rolle. Allein im letzten Jahr wurden über 40.000 Datensätze bearbeitet.

Die Labeldatenbank arbeitet eng mit anderen Datenquellen zusammen, um Steuergerätebeschreibungsdateien (SGBDen) zu erzeugen.

### **2.2.2 Diagnose Parameter Assistent**

Dieses Tool zeigt dem Anwender die Möglichkeiten des Steuergeräts und unterstützt ihn bei der Erstellung von Diagnose-Scripts. Damit der Diagnose Parameter Assistent automatisch Diagnoseskripte erstellen kann, benötigt er vom Anwender Informationen, welche Diagnosefunktionen von welchem Steuergerät mit welchen Parameter zu verwenden sind. Diese Informationen werden in

Konfigurationen gruppiert und gespeichert. Eine Konfiguration kann aus mehreren EDIABAS bzw. MCD-3<sup>1</sup> Projekten bestehen. Die Applikation ist also nicht nur für die Arbeit mit dem BMW eigenen Diagnosesystem EDIABAS entwickelt worden, sondern auch für die Arbeit mit allen Diagnosesystemen, welche die MCD-3 Schnittstelle implementieren.



**Abbildung 3: Diagnose Parameter Assistent – Liste aller Konfigurationen**

Mit dem Anlegen einer Neuen bzw. dem Öffnen einer bestehenden Konfiguration fängt die eigentliche Parametrierung an. Der Dialog in Abbildung 4, welche nach

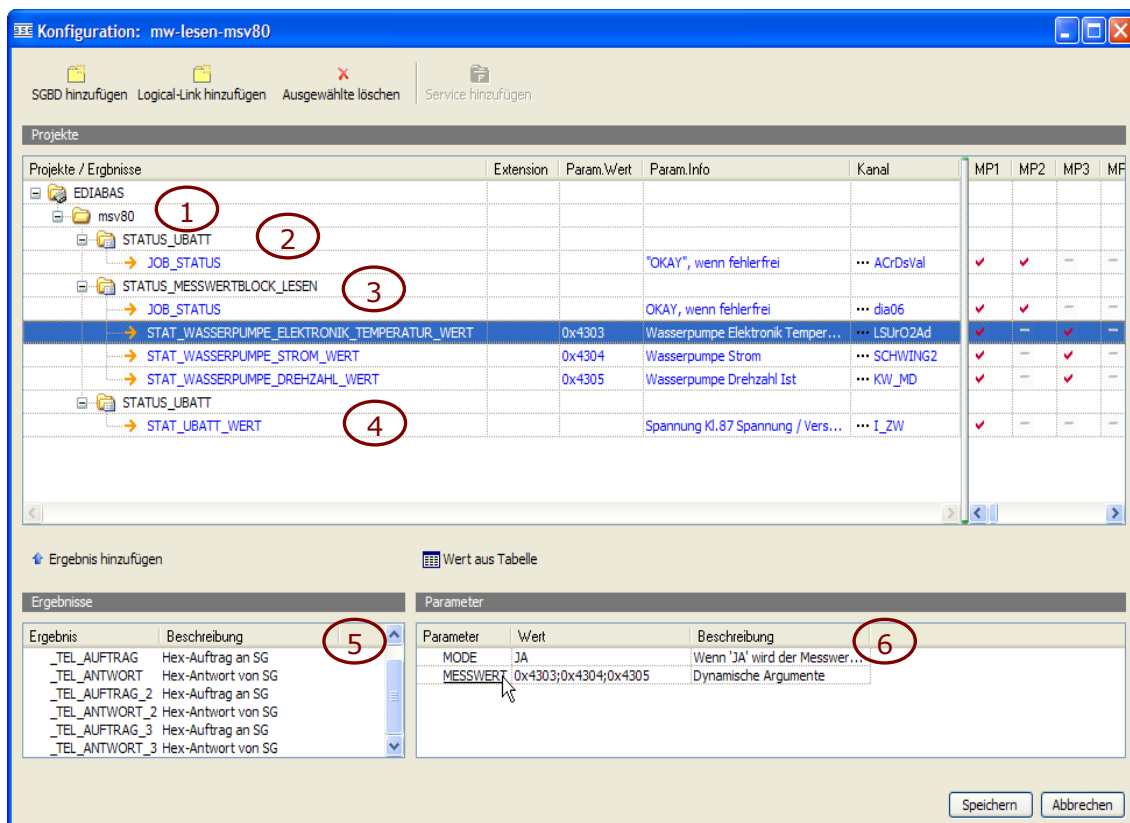
---

<sup>1</sup> MCD-3 ist ein Standard der Association for Standardisation of Automation and Measuring Systems (ASAM), welcher die Schnittstelle zu einem Diagnosesystem spezifiziert. [http://www.asam.net/index.php?option=com\\_content&task=view&id=124&Itemid=206](http://www.asam.net/index.php?option=com_content&task=view&id=124&Itemid=206)  
(24.11.2010)



dem Öffnen erscheint, enthält alle Elemente einer bestimmten Diagnosekonfiguration. Das Beispiel beinhaltet ein EDIABAS Projekt mit drei verschiedenen SGBDn.

Der Dialog zur Parametrierung besteht aus einer Baumstruktur mit Tabellenelementen. Mit Hilfe dieser Tabelle werden für jedes Steuergerät die benötigten Diagnoseservices ausgewählt. Wenn ein Diagnoseservice Eingangswerte benötigt, können diese angegeben werden. Die Diagnoseservices liefern viele Ergebnisdaten zurück und der Anwender kann entscheiden, welche dieser Daten für ihn von Bedeutung sind.



**Abbildung 4: Diagnose Parameter Assistent - Parametrierung (Ein EDIABAS und ein MCD-3 Projekt)**

### 2.2.2.1 Beschreibung der Elemente des Parametrier-Dialogs

1. Die erste Ebene sind Projekte. Es werden EDIABAS und MCD-3 Projekte unterschieden.

2. Die zweite Ebene entspricht den SGBDn (unter EDIABAS) bzw. den Logical-Links (unter MCD-3).
3. Ebene drei sind die eigentlichen Services. Mit dem Klick auf ein Service werden in der linken unteren Liste (Liste aller Ergebnisse - Nummer 5) die statischen Ergebnisse eines Services aufgezählt. Diese Ergebnisse werden bei jeder Ausführung des Services zurückliefert. Die Liste, welche in der Abbildung 4 mit der Nummer 6 (unten rechts) gekennzeichnet ist, enthält dann Argumente eines Services.
4. Die Ebene vier sind relevante Ergebnisse des Services. Diese Ergebniswerte können auch für zwanzig verschiedene Messwert-Pläne aktiviert werden. Um ein Ergebnis als relevant zu markieren, wählt man es in der Ergebnisliste aus und klickt auf den Button „Ergebnis hinzufügen“. Das Ergebnis wird dann dem Service im Baum angeschlossen.
5. Dieses Fenster enthält die Liste der statischen Ergebnisse eines Services, d.h. Ergebnisse, welche bei jeder Serviceausführung entstehen, mit der entsprechenden Beschreibung. Im Unterschied dazu generieren gewisse Services (z.B.: STATUS\_MESSWERTBLOCK\_LESEN) dynamische Ergebnisse abhängig von den Argumenten der Parameter.
6. Liste aller Parameter eines Services mit Beschreibung.

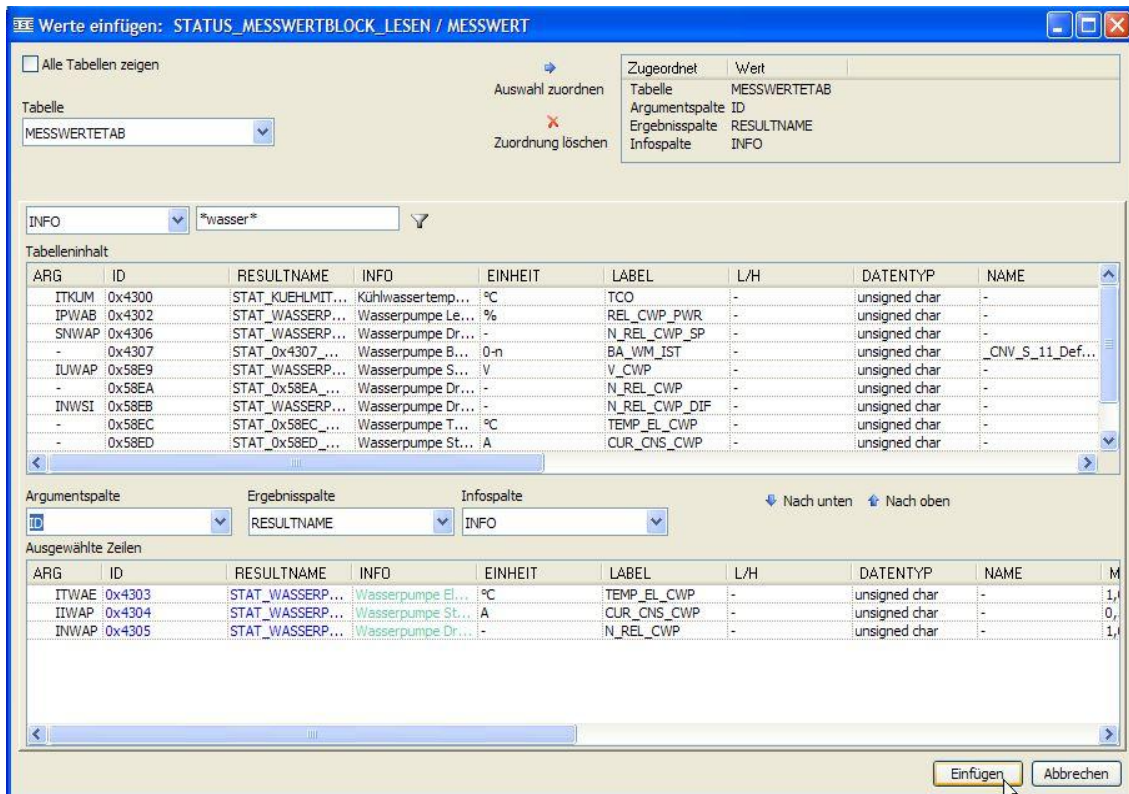
#### 2.2.2.2 Der Ablauf

Zuerst wird eine SGBD (beim Arbeiten mit EDIABAS) oder ein Logical-Link (beim Arbeiten mit einem MCD-3 System) ausgewählt und hinzugefügt.

Des Weiteren werden die Eingangsparameter durch Benutzereingabe oder durch richtige Auswahl aus den Tabellen, welche sich in der SGBD befinden, vorgegeben. Der letzte Schritt ist die Bestimmung der relevanten Ergebnisdaten des entsprechenden Jobs.

Der zusätzliche Dialog zur Auswahl von Tabellenwerten, ist sehr umfangreich. Hier findet ein wichtiger Arbeitsprozess statt. Neben der Auswahl der richtigen Werte können hier Daten verknüpft werden, so dass die folgenden Parametrierungen immer schneller ablaufen können.

Zu einem bestimmten Parameter können somit die entsprechen Tabelle und Spalten zugeordnet werden, damit beim nächsten Öffnen des Dialogs diese Auswahl bereits getroffen ist. Somit muss nicht jeder Benutzer aufs Neue herausfinden, wo sich die möglichen Parameter für einen bestimmten Job befinden.



**Abbildung 5: Diagnose Parameter Assistent - Tabellen einer Steuergerätedatei**

Ist der Benutzer fertig, kann der VBScript Code generiert werden. Pro SGBD bzw. Logical-Link wird eine Klasse erzeugt und für jeden parametrisierten Job wird eine Funktion in dieser Klasse geschrieben. Der generierte Code ist bereits lauffähig.

### 2.2.2.3 Die wichtigsten Erkenntnisse aus diesem Projekt

Durch dieses Projekt kamen viele ausschlaggebende Zusammenhänge ans Licht, welche auch die Rahmenbedingungen für das Projekt Wasaka festlegen:

- Die Namensgebung für die Jobs in den SGBDs ist nicht zu hundert Prozent stimmig, was eine automatische Bestimmung schwierig macht. Zum Beispiel heißt der Job zum Lesen der Messwerte in der einen SGBD STATUS\_MESSWERTEBLOCK\_LESEN und in einer anderen MW\_SELECT\_LESEN\_NORM. Ferner können sich auch die verwendeten Argumente voneinander unterscheiden.
- Die Listen mit möglichen Werten und deren Beschreibungen für die Argumente sind in Tabellen aufgelistet. Es gibt keine Möglichkeit aus dem

Diagnosesystem herauszulesen, welche Tabelle für welches Argument die richtige ist. Dieses Wissen ist nicht durchgehend dokumentiert.

- Wenn ein Job mit einer Argumentliste aufgerufen wird, muss die Reihenfolge der Ergebnisse nicht der Reihenfolge der Argumente entsprechen. Der Schlüssel bzw. die ID des entsprechenden Ergebnisses zu dem entsprechenden Argument ist zwar in einer weiteren Tabelle der SGBD vermerkt, aber auch dieses Wissen ist nicht dokumentiert.

## 2.3 Diagnose mit DiaAccess

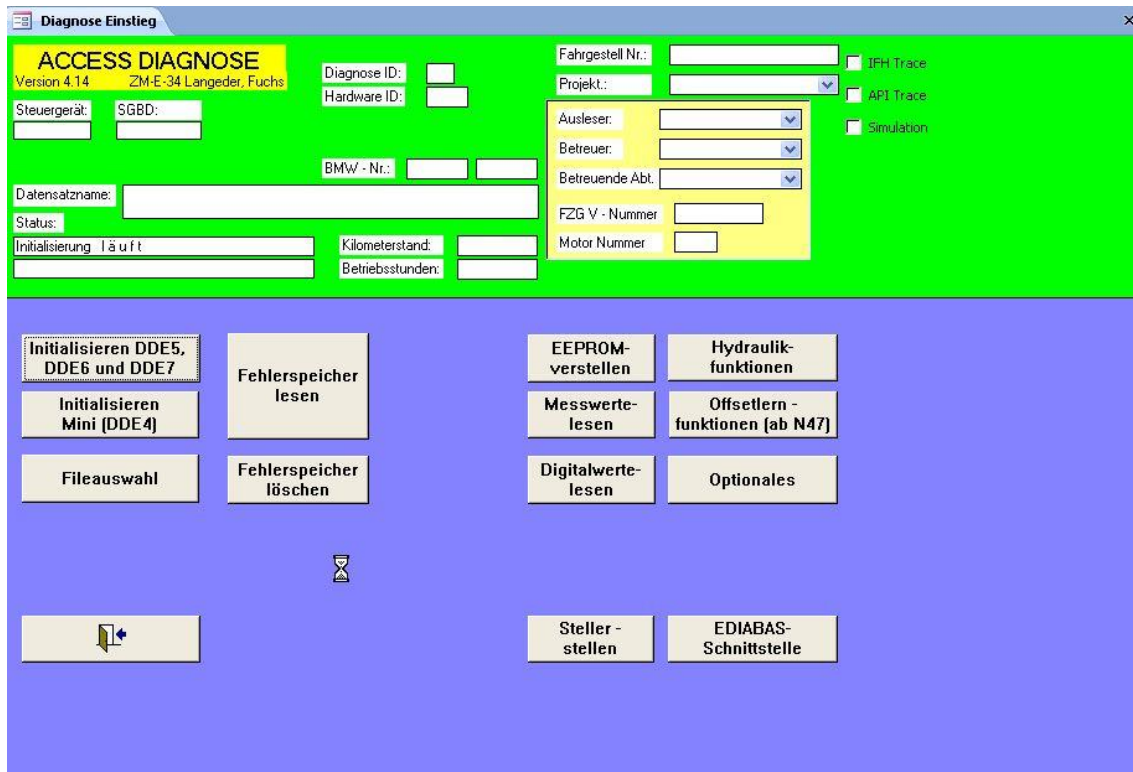
Die Microsoft Access Applikation DiaAccess schließt die eher trivialen Jobs des Diagnosesystems EDIABAS in komplexere Diagnoseprozesse zusammen. Das Tool beinhaltet eine Reihe von Formularen, Tabellen und Codebibliotheken, um dem Benutzer zu unterstützen und ihm eine klare Sicht auf die relevanten Daten zu geben.

In dieser Zeit gab es insgesamt vier verschiedene Steuergerätefamilien. Die verschiedenen Familien unterscheiden sich zum Teil stark im Bezug auf Diagnosejobs, deren Aufbau und Verwendung. Wenn eine neue Familie entwickelt wurde, wurden auch die Formulare und die dahinterliegenden Codefunktionen teilweise komplett neu programmiert. Die Änderung war allerdings so, dass der Code einfach kopiert und angepasst wurde. So entstand ein weiterer Ablauf. Die Wiederverwendung vom Code fand in diesem Projekt fast gar nicht statt.

War eine Funktion fehlerhaft und wurde von einem Mitarbeiter erstellt, welcher nicht mehr in der Abteilung war, so erstellte der neue Kollege eine komplett neue Funktion, kopierte den funktionierende Teil des Codes in die Funktion und versuchte den Ablauf diesmal richtig zu implementieren.

### 2.3.1 Funktionsweise und Anwendung

Die DiaAccess Versionen wurden durch einfaches „Copy-Paste“ verteilt. Die Abbildung 6 zeigt das Einstiegsformular. Es erlaubte dem Benutzer, sich mit dem Steuergäret zu verbinden bzw. das Steuergerät zu initialisieren. Des Weiteren wurden hier projektrelevante Daten angezeigt, verändert und gespeichert.



**Abbildung 6: DiaAccess - Einstiegsformular**

War die Initialisierung erfolgreich, konnten die verschiedenen Diagnosefunktionen genutzt werden.

## 2.3.2 Typische Diagnoseprozesse

In diesem Unterkapitel werden einige kennzeichnende Diagnoseaufgaben dargestellt.

### 2.3.2.1 Automatisches Initialisieren

Das Automatische Initialisieren zur Ermittlung der richtigen SGBD ist der erste Ablauf, der analysiert wird.

Wenn man sich an ein Steuergerät anschließt, dann wird die richtige SGBD gebraucht um mit dem Steuergerät korrekt kommunizieren zu können. Dazu muss das richtige Diagnoseprotokoll (KWP2000 oder UDS), welches zur Kommunikation verwendet wird, ermittelt werden. Zu jedem der Diagnoseprotokolle existiert eine so genannte globale bzw. allgemeine SGBD. Diese kann immer verwendet werden

um den Job IDENT, welcher unabhängig von der Version oder der Familie des Steuergerätes ist, auszuführen.

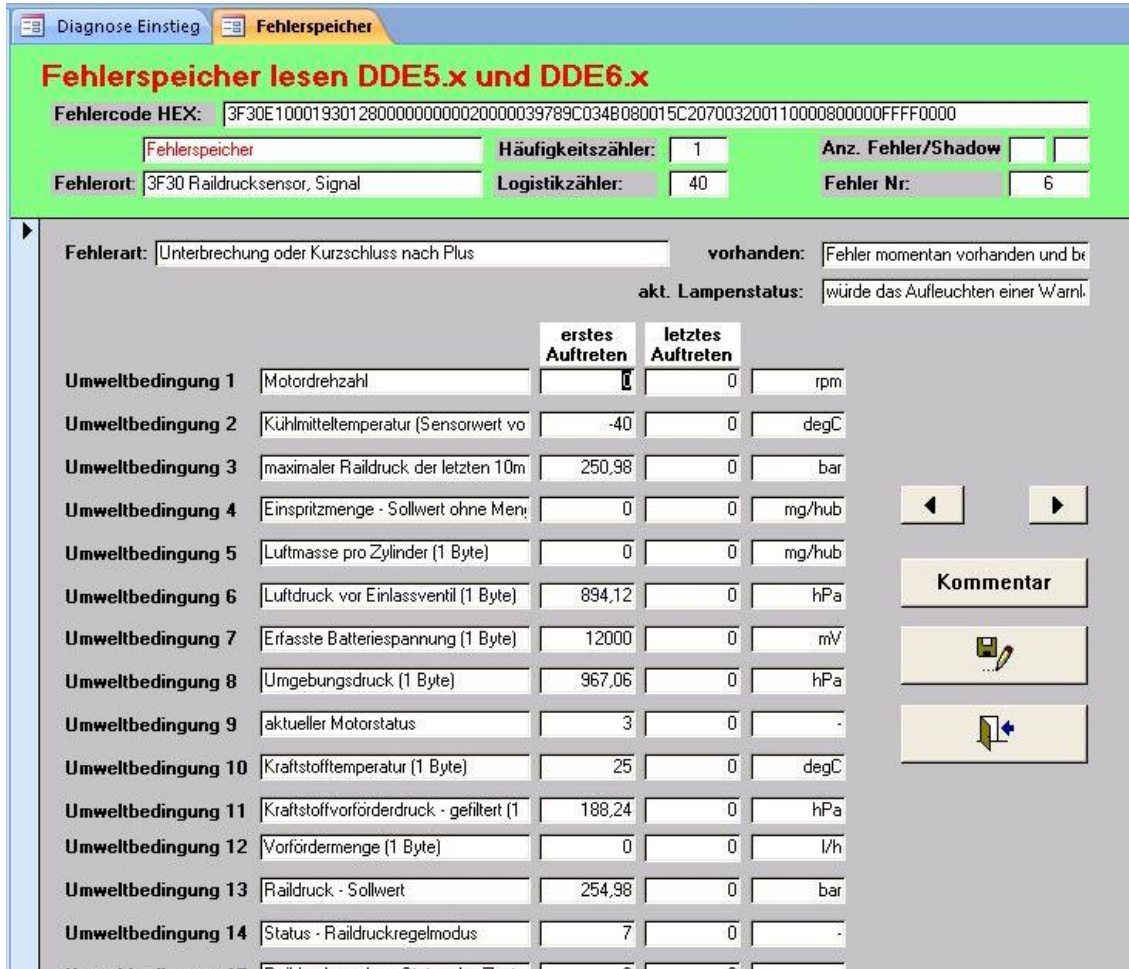
Dieser Job liefert dem Anwender einige grundlegende Informationen, unter anderem den Hardwareindex und den Diagnoseindex. Die Zuordnung dieser zwei Indices zu dem SGBD Namen wird in der Labeldatenbank gewartet und kann dort abgefragt werden.

Die einzelnen Teilschritte um diesen Ablauf richtig durchzuführen sind folgende:

- 1) Nehme die erste globale SGBD aus der Liste
- 2) Führe den Job IDENT aus
- 3) Prüfe die Ergebnisse
  - a. Ist die Antwort positiv:
    - i. Ermittle den Hardwareindex und den Diagnoseindex
    - ii. Ermittle den SGBD Namen aus der Labeldatenbank anhand der Indices
  - b. Falls die Antwort negativ ist, nehme die nächste globale SGBD und gehe zu Schritt 2

#### *2.3.2.2 Lesen des Fehlerspeichers*

Das Lesen bzw. das Löschen des Fehlerspeichers ist nach dem automatischen Initialisieren wohl die meistverwendete Funktion. Die Fehler werden mittels zwei Jobs aus dem Speicher des Steuergerätes ausgelesen.



**Fehlerspeicher lesen DDE5.x und DDE6.x**

Fehlercode HEX: 3F30E10001930128000000000020000039789C034B080015C207003200110000800000FFFF0000

Fehlerspeicher Häufigkeitszähler: 1 Anz. Fehler/Shadow

Fehlerort: 3F30 Raildrucksensor, Signal Logistikzähler: 40 Fehler Nr.: 6

Fehlerart: Unterbrechung oder Kurzschluss nach Plus vorhanden: Fehler momentan vorhanden und bei akt. Lampenstatus: würde das Aufleuchten einer Warnl.

		erstes Auftreten	letztes Auftreten	
Umweltbedingung 1	Motordrehzahl	0	0	rpm
Umweltbedingung 2	Kühlmitteltemperatur (Sensorwert vor	-40	0	degC
Umweltbedingung 3	maximaler Raildruck der letzten 10m	250,98	0	bar
Umweltbedingung 4	Einspritzmenge - Sollwert ohne Men	0	0	mg/hub
Umweltbedingung 5	Luftmasse pro Zylinder (1 Byte)	0	0	mg/hub
Umweltbedingung 6	Luftdruck vor Einlassventil (1 Byte)	894,12	0	hPa
Umweltbedingung 7	Erfasste Batteriespannung (1 Byte)	12000	0	mV
Umweltbedingung 8	Umgebungsdruck (1 Byte)	967,06	0	hPa
Umweltbedingung 9	aktueller Motorstatus	3	0	-
Umweltbedingung 10	Kraftstofftemperatur (1 Byte)	25	0	degC
Umweltbedingung 11	Kraftstoffvorförderdruck - gefiltert (1	188,24	0	hPa
Umweltbedingung 12	Vorfördermenge (1 Byte)	0	0	l/h
Umweltbedingung 13	Raildruck - Sollwert	254,98	0	bar
Umweltbedingung 14	Status - Raildruckregelmodus	7	0	-
Umweltbedingung 15	Raildruckregelmodus Status des Zugs	0	0	-

Kommentar

akt. Lampenstatus: würde das Aufleuchten einer Warnl.

**Abbildung 7: DiaAccess - Darstellung des Fehlerspeichers**

Zuerst wird die Liste der Fehler mit dem Job FS\_LESEN ausgelesen. Danach werden für jeden aufgetreten Fehlerort die Details mit FS\_LESEN\_DETAIL ermittelt. Diese beiden Jobs sind in jedem Steuergerät einheitlich benannt und identisch anzuwenden.

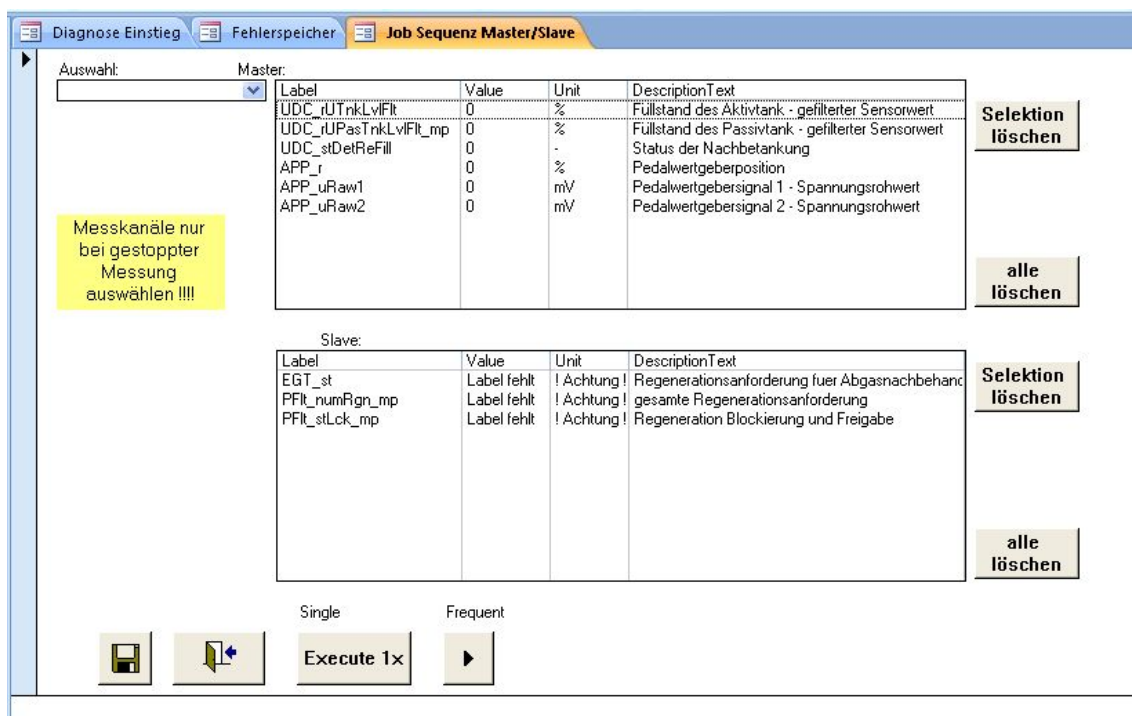
Folgende Schritte werden für das Lesen des Fehlerspeichers durchgeführt:

- 1) Ausführen des Jobs FS\_LESEN
- 2) Prüfung der Ergebnisse und Ermittlung der Fehlercodes
- 3) Pro Fehlercode wird der Job FS\_LESEN\_DETAIL mit dem Fehlercode als Argument ausgeführt
- 4) Prüfung der Ergebnisse und Ermittlung der Fehlerdetails

## 5) Darstellung des Fehlerspeichers

## 2.3.2.3 Das Messen von Werten

Das Messen von Werten ist ein sehr einfacher Prozess. Es gilt lediglich den richtigen Job (Abhängig vom Diagnoseprotokoll und der Steuergerätefamilie) für das Messen zu ermitteln, mit entsprechenden Argumenten auszuführen und die Ergebnisse zu prüfen. Dieser Job kann beliebig viele Argumente annehmen und ebenso viele Ergebnisse liefern. Die Herausforderung dabei ist zu wissen, wie das Argument für eine bestimmte Messgröße und das dazugehörige Ergebnis lauten.



**Abbildung 8: DiaAccess – Dialog zum Messen**

Der wesentliche Prozess hier ist die Ermittlung von Argument und Ergebnis, sowie die Bestimmung der Tabelle, die diese beiden Informationen bereithält.

Das Messen von Werten wird wie folgt durchgeführt:

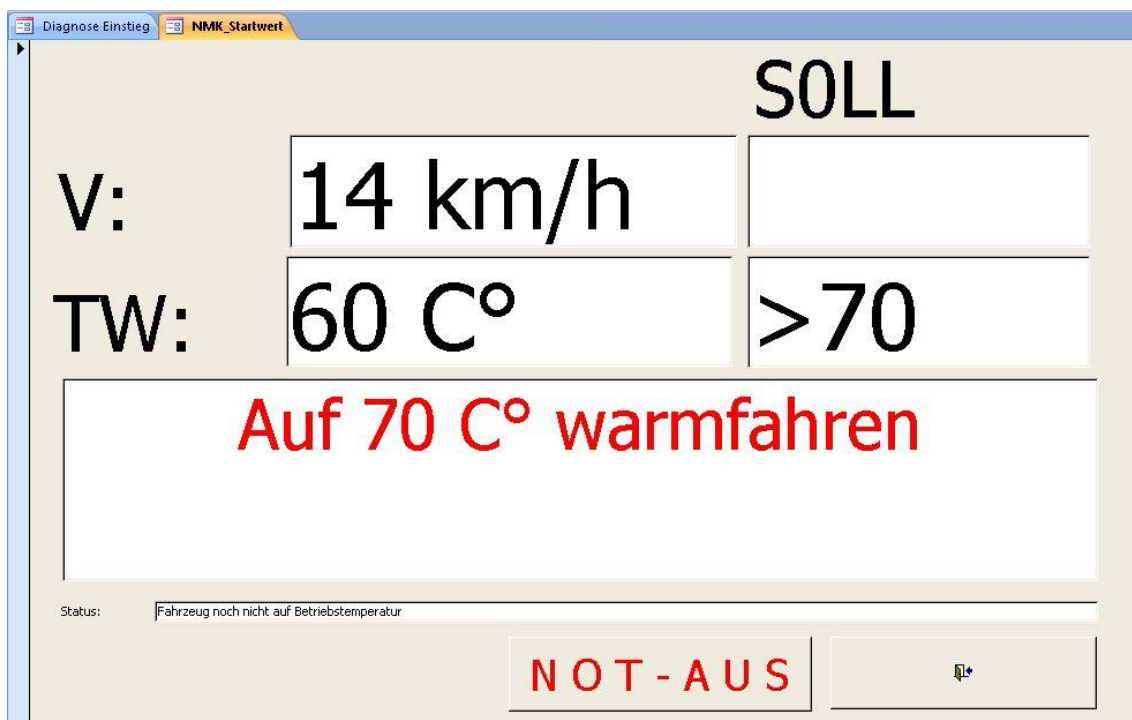
- 1) Auswahl der Messgrößen für den Job
- 2) Ermitteln der Argument- und der Ergebnisnamen zu den Messgrößen



- 3) Messjob ausführen
- 4) Ergebnisse anzeigen

#### 2.3.2.4 NMK Schnelllernen

Dies ist wohl einer der umfangreichsten Diagnoseprozesse. Der Ablauf ist außerdem viel komplexer als bei den vorhergehenden Diagnoseaufgaben.



**Abbildung 9: DiaAccess - NMK Schnelllernen**

Das Modul NMKFast bietet Zugriff auf die NMK-Werte (Nullmengenkalibrierung) und die damit in Zusammenhang stehenden Größen. Das Ziel ist die Verbesserung von Komfort (Geräusch) und Emissionen durch Driftkompensation der Voreinspritzmenge über die Injektor-Lebensdauer.

Prozess „NMK Schnelllernen“:

- 1) Fixieren der Lern-Konfiguration (Grenzwerte und relevante Messgrößen)
- 2) Motor auf ca. 70 C° warmlaufen lassen
- 3) Programmieren der NMK Startwerte

- 4) Zündung auf AUS stellen
- 5) Nachlauf abwarten, damit die Werte in das Steuergerät programmiert werden
- 6) Zündung auf EIN stellen
- 7) Starten des Motors (Drehzahl > 500)
- 8) Auf Geschwindigkeit von ca. 65 km/h schleppen und den vierten Gang einlegen
- 9) Das Lernen der Werte ist der wichtigste Unterprozess in diesem Zusammenhang, welcher aber nicht weiter erörtert wird um den Rahmen dieser Arbeit nicht zu sprengen.
- 10) Ist das Lernen abgeschlossen, werden die gelernten Werte gespeichert

### **2.3.3 Die Analyse der gesammelten Erfahrungen**

Um die Anforderungen an das Nachfolgesystem genauer beschreiben zu können, wurde DiaAccess genauer unter die Lupe genommen. Für die Analyse wurde mir der gesamte alte DiaAccess Code zur Verfügung gestellt.

Da viele Funktionalitäten und Formulare bereits überholt und nicht mehr in Verwendung waren, wurde eine Liste an Ansichten und Abläufen festgelegt, welche in der neuen Software implementiert werden müssen.

Die Analyse von DiaAccess und der Geschäfts- bzw. Diagnoseprozesse welche sie inne hatte, erwies sich als sehr schwierig. Die Software hat keine klare bzw. saubere Struktur. Die vielen globalen Variablen und die schlechte Namensgebung erschwerten die Sache erheblich. Die Funktionalität der Software wurde nirgends dokumentiert, daher konnten viele Abläufe nicht mehr richtig nachvollzogen werden.

## **2.4 Zusammenfassung**

Für die erste Version von Wasaka wurden über zwanzig Diagnoseabläufe definiert, die von der neuen Software unterstützt werden müssen. Dazu ist die Kommunikation mit dem Diagnosesystem und im zweiten Schritt die Anbindung an die Labeldatenbank notwendig.

Am Beginn der Arbeit und der Designphase ist der generelle Ablauf klar und die Diagnose scheint sich auf ganz einfache Abfolge von Jobausführungen zu beschränken. Sehr viele Details haben aber bereits erkennen lassen, dass die vielen steuergerätespezifischen Ausnahmen die Komplexität verstärken werden.

Die weiteren Anforderungen an Wasaka, wie die bezüglich grafischer Unterstützung, Dokumentation oder Lizenzierung, werden nicht weiter angeführt, da diese nicht zum Kernthema dieser Diplomarbeit gehörten.

## 3 Das Entwickeln von Abläufen

---

Die Anforderung, mit welcher sich diese Arbeit im Folgenden auseinandersetzt, ist das Entwerfen und Entwickeln sowie die Ausführung von Diagnoseabläufen. Vor allem das Gestalten der einzelnen Diagnoseschritte sollte einfach und möglichst mit grafischer Unterstützung stattfinden. Zu diesem Zweck zielte die Recherche zu dieser Arbeit auf Workflow Management Systeme und Modelle.

Die Standards und Projekte, welche sich ausschließlich auf das Web beschränken, wurden hier nicht behandelt, da es einerseits den Rahmen sprengen würde und andererseits nicht zu den Anforderungen an das neue System passt.

### 3.1 Workflow Management Systeme

Das folgende Unterkapitel erklärt die Zusammenhänge und Bedeutungen der Begriffe zu diesem Thema. Es sind mittlerweile viele Definition in der Literatur zu finden, wenn es sich aber um Workflow Systeme handelt, dann ist die Workflow Management Coalition (WfMC) wahrscheinlich die erste Adresse, zumal sie genau zu diesem Zweck gegründet wurde. Diese Organisation wird im Unterkapitel 3.1.2.1 weiter vorgestellt.

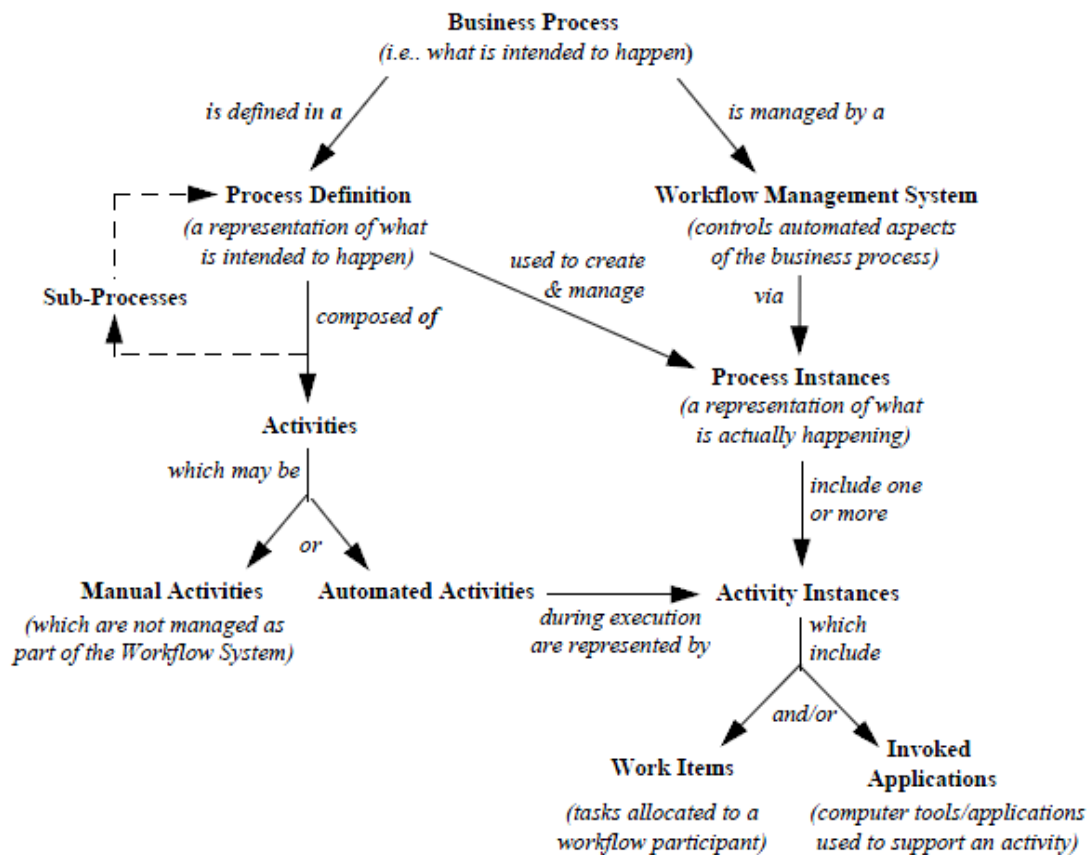
Zunächst aber wird eine gemeinsame Begriffsbasis geschaffen.

#### 3.1.1 Elemente, Begriffe und Definitionen

Eine der wichtigsten Quellen ist also das Glossar, welches für die registrierten Benutzer des Internetportals des WfMC<sup>2</sup> zugänglich ist.

---

<sup>2</sup> Workflow Management Coalition: <http://www.wfmc.org/> (24.03.2010)



**Abbildung 10: WfMS – Beziehungen der Basisterminologie (Workflow Management Coalition, 1999)**

Die Grafik zeigt die Ordnung der verschiedenen Termini, wobei der linke Teil sich mehr auf die Modellierung und Prozessdefinition bezieht. Der rechte Teil des Baumes hat mit der Ausführung der Workflows zu tun.

### 3.1.1.1 Workflow

Die Definition des Wortes wird dem Glossar (Workflow Management Coalition, 1999) des WfMC entnommen:

**Definition:** „The automation of a business process, in whole or part, during which documents, information or tasks are passed from participant to another for action, according to a set of procedural rules.“

Ein Workflow besteht aus einer definierten Anzahl von Aktivitäten und Subworkflows. Das heißt, Workflows lassen sich beliebig schachteln. Er hat einen definierten Anfang und ein definiertes Ende.

Grundsätzlich kann noch zwischen Produktionsworkflows (engl. production workflow) und Ad-Hoc-Workflows unterschieden werden. Bei den ersteren sind alle prozeduralen Regeln im Vorfeld definiert. Die Regeln für die letzteren allerdings können sich während der Ausführung der einzelnen Workflow-Aktivitäten ändern.

### *3.1.1.2 Workflow Management Systeme*

Der WfMC beschreibt diesen Begriff wie folgt (Workflow Management Coalition, 1999):

**Definition:** *„A system that defines creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and application.*

Ein Workflow Management System (WfMS) ist demnach eine Software, welche die Arbeitsabläufe auf vielen Ebenen durch Computerunterstützung automatisieren soll. Auf der einen Seite werden die Workflows hier entwickelt, definiert und den zuständigen Teilnehmern zugewiesen. Die zweite Aufgabe ist die tatsächliche Ausführung, welche die Steuerung und Kontrolle der Prozessabläufe inne hat.

### *3.1.1.3 Geschäftsprozess (Business Process)*

Der Geschäftsprozess wird durch eine logische Verkettung von einzelnen Aktivitäten dargestellt. Er unterliegt vorgegebenen Geschäftsregeln, welche die Vorgehensweise und die Kombination von bestimmten Einflüssen festlegen. Sowohl der Beginn als auch das Ende des Geschäftsprozesses sind genau definiert.

### *3.1.1.4 Prozessdefinition*

Die Prozessdefinition ist die Darstellung eines Geschäftsprozesses in einer digitalen Form. Des Weiteren ist eine solche Definition maschinell lesbar und manipulierbar. Sie beinhaltet alle Elemente, die zur Definition eines Workflows notwendig sind.

Ein solches Format ist zum Beispiel das XPD, welches vom WfMC verabschiedet wurde und im Unterkapitel 3.3.2 beschrieben wird.

### *3.1.1.5 Aktivität*

Die Aktivität ist die kleinste logische Einheit in einem Workflow. Sie beschreibt eine Tätigkeit, welche aber nicht unbedingt vom Computer ausgeführt wird. Das heißt, es wird zwischen manuellen Tätigkeiten (ohne Computerunterstützung) und automatisierten Tätigkeiten (mit Computerunterstützung) unterschieden.

Eine automatisierte Aktivität wird einem Workflow-Teilnehmer zugewiesen.

### *3.1.1.6 Prozess- oder Aktivitätsinstanz*

Eine Instanz repräsentiert eine konkrete Ausführung einer Aktivität oder eines Prozesses samt zugehörigen Daten. Prozess- bzw. Aktivitätsinstanzen werden zur Ausführungszeit vom Workflow-Management-System erstellt und verwaltet.

Wird ein Prozess parallel ausgeführt, so entstehen zwei Prozessinstanzen eines und desselben Prozesses.

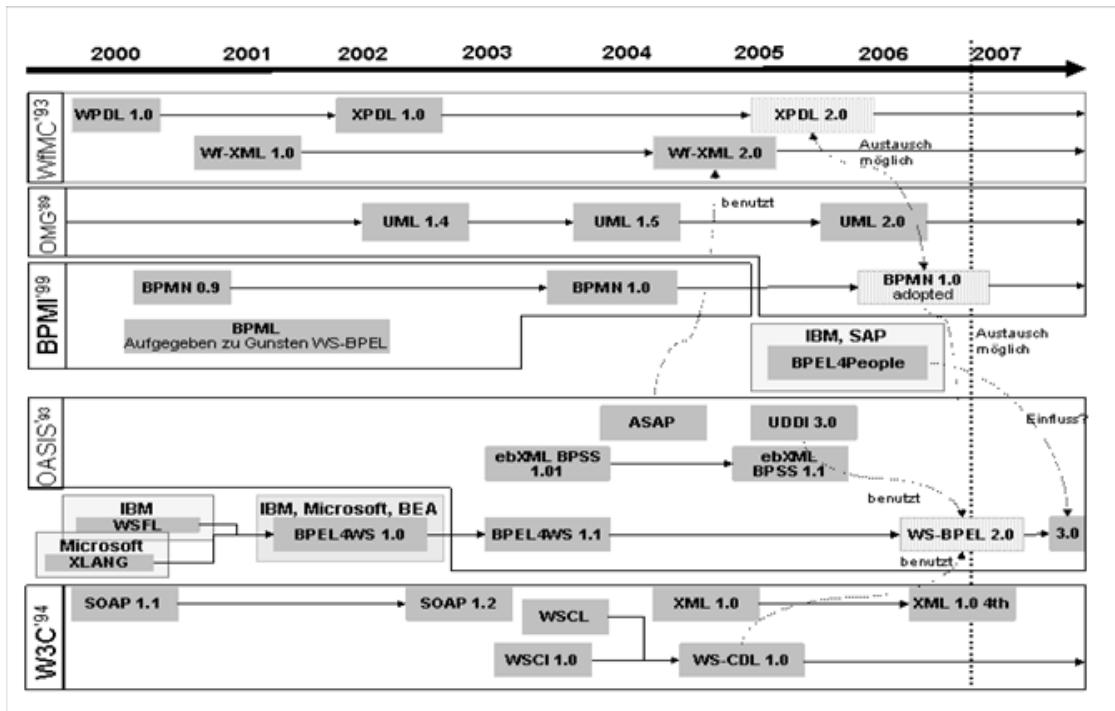
### *3.1.1.7 Teilnehmer*

Teilnehmer des Workflows sind üblicherweise Menschen, welche ihnen zugewiesenen Aktivitäten durchführen. Die Tätigkeit könnte aber auch ein intelligentes Computerprogramm ausführen.

## **3.1.2 Organisationen zur Standardisierung und Entwicklung der WfMS-Technologien**

Die Recherche zielt auf standardisierte Notationen, Sprachen und Vorgehensweisen. Durch die vielen Standardisierungsorganisationen, deren Arbeit sich oft überschneidet, ist es schwer den Überblick zu behalten.

Die folgende Abbildung zeigt die zeitliche Entwicklung der Standards und der involvierten Organisationen vom Jahr 2000 bis ins Jahr 2007.



**Abbildung 11: WfMS – Zeitliche Entwicklung der Standards (Bartonitz, BPEL - Business Process Execution Language, 2007)**

Auf die einzelnen Standards wird noch innerhalb dieses Kapitels 3 näher eingegangen.

### 3.1.2.1 Workflow Management Coalition

Diese internationale, nicht kommerzielle Organisation wurde bereits 1993 von Entwicklern, Anwendern, Herstellern und Forschungseinrichtungen, welche sich mit Workflow Management Systemen beschäftigen, ins Leben gerufen. In der Zwischenzeit finden sich prominente Unternehmen wie Adobe, IBM, Oracle und Sun Microsystems unter den Mitgliedern. Zu den wichtigsten Aufgaben gehören die Entwicklung der Prozessbedingten Standards und Schnittstellen, sowie die dazugehörige Begriffsbestimmung (Workflow Management Coalition, 2010).

Die bedeutungsvollsten Ergebnisse erzielte die WfMC durch die Spezifikation des XPDL Dateiformats für die Modellierungsnotation der Geschäftsprozesse (Business Process Modelling Notation - BPMN).



### 3.1.2.2 Object Management Group und Business Process Modeling Initiative

Die *Object Management Group* (OMG) ist internationales Konsortium, welches 1989 gegründet wurde. Die OMG Arbeitsgruppen entwickeln herstellerunabhängige Standards für eine große Auswahl an Technologien und eine noch größere Gruppe an Branchen. Zu den Gründern gehörten Apple, IBM und Sun Microsystems. Seit 2008 zählt auch Microsoft zu den fast 800 Mitgliedern (OMG, 2010).

Im Jahr 2005 gingen die OMG und die *Business Process Modelling Initiative* (BPMI) eine Kooperation ein, um gemeinsam die *Business Process Modelling Notation* (BPMN) zu pflegen. Die BPMI entstand im Jahr 2002, nach dem der IBM-Mitarbeiter Stephen A. White die BPM-Notation erarbeitet hatte (White, Introduction to BPMN, 2004).

### 3.1.2.3 RosettaNet

RosettaNet ist ein Non-Profit-Konsortium von weltweit mehr als 600 Unternehmen, vornehmlich aus den Bereichen Informationstechnologie, elektronische Bauelemente und Halbleiterfertigung, Logistik, Telekommunikation, Dienstleistung etc. Ziel ist eine Standardisierung von Datenübertragungsschnittstellen im gegenseitigen elektronischen Datenaustausch (RosettaNet, 2010).

Die Arbeit der Organisationen basiert auf globalen, offenen XML-Standards. Diese konzentrieren sich vorwiegend auf die Optimierung bestehender Systeme und Schnittstellen zwischen den Partnerunternehmen. Dadurch soll vor allem die Kompatibilität von Geschäftsprozessen vorangetrieben werden, die in weiterer Folge zu kostengünstigem, schnellen und fehlerfreien Datenaustausch führen soll (RosettaNet, 2010).

Während RosettaNet vorwiegend in den USA und den asiatischen Staaten unterstützt wird, genießt in Europa EDIFACT<sup>3</sup> eine größere Verbreitung.

### 3.1.2.4 World Wide Web Consortium (W3C)

Diese Organisation hat zwar nicht direkt mit der Entwicklung der Workflow-Management Sprachen zu tun, ist aber für die Standardisierung des Formats XML verantwortlich, das fast ausschließlich von den Organisationen als Austausch- und Beschreibungsformat verwendet wird.

---

<sup>3</sup> *Electronic Data Interchange For Administration, Commerce and Transport*: <http://www.unecce.org/trade/untdid/directories.htm> (20.02.2010)

## 3.2 Ablaufmodellierung

Als gemeinsame Sprache zwischen Menschen unterschiedlicher Branchen wurden immer Modelle verschiedenster Art verwendet. In den letzten zwanzig Jahren gab es im Bereich der Ablaufmodellierung zwei wesentliche Entwicklungen.

Zum einen gibt es da die ereignisgesteuerte Prozesskette, welche im Unterkapitel 3.2.1 und 3.2.2 beschrieben werden, und zum anderen den OMG Standard BPMN, welcher inzwischen eine weithin akzeptierte und verbreitete Spezifikationssprache ist (Nissen, Petsch, & Schorcht, 2008). BPMN wird im Unterkapitel 3.2.3 dargestellt.

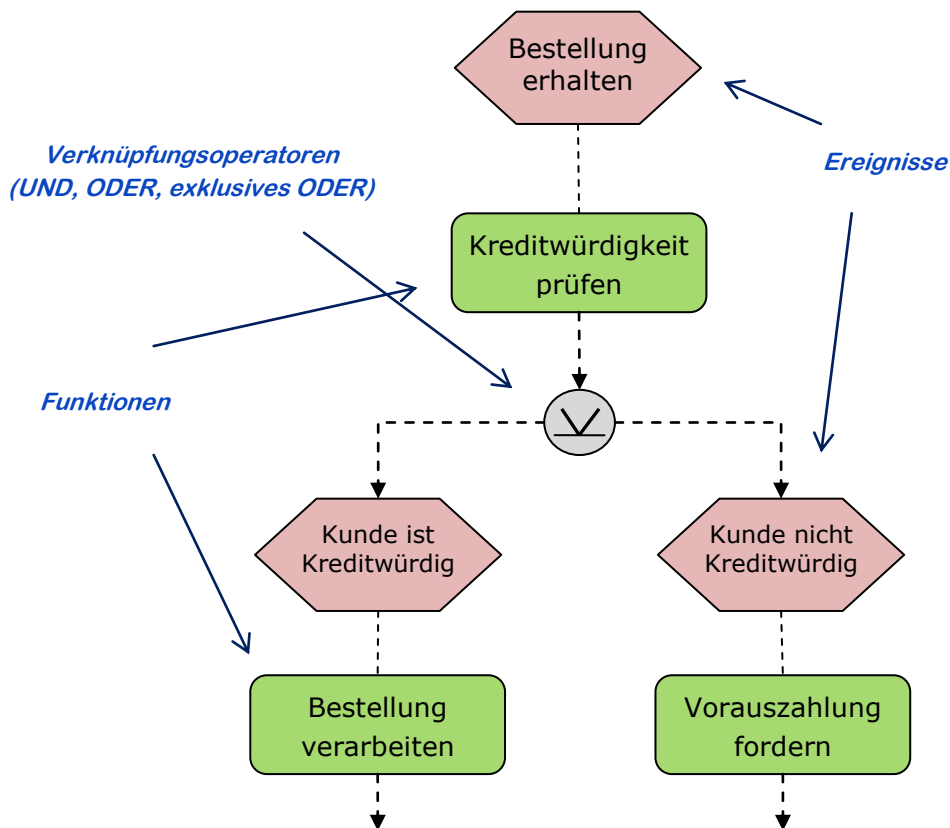
### 3.2.1 Event-driven Process Chain (EPC)

Die ereignisgesteuerte Prozesskette (EPK) ist eine grafische Modellierungssprache zur Darstellung von Geschäftsprozessen, welche 1992 von einer Arbeitsgruppe an der Universität des Saarlandes in Saarbrücken in Zusammenarbeit mit SAP AG entwickelt wurde. Die Methode ist ein wesentliches Element des ARIS-Konzeptes<sup>4</sup> welches ebenfalls von der gleichen Universität entwickelt wurde.

Die Notationselemente samt Erläuterungen werden in der Abbildung 12 dargestellt.

---

<sup>4</sup> Wikipedia - Ereignisgesteuerte Prozesskette:  
[http://de.wikipedia.org/wiki/Event-driven\\_Process\\_Chain](http://de.wikipedia.org/wiki/Event-driven_Process_Chain) (02. 11. 2009)



**Abbildung 12: EPC – Anwendung der Notation an einem einfachen Beispiel (Day, Bonati, Büttiker, & Lee, 2006)**

Die ereignisgesteuerte Prozesskette verwendet drei grundlegende Notationselemente. Das sind **Funktionen** und **Ereignisse** als grundlegende Objekte des Modelles, welche mit **Verknüpfungoperatoren** zu einem gerichteten Graphen verbunden werden.

Die Kette beschreibt, welche Ereignistypen welche Funktionstypen auslösen und welche Ereignistypen von welchen Funktionstypen erzeugt werden. Dadurch, dass ein Ereignistyp, der von einem Funktionstyp erzeugt wird, auch Auslöser für einen folgenden Funktionstyp ist, entsteht eine zusammenhängende Kette (Keller, Nüttgens, & Scheer, 1992).

### 3.2.1.1 Ereignisse

Wie beschrieben, basiert die Notation auf Ereignissen (engl. events), welche das Eintreten eines definierten Zustandes bedeuten und eine Folge von Aktivitäten bewirken. Beispiele für ein Ereignis sind:

- Auftrag ist eingetroffen
- Kundenanfrage ist abgelehnt
- Überweisung ist vorbereitet

Ereignisse repräsentieren also einen Zustand. Sie werden immer mit Funktionen verknüpft, entweder als Auslöser oder als Resultat. Jeder Geschäftsprozess beginnt und endet mit einem Ereignis (Day, Bonati, Büttiker, & Lee, 2006).

### *3.2.1.2 Funktionen*

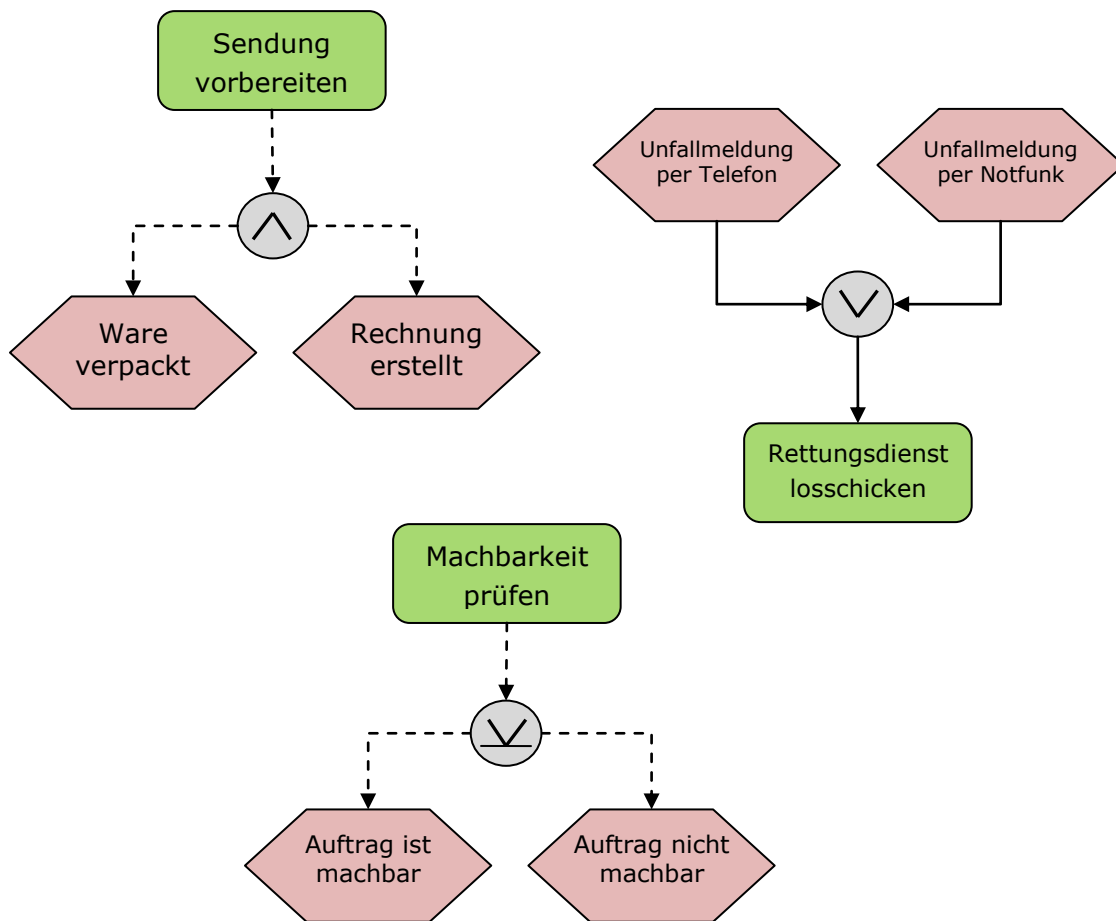
Eine Funktion kann in der Literatur nicht eindeutig erklärt werden, da sie sehr vom Kontext abhängt. In dieser Arbeit wird eine Funktion im Sinne einer Aufgabe gesehen. Eine Funktion kann in weitere Funktionen und Ereignisse unterteilt werden. D.h. ein EPK Modell kann in mehrere Detailstufen eingeteilt werden. Beispiele für Funktionen sind:

- Lagerbestand prüfen
- Bestellung erfassen
- Auftragsbearbeitung

Eine Funktion stellt eine Aktivität dar, welche von einer bestimmten Ressource (Person oder Computer) ausgeführt wird. Ausgelöst werden die Funktionen durch Ereignisse und produzieren wiederum als Resultat ein oder mehrere Ereignisse (Day, Bonati, Büttiker, & Lee, 2006).

### *3.2.1.3 Verknüpfungsoperatoren*

Das Modell unterstützt drei Verknüpfungsoperatoren: die konjunktive, disjunktive und die adjunktive Verknüpfung. Operatoren ermöglichen komplexere Strukturen (Keller, Nüttgens, & Scheer, 1992). Abbildung 13 zeigt die möglichen Verknüpfungen.



**Abbildung 13: EPC – Anwendungsbeispiele für Operatoren (Day, Bonati, Büttiker, & Lee, 2006)**

Der *UND* Operator zeigt, dass beide Pfade ausgeführt werden müssen. Beim *ODER* Operator sind es entweder beide oder zumindest einer der Pfade, die eingeschlagen werden müssen. Genau ein Pfad muss beim *exklusiven ODER* durchgeführt werden.

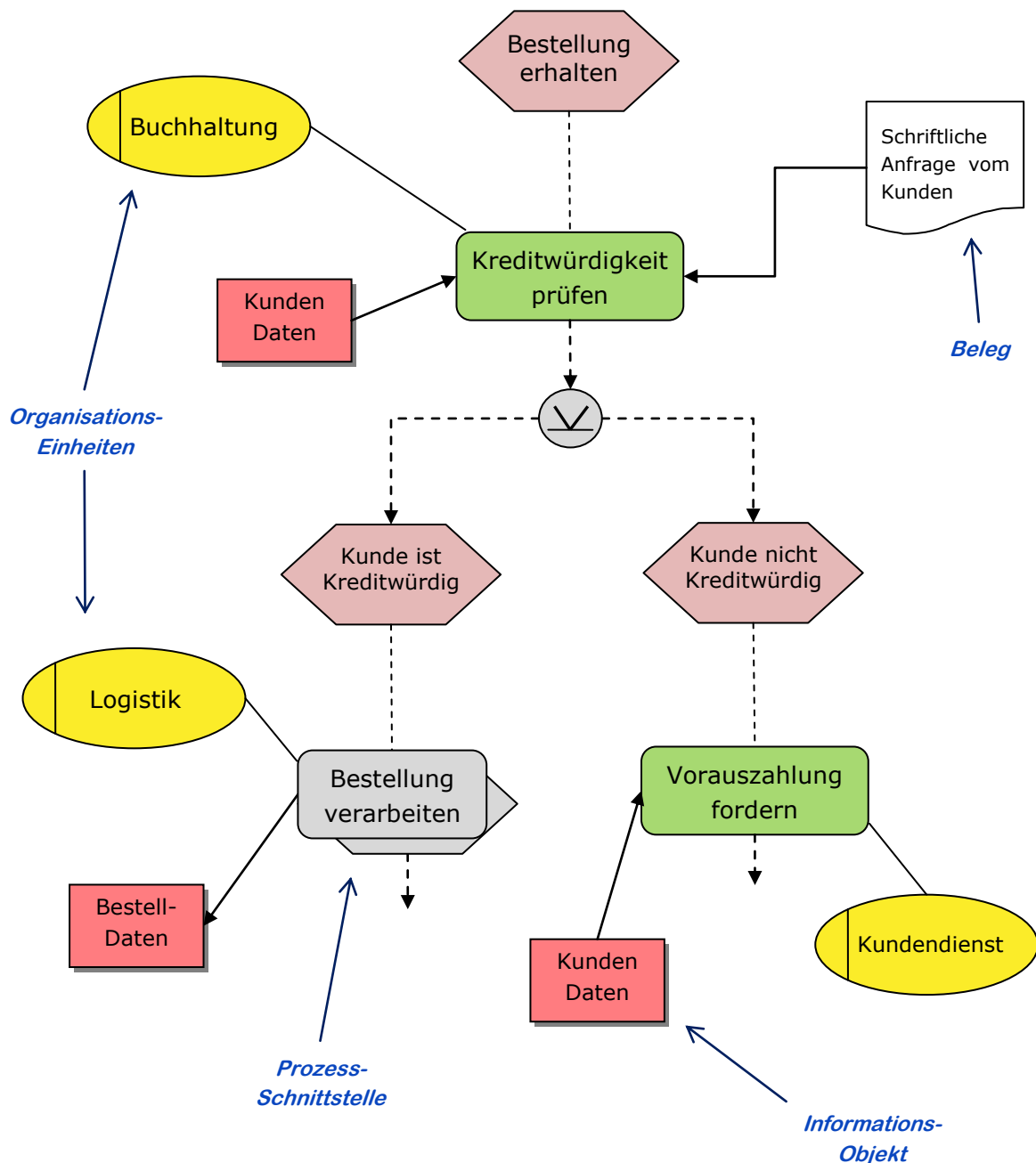
Insgesamt können dadurch zwölf verschiedene Verknüpfungen entstehen, bei denen aber zwei existieren, die ungültig sind:

1. Aus einem Ereignis können nie zwei Funktionen mit *ODER* verknüpft werden.
2. Aus einem Ereignis können nie zwei Funktionen mit *exklusiven ODER* verknüpft werden.

Der Grund liegt in der Natur der Sache. Ein Ereignis kann nicht entscheiden, welcher der beiden Wege ausgeführt werden soll (Day, Bonati, Büttiker, & Lee, 2006).

### 3.2.2 Extended Event-driven Process Chain (eEPC)

Die „erweiterte ereignisgesteuerte Prozesskette“ (eEPK) bringt weitere vier Notationselemente ein.



**Abbildung 14: eEPC – Das einfache Beispiel der EPC aus Abbildung 12 erweitert um Elemente der eEPC (Day, Bonati, Büttiker, & Lee, 2006)**

Die Elemente Organisationseinheit, Informationsobjekt und Beleg werden mit Funktionen verknüpft. Mittels Prozessschnittstellen können Verknüpfungen zu vor- bzw. nachgelagerten Prozessen dargestellt werden (Fischer, 2009).

### *3.2.2.1 Organisationseinheit*

Durch die Zuordnung von Organisationseinheiten zu Funktionen wird das Modell um den Aspekt der Organisationssicht erweitert. Es wird somit festgelegt wer die Funktion auszuführen hat (Fischer, 2009).

Typische Beispiele für Organisationseinheiten sind: das Personalwesen, der Vertrieb bzw. die Buchhaltung.

### *3.2.2.2 Informationsobjekt*

Die Informationsobjekte stellen in diesem Zusammenhang Mengen realer und abstrakter Dinge, die für ein Unternehmen von Interesse sind, dar (Keller, Nüttgens, & Scheer, 1992).

Beispiele für Informationsobjekte:

- Auftragsdatenbank
- Produktkatalog

Es können aber auch nicht elektronische Daten berücksichtigt werden. Mit einem Richtungspfeil zwischen den Informationsobjekt und der Funktion wird der Datenfluss angezeigt, da das Informationsobjekt sowohl als Ablage als auch als Quelle der Daten dienen kann (Day, Bonati, Büttiker, & Lee, 2006).

### *3.2.2.3 Belege*

Belege werden nicht in allen Quellen zu eEPKs gezeigt oder beschrieben. Sie stellen Beglaubigungen oder Dokumente dar, welche von einer bestimmten Funktion produziert oder als Input verlangt werden (Fischer, 2009). Das sind zum Beispiel:

- Bestellformular (als Input)
- Warengangbestätigung (als Output)

### 3.2.2.4 Prozesswegweiser

Im Grunde könnte der Prozesswegweiser auch durch eine Funktion ersetzt werden, da diese wiederum in weitere Funktionen und Ereignisse aufgeteilt werden kann - ähnlich einem Gruppierungselement.

Mit Prozesswegweisern an dieser Stelle ergibt sich ein viel transparenteres Modell als bisher und die Zusammenhänge verschiedener Geschäftsprozesse können klarer veranschaulicht werden. Ein Unterprozess kann somit auch an verschiedenen Orten eingesetzt werden.

## 3.2.3 Business Process Modeling Notation (BPMN)

Die *Business Process Modelling Notation* ist eine weitere grafische Spezifikationssprache, welche vom IBM-Mitarbeiter Stephen A. White entwickelt wurde. Die Veröffentlichung der Version 1.0 fand durch die „*Business Process Management Initiative*“ (BPMI) im Mai 2004 statt. Ein Jahr darauf fusionierten die Organisationen BPMI und OMG, so dass die Notation in weiterer Folge als ein OMG Standard galt. Diese Notation führte gleichzeitig zur einem bis dato neuen Impuls in der Unternehmensarchitektur, dem Prozessmanagement.

Im Wesentlichen bestimmt das BPMN einen Diagrammtyp – das „*Business Process Diagram*“ (BPD). Das erste Ziel dieser Notation ist es ein leicht leserliches und verständliches Modell für alle Anwender der Geschäftsfälle anzubieten. Sie soll also die Lücke zwischen dem Geschäftsprozessdesign und der Prozessimplementierung füllen. Die Spezifikationssprache ist auch mit dem Blick auf die Webdienste entwickelt worden und bietet weit mehr grafische Elemente zur Beschreibung der Abläufe als die EPK. Mit Hilfe dieser Objekte ist eine leichte Entwicklung von einfachen Diagrammen möglich, welche den Flussdiagrammen bzw. den Programmablaufplänen sehr ähnlich sehen und daher den meisten Anwendern bekannt sein dürften (White, Introduction to BPMN, 2004).

Die Unterscheidung in vier grundlegende Kategorien der Objekte trägt zur Übersichtlichkeit bei, gleichzeitig können durch hinzufügen ergänzender Informationen zu den Basisobjekten komplexere Aufgabenstellungen bewältigt werden, ohne das fundamentale „Look-And-Feel“ des Diagrammes zu ändern.

Die vier Basiskategorien der Objekte, wobei die letzteren zwei für Modelle mit höherer Modellierungsgenauigkeit verwendet werden, sind:

#### ➤ **Flow Objects**

Ablaufobjekte sind Aufgaben, Subprozesse, Ereignisse und Entscheidungspunkte.



➤ **Connection Objects**

Zu den Verbindungselementen gehören Ablafrichtung, Nachrichtenfluss sowie Zuordnungen.

➤ **Swimlanes**

Mit Verantwortlichkeitsbereichen (engl. Swimlanes) können Aktivitäten in verschiedene visuelle Kategorien unterteilt und den zugehörigen Systemteilnehmern zugeordnet werden.

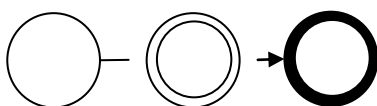
➤ **Artifacts**

Artefakte sind Annotationen, Beschreibungen, Datenobjekte und Gruppen.

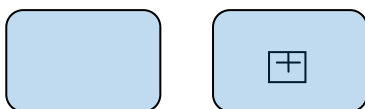
Die folgenden Unterkapitel und Beispiele verdeutlichen die einzelnen BPMN Objekte und deren Verwendung (White, Introduction to BPMN, 2004).

### 3.2.3.1 Ablaufobjekte (Flow Objects)

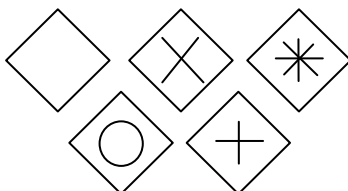
Zu den Elementen dieser Kategorie gehören Ereignisse (*Events*), Aktivitäten (*Activities*) und die Entscheidungspunkte (*Gateways*).



Ein Ereignis wird durch einen Kreis repräsentiert und ist etwas was „passiert“. Ereignisse haben üblicherweise eine Ursache und beeinflussen den Prozessfluss



Aktivitäten sind Rechtecke mit abgerundeten Kanten. Enthält eine Aktivität ein Kästchen mit einem Plus, so handelt es sich um einen Unterprozess.




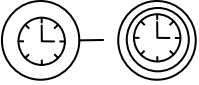
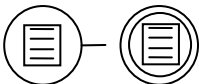


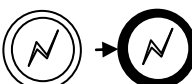

Mit einer Raute werden Entscheidungen dargestellt. Pfadzusammenführungen und Pfadabzweigungen können mittels Gateways auch realisiert werden.

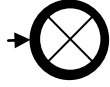
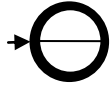
**Abbildung 15: BPMN – Ablaufobjekte**

Zusammen mit den Ereignissen stellen sie die wichtigsten Objekte der Modellierungssprache dar (White, Introduction to BPMN, 2004).

### Ereignisse

Genau wie in einem EPK Diagramm beginnt und endet ein BPD mit einem Ereignis. In der Notation wird zwischen drei Ereignistypen unterschieden: Start-, Zwischen- und Endereignis. Die Einteilung ist deshalb so wichtig, da die Notation über neun verschiedene Ereignisauslöse-Arten kennt, wobei für gewisse Arten Beschränkungen bestehen. In der folgenden Tabelle werden die einzelnen Ereignisse beschrieben.

Ereignis (Start - Intermediate - End)	Beschreibung
	<p>Eine Nachricht kann einen Prozess auslösen, diesen nach einer Unterbrechung fortführen oder als Ausgang dieses Prozesses dienen.</p>
	<p>Timer können Ereignisse zu einem bestimmten Zeitpunkt oder in einem Zeitzyklus auslösen. Ein Timer kann kein Endereignis hervorrufen.</p>
	<p>Das Ereignis tritt durch das Erfüllen bestimmter Regeln ein. Regelbasierte Ereignisse können genauso wenig wie die Zeitbasierten Endereignisse auslösen.</p>
	<p>Der Link verbindet das Endereignis eines Prozesses mit dem Startereignis eines weiteren Prozesses.</p>
	<p>Zeit an wenn für ein Ereignis mehrere Auslöser in Frage kommen. Durch Attribuierung wird festgestellt welche Auslöser in Frage kommen</p>
	<p>Ausnahme-Ereignis (Exception) weist darauf hin, dass ein Fehler generiert werden sollte. Logischer weise kann eine <i>Exception</i> kein Startereignis sein.</p>
	<p>Abgleich-Ereignis wird von einem Ausnahme-Ereignis verwendet, wenn der Ablauf zurückgerollt bzw. wiederholt wird.</p>

<i>Ereignis (Start - Intermediate - End)</i>	<i>Beschreibung</i>
	Der Benutzer bricht den Prozess ordentlich (mit Ereignisbehandlung) ab.
	„End-Kill“-Ereignis im Falle von fatalen Fehlern zeigt an, dass alle Aktivitäten des Prozesses augenblicklich beendet werden ohne Ereignisbehandlung.

**Tabelle 2: BPMN – Ereignisauslöser (Owen & Raj, 2003)**

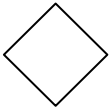

### Prozesse, Sub-Prozesse und Aktivitäten

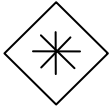
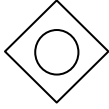

Genauso wie in EPK, spielen auch in BPMN die Aktivitäten und Prozesse die wichtigste Rolle. Sie beschreiben die eigentlichen Tätigkeiten in dem Prozessfluss. Die zwei grafischen Elemente zeigen lediglich die hierarchische Abhängigkeit. Ist in einem Rechteck ein weiteres kleines Rechteck mit einem Pluszeichen vorhanden, wird dadurch ein Unterprozess symbolisiert, welcher in einem eigenen Diagramm dargestellt. Im anderen Fall handelt es sich um eine atomare Tätigkeit, welche nicht weiter modelliert wird (White, Introduction to BPMN, 2004).

Auf diese Art ist es möglich sehr komplexe Systeme in mehreren Ebenen zu modellieren, wobei in jeder weiteren Ebene mehr Details gezeigt werden.

### Entscheidungspunkte

Mit dem sogenannten Gateway Symbol werden Entscheidungen getroffen. Im Prozessfluss kann somit eine Pfadverzweigung oder Pfadzusammenführung stattfinden. Insgesamt stehen hier fünf grafische Objekte zur Verfügung. Diese sind in der Tabelle 3 beschrieben.

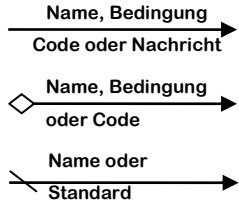
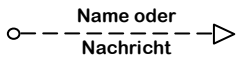
<i>Gateway</i>	<i>Beschreibung</i>
	XOR-Gateway – Datenbasierend. Ausgang dieses Gateways kann nur eine Aktivität sein. Diese Art gehört zu den häufigsten benutzten Gateways.
	XOR-Gateway – Eventbasierend. Ist eine neuere Art der Entscheidungen im BPMN und führt zu genau einem Ereignis.

Gateway	Beschreibung
	Definiert eine komplexe Bedingung, anhand welcher der Ausgangspfad berechnet wird.
	Inklusiv-Gateway bedeutet, dass mindestens einer der Ausgangspfade aktiviert wird. Dafür muss ein Standardpfad definiert werden.
	Parallel-Gateway wird auch „UND“-Gateway genannt. Alle Richtungen, die hier rausgehen werden durchschritten. Auf der anderen Seite, wird auf alle Eingangssignale gewartet bis das Gateway durchschritten wird.

**Tabelle 3: BPMN – Gateways (Owen & Raj, 2003)**

### 3.2.3.2 Verbindungselemente (Connecting Objects)

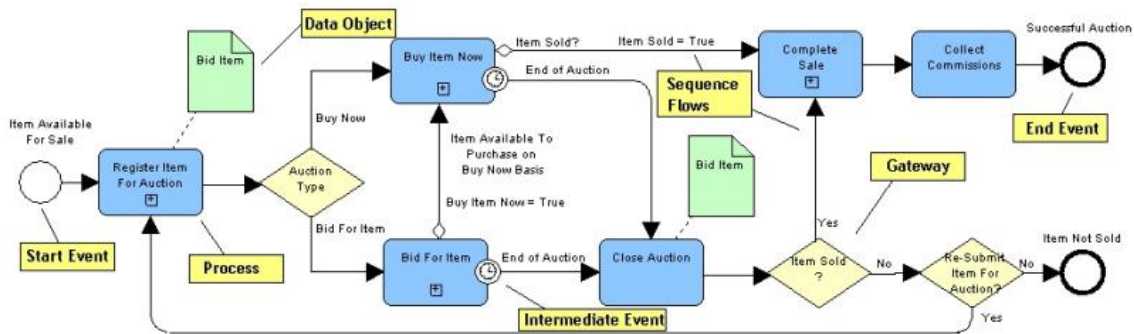
Zusammen mit den Ablaufelementen bilden die Verbindungselemente das Grundgerüst eines BP-Diagramms. Die drei Verbindungskategorien sind in der folgenden Abbildung dargestellt (BPMN).

Verbindungstyp	Beschreibung	Grafikelement
<p><b>Sequence Flow</b></p> <p><b>Conditional Flow</b></p> <p><b>Default Flow</b></p>	<p>Innerhalb eines Prozesses oder einer Organisation zeigen diese Pfeile die Reihenfolge in welcher Tätigkeiten ausgeführt werden. Der <i>Conditional Flow</i> ist dabei an eine Bedingung geknüpft und der <i>Default Flow</i> wird gewählt, falls kein anderer Pfad durchlaufen werden kann.</p>	
<b>Message Flow</b>	<p>Die Nachrichten werden immer zwischen zwei Prozessteilnehmern (als <i>Pools</i> dargestellt) gesendet und empfangen.</p>	

Verbindungstyp	Beschreibung	Grafikelement
<b>Association</b>	Dieser Verbindungstyp verbindet verschiedene <i>Artifacts</i> mit den Ablaufobjekten.	..... .....>

**Tabelle 4: BPMN – Verbindungselemente**

Mit den ersten zwei Elementkategorien ist es bereits möglich einfache BPDs zu modellieren. Die Abbildung 16 verdeutlicht das an Hand eines Beispiels einer Online-Auktion.

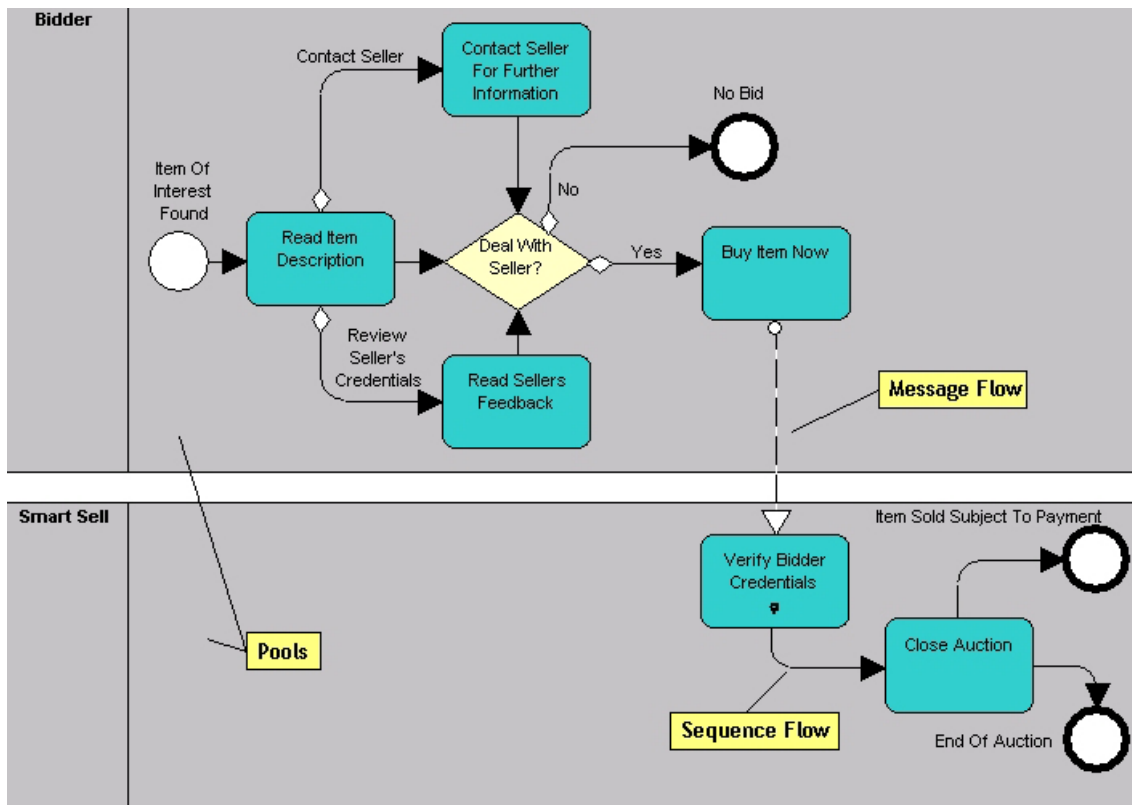


**Abbildung 16: BPMN – Ein einfaches BPMN Beispiel (Owen & Raj, 2003)**

Modelle, welche ein höheres Niveau an Genauigkeit verlangen – angenommen um Teilnehmer oder Datenverbindungen darstellen zu können, brauchen die zwei folgenden Objektkategorien.

### 3.2.3.3 Verantwortlichkeitsbereiche (Swimlanes)

Die Verantwortlichkeitsbereiche ermöglichen eine Sicht auf die Organisationsstrukturen bzw. teilnehmende Personen oder Abteilungen im Bezug auf den Prozess. Die Aktivitäten können nach Verantwortung und Zuständigkeit visuell aufgeteilt werden. Zu diesem Zweck gibt die Objekte *Pools* und *Lanes* (Owen & Raj, 2003).



**Abbildung 17: BPMN – Verantwortlichkeitsbereiche**

Bezogen auf das Beispiel vom Beginn des Kapitels, können die Aktivitäten wie in Abbildung 17 aufgeteilt werden.

### Pools

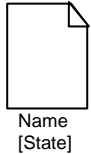

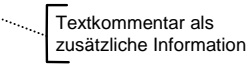
Die *Pools* bezeichnen die Teilnehmer des Prozesses und fungieren als grafische Container insbesondere bei *Business-to-Business* Situationen. Sie sind grafisch voneinander getrennt und deren Kommunikation wird mit Nachrichtenflüssen (*Message Flows*) dargestellt. Andere Kommunikationsmöglichkeiten bzw. Verbindung sind hier nicht erlaubt (White, Introduction to BPMN, 2004).

### Lanes

Die *Lanes* sind Organisations-Unterelemente innerhalb der *Pools* und weisen die Tätigkeiten zum Beispiel bestimmten Abteilungen zu oder binden diese an Funktionen oder Rollen im Unternehmen (White, Introduction to BPMN, 2004).

### 3.2.3.4 Artefakte (Artifacts)

Mit der Einführung von Artefakten dehnt BPMN die Standardfunktionalität aus und wird auch flexibler im Bezug auf zusätzliche Modellierungssituationen. BPMN definiert drei Artefakt-Typen vor, tatsächlich können aber beliebige kontextbezogene Artefakt-Objekte verwendet werden.

Artefakt	Beschreibung	Grafikelement
<b>Data Object</b>	Datenobjekte lassen besser erkennen welche Daten von den Aktivitäten gebraucht werden bzw. welche Daten als Ergebnis produziert werden. Die Datenobjekte verknüpft man mittels Assoziationen an die jeweiligen Aktivitäten oder Prozesse.	
<b>Group</b>	Das Gruppierungselement hat keinen Einfluss auf die Ablaufsteuerung des Prozesses, sonder dient dem Dokumentations- und dem Analyse-Zweck	
<b>Annotation</b>	Die textuellen Beschreibungen setzten da an, wo die grafischen Elemente nicht genug Aussagekraft haben. Modelle werden klarer und Missverständnisse können minimiert werden.	

**Tabelle 5: BPMN – Artefakte (White, Introduction to BPMN, 2004)**

Modellierer können also eigene Artefakte hinzufügen, ohne dass dabei die grundlegende Form des BP-Diagramms beeinflusst wird.

### 3.2.3.5 BPMN vs. UML -> BPMN und UML

Der essentielle Unterschied dieser zweier Notationen besteht wohl im Ansatz zur Modellierung der Geschäftsprozesse. UML bietet eine objektorientierte Sicht der Dinge, welche als Grundlage ein Klassendiagramm mit involvierten Objekten verlangt.

In weiterer Folge wird dann mittels Verhaltensdiagrammen wie dem Aktivitäten- (engl. Activity Diagram) und dem Anwendungsfalldiagramm (engl. Use-Case Diagram) gezeigt, wie diese Objekte unter einander interagieren.

Diese Herangehensweise ist den meisten Geschäftsprozessanalytikern fremd und erscheint ihnen unlogisch. Die prozessorientierte Darstellung im BPMN hingegen ist natürlicher und intuitiver. Mit dieser Notation werden zunächst die Kontroll- und die Nachrichtenflüsse der Prozesse modelliert. Das Objektmodell wird dadurch implizit definiert.

UML kann auch als eine Ansammlung diverser Modelle und Diagramme, welche sich in der Praxis bewährt haben, gesehen werden. Die einzelnen Diagramme wurden nicht speziell dafür entwickelt, um miteinander zu kooperieren. Das heißt, die Designer können nur Teile ihrer Applikation modellieren, wobei die Details auf der Strecke bleiben. Somit definiert UML auch keine Ausführungs-Metadaten für Geschäftsprozesse.

Hingegen ist BPMN „aus einem Guss“ und genau für solche Fälle konzipiert. Aus den Modellen kann automatisch und ohne weiteres Zutun eine ausführbare Definition wie BPEL erzeugt werden. In der Zukunft wird also eine Zusammenarbeit dieser beiden Notationen angestrebt (Owen & Raj, 2003).

Lesenswert zu diesem Thema ist eine Veröffentlichung von Wil van der Aalst, Arthur ter Hofstede, Bartek Kiepuszewski und Alistair Barros, welche BPMN und die UML Diagramme anhand 21 ausgewählter Workflow Patterns analysiert. Die Patterns reichen von einfachen bis sehr komplexen und wurden von den Forschern auf einer Webseite mit Beispielen und zusätzlichen Dokumenten präsentiert (White, Process Modeling Notations and Workflow Patterns, 2004).

### **3.3 Ablaufbeschreibung bzw. -definition**

Mit dem vorhergegangenen Kapitel wurde gezeigt, wie Geschäftsprozesse grafisch dargestellt werden können, um sie Fachleuten, Abteilungsleitern oder anderen Entscheidungsträgern, die das Programmieren und die Entwicklung von Software nicht beherrschen bzw. verstehen müssen, zu präsentieren. Wie auch immer, die grafische Notation allein reicht natürlich nicht aus, um diese Prozesse maschinell bearbeiten zu können.

Die folgenden Abschnitte zeigen eine Auswahl derzeitiger Standards, welche Geschäftsprozesse maschinenlesbar beschreiben. Wie auch immer sich die einzelnen Standards unterscheiden, eines haben Sie alle gemeinsam. Alle basieren auf der Extensible Markup Language (XML).



### 3.3.1 Electronic Business using XML (ebXML)

Im November 1999 startete die Organization for the Advancement of Structured Information Standards (OASIS) und das United Nations Centre of Trade Facilitation and Electronic Business (UN/CEFACT) das Projekt Electronic Business using XML, kurz ebXML. Über 100 teilweise namhafte Unternehmen wie IBM oder Sun und viele internationale Standardisierungsorganisationen aus insgesamt 14 Ländern waren involviert. Die Entwicklung wurde in zwei Phasen aufgeteilt. Mit Ende der zweiten Phase im Herbst 2003 wurde ebXML für technisch vervollständigt und beendet erklärt.

Die Entwicklung verfolge zwei wesentliche Ziele:

1. Den Entwurf einer offenen technischen Spezifikationen für einen weltweit einheitlichen Austausch elektronischer Geschäftsdaten auf Grundlagen von XML, und
2. Die Senkung der Eintrittsbarrieren zum E-Business für kleine und mittelständische Unterhemen als auch für Entwicklungsländer.

Aus den Anforderungsspezifikationen, welche die Visionen und Ziele beinhalten, liest sich folgender Satz:

*„The ebXML vision is to deliver: A single set of internationally agreed upon technical specifications that consist of common XML semantics and related document structures to facilitate global trade.“ (ebXML Requirements Team, 2001)*

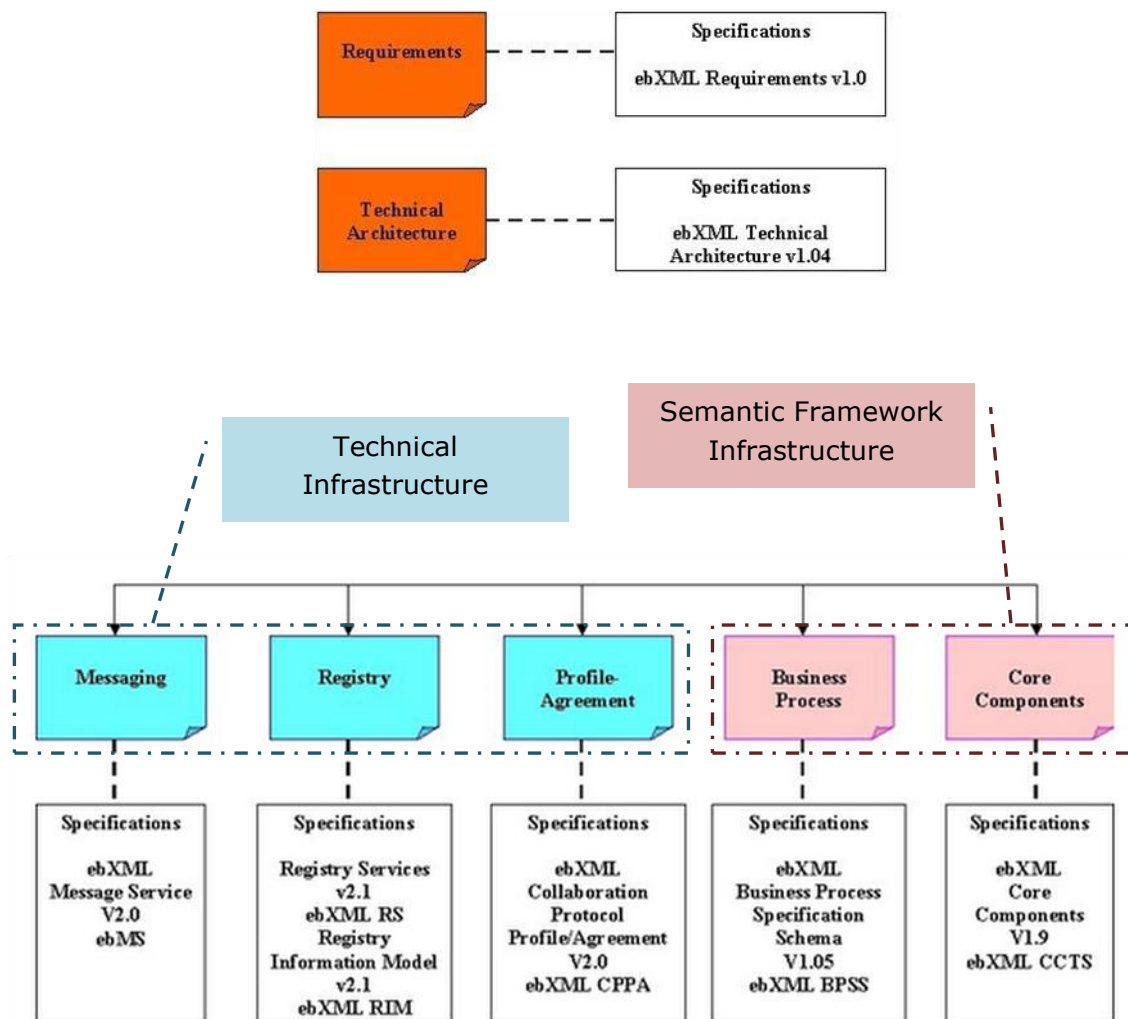
Das Ergebnis ist also eine modulare Spezifikationsfolge bzw. einer Familie an Standards, welche es Unternehmen jeglicher Größe und geografischer Lage ermöglichen soll ihre Geschäftsprozesse über das Internet abzuwickeln (ebXML, 2006).

#### 3.3.1.1 Die Architektur und die Komponenten

ebXML ist also ein Framework, welches eine Menge an Spezifikationen beinhaltet, mittels denen verschiedene Geschäftsbeziehungen abgewickelt werden können. Dafür stehen folgende Komponenten zur Verfügung (Löschnigg, 2005):

- Komponente zur Beschreibung von Geschäftsprozessen und allen damit verbundenen Informationen (**Business Process Specification**)
- Komponente zur Verwaltung der ebXML-Geschäftsprozesse (**Registry**)

- Mechanismus zur Speicherung und Registrierung der Informationen
- Mechanismus zum Wiederfinden der Informationen, wie z.B.:
  - Geschäftsprozesse, die unterstützen werden,
  - Schnittstellenbeschreibungen,
  - Geschäftsnachrichtentypen,
  - Informationen über implementierte Transport-, Sicherheits- und Verschlüsselungsprotokolle.
- Komponente zur Beschreibung der Vereinbarung von Geschäftsbeziehungen (**Profile Agreement**)
- Ein normierter geschäftlicher Nachrichtenservice-Rahmen (**Messaging Service**)
  - Konfiguration des Messaging Services, wie in der Geschäftsvereinbarung beschrieben
  - Sicherer und zuverlässiger Nachrichtenaustausch zwischen den Handelspartnern
- Spezifikation der Architektur und der Beziehungen der Konzepte in zwei Kategorien
  - Semantische Infrastruktur (**Semantic Framework Infrastructure**)
  - Technische Infrastruktur (**Technical Infrastructure**)

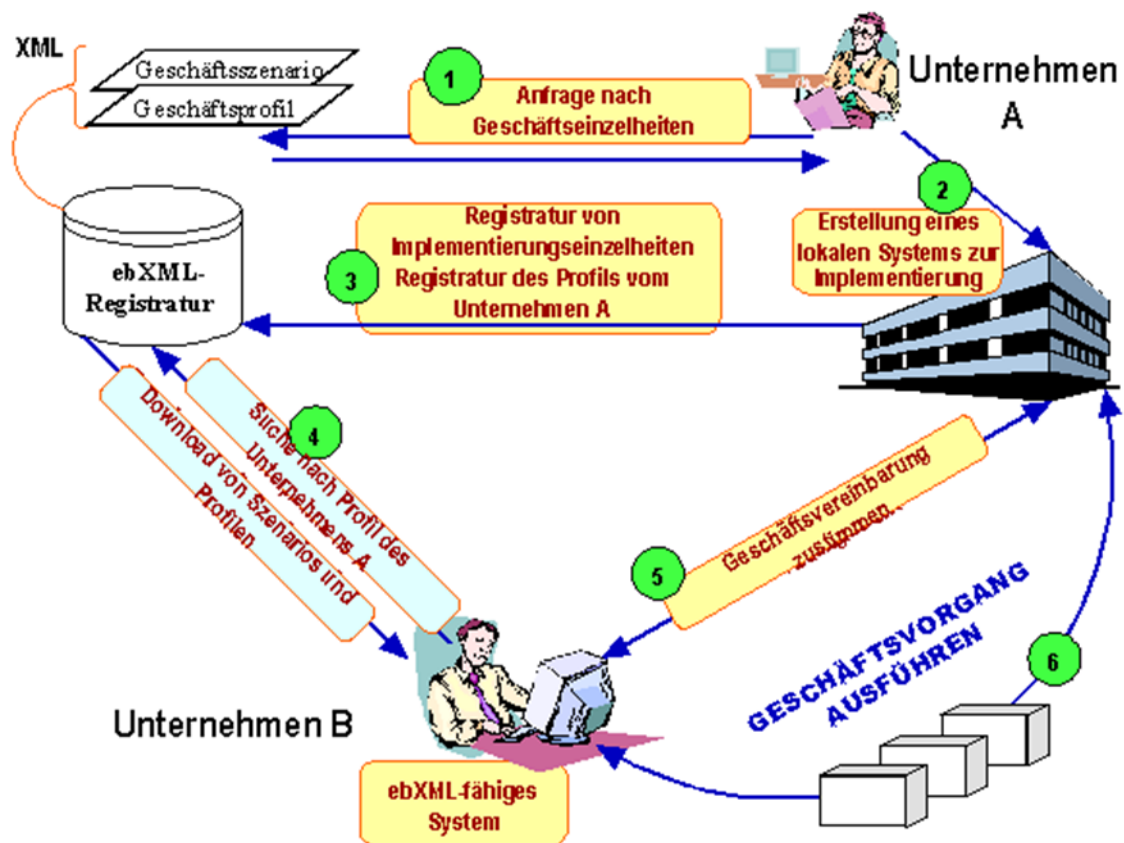


**Abbildung 18: ebXML – Komponentenarchitektur (Europäisches ebXML-Informationszentrum, 2009)**

Das Zusammenspiel der Komponenten wird im folgenden Unterkapitel anhand eines praktischen Szenarios dargestellt.

### 3.3.1.2 Das Anwendungsszenario

Um das Zusammenspiel der Komponenten zu veranschaulichen, wird das Beispiel aus der technischen Spezifikation der ebXML Architektur (ebXML Technical Architecture Project Team, 2001) dargestellt. Das Szenario in der Abbildung 19 zeigt die notwendigen Schritte zweier Handelspartner, die sich entscheiden ebXML zu verwenden.



**Abbildung 19: ebXML – Anwendungsszenario (Europäisches ebXML- Informationszentrum, 2009)**

Im ersten Schritt erfährt das Unternehmen A von einer ebXML-Registratur. Im zweiten Schritt prüft das Unternehmen die Geschäftseinheiten und entscheidet ebenfalls eine ebXML-fähige Anwendung einzusetzen. Hierbei gibt es die Möglichkeit die Software selbst zu entwickeln oder kommerzielle, auf das Unternehmen zugeschnittene Lösungen zu verwenden.

Der dritte Schritt ist die Registrierung des Firmenprofils samt Datenstruktur, Implementierungsangaben und Referenzen. Somit können andere Unternehmen nach Geschäftsprofilen und unterstützten Geschäftsszenarien suchen. Wird das Format und die Verwendung der Geschäftsszenarien beglaubigt, erhält Unternehmen A eine Bestätigung. Die ersten drei Schritte sind also die Konfiguration des Systems. Daraufhin folgt die eigenlichte Anwendung des Systems.

Das Unternehmen B sucht in der Registry nach einem Handelspartner und findet das Unternehmen A (Schritt 4). Das Unternehmen B fordert das Profil von A an und erwirbt eine ebXML-fähige, zugeschnittene Anwendung. Einigen sich die Partner im sechsten und letzten Schritt über die beidseitig vereinbarten Geschäftsszenarien und spezifischen Vereinbarungen, sind die beiden Unternehmen bereit „electronic Business“ mittels ebXML einzusetzen.

### 3.3.2 XML Process Definition Language (XPDL)

Die aktuelle Spezifikation der *XML Process Definition Language*<sup>5</sup> liegt derzeit in der Version 2.1 vor, welche im April 2008 verabschiedet wurde. Die wichtigsten Aufgaben dieses XML-Formats liegen im Beschreiben, Erstellen und Austauschen von Geschäftsprozessen. Ein besonderes Augenmerk wurde auf die Erweiterbarkeit geworfen, denn das Ziel war es einen einheitlichen Standard zu schaffen, der den Anforderungen beteiligter Unternehmen und Organisationen gerecht wird.

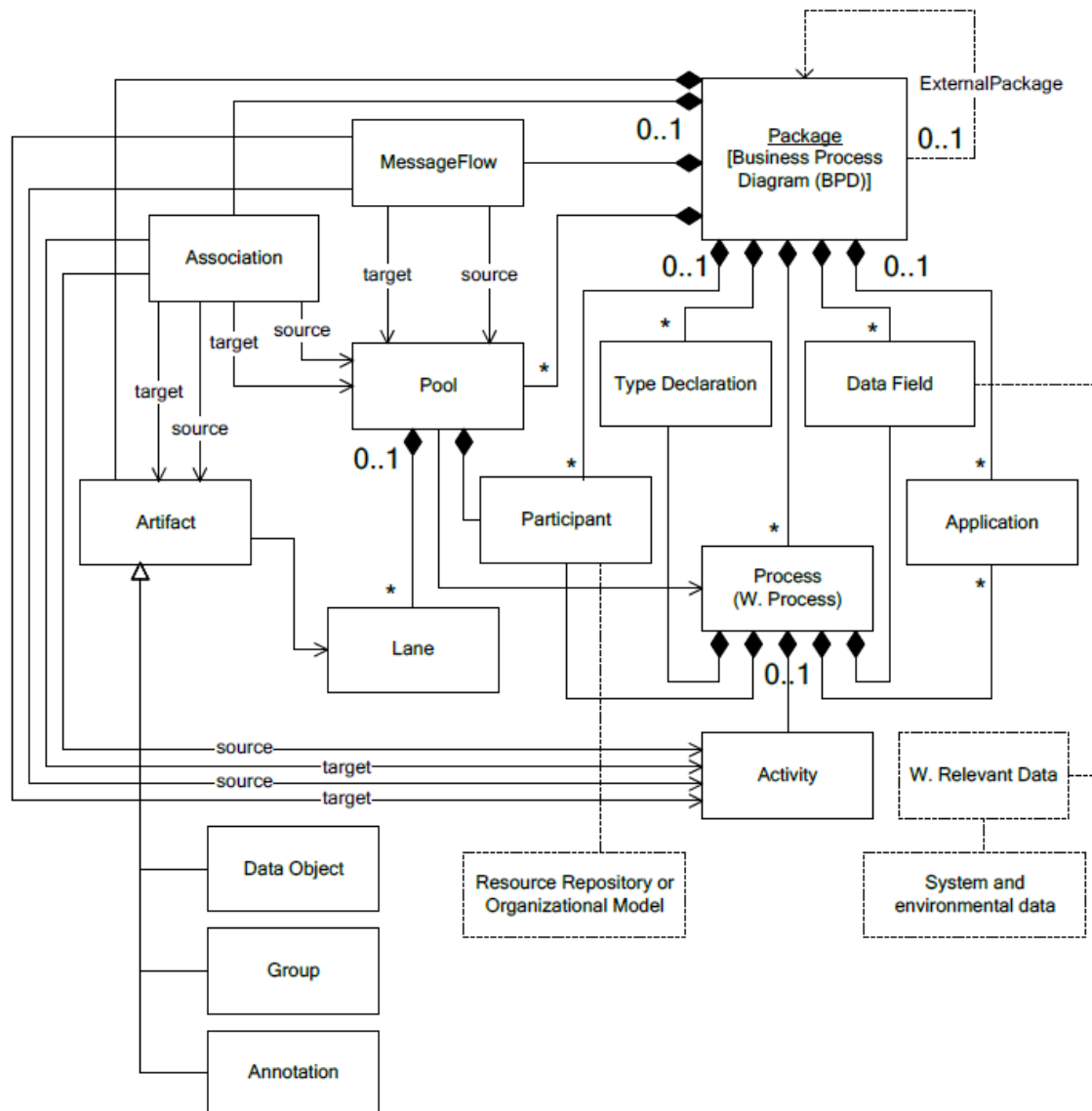
XPDL stellt also Meta-Modelle zur Verfügung, welche die Top-Level Elemente für die Prozessdefinition beschreiben. Die Modelle definieren Relationen, Attribute und Simulationsparameter sowie Gruppierungselemente und verwandte Prozessdefinitionen, um nur einige der Elemente zu nennen. Im Folgenden werden die Elemente beschrieben (Workflow Management Coalition, 2008).

#### 3.3.2.1 Pakete und das Paket-Meta-Modell

Die Pakete (Packages) dienen als Container und haben die Funktion individuelle, aber zusammengehörende Prozessdefinitionen sowie gemeinsam verwendete Entitäten dieser Definitionen zusammenzufassen.

---

<sup>5</sup> Wikipedia - XPDL: <http://en.wikipedia.org/wiki/Xpdl> (03.11.2009)



**Abbildung 20: XPD – Package Meta-Model (Workflow Management Coalition, 2008)**

Zu den Entitäten, welche einen paketübergreifenden Gültigkeitsbereich haben, gehören:

➤ **Ressourcen** (Participant Declaration)

Ressourcen werden als Attribute den Aktivitäten zugeordnet und können diese ausführen. Eine Ressource kann eine Einzelperson, eine Gruppe an Personen, eine bestimmte Fähigkeit oder eine maschinelle Ressource sein.

➤ **Applikationsbeschreibungen** (Application Declaration)

Applikationsbeschreibungen sind Schnittstellen zu Tools, Services oder einfachen Prozeduren innerhalb des Workflow-Management-System-Frameworks welche als Hilfs- oder Automatisierungsfunktionen ausgeführt werden sollen. Diese Elemente werden den Aktivitäten als Attribute samt zugehörigen Parametern zugeordnet.

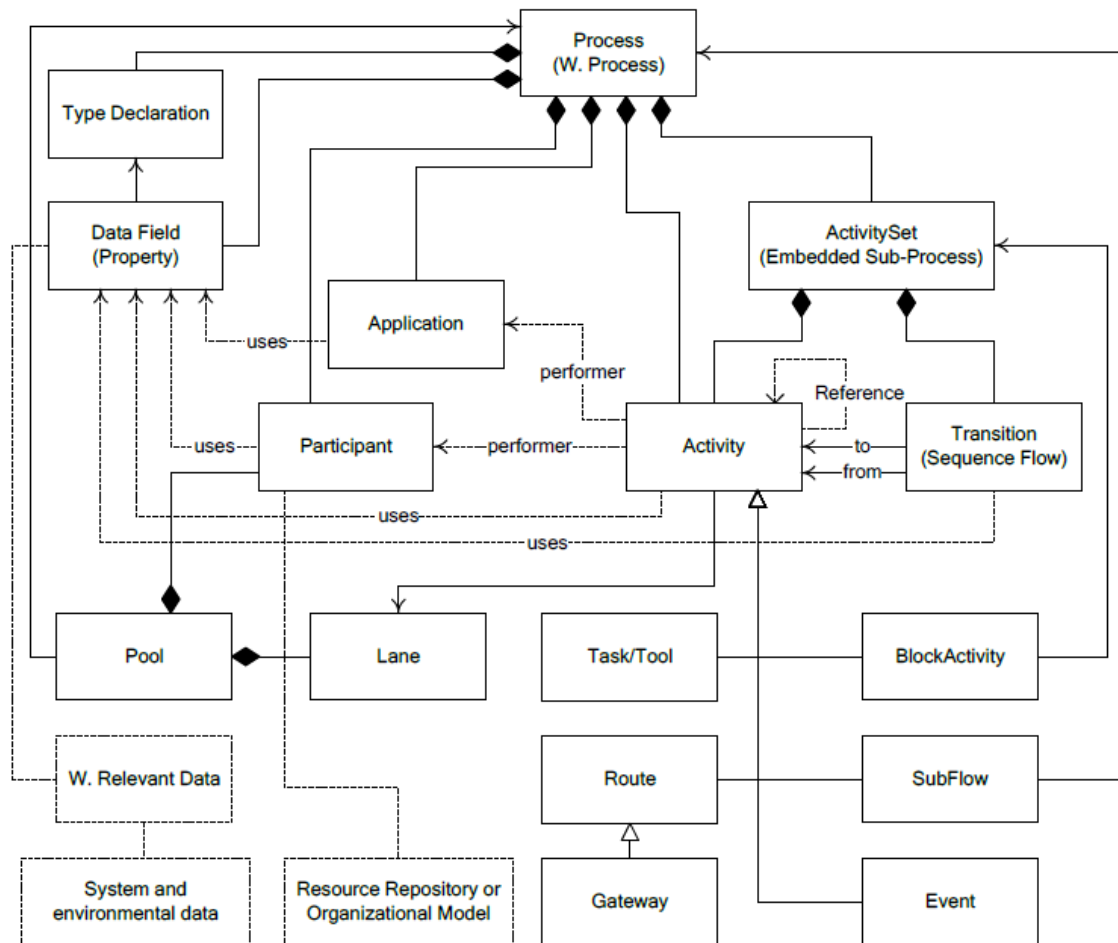
➤ **Datenbereich** (Relevant Data Field)

Der Datenbereich entspricht einer BPMN Property. Das sind Daten welche während einer Prozessausführung verwendet werden, sowie Informationen, die mit den Ressourcen ausgetauscht werden. Diese sogenannten „relevanten Daten“ sind von einem bestimmten XPDL Typ, welcher weiter unten beschrieben wird.

Diese Entitäten werden nur einmal im Paket definiert und können von verschiedenen Prozessbeschreibungen des Paketes referenziert werden. Des Weiteren ermöglicht das Paket die Definition gemeinsamer Attribute, welche von den enthaltenden Prozessen geerbt werden können.

### *3.3.2.2 Prozessdefinition und das Prozess-Meta-Modell*

Ein XPDL Prozess entspricht einem BPMN Prozess (siehe Unterkapitel 3.2.3.1). Das Meta-Modell (Process Meta-Model) zeigt die fundamentalen Elemente einer Prozessdefinition. Diese Elemente müssen entweder auf Prozessebene oder durch Vererbung auf Paketebene deklariert werden.



**Abbildung 21: XPDL – Prozess Meta-Modell**

Die Prozessdefinition beschreibt außerdem kontextbezogene Informationen wie Autor und Erstellungsdatum. Des Weiteren werden auch Daten (Startparameter, Ausführungspriorität, Dauer,...), welche bei der Ausführung relevant sind, gespeichert.

### 3.3.2.3 Aktivitäten und Aktivitätenset

Eine Prozessdefinition enthält eine Menge an Aktivitäten (ActivitySet), welche aus einem oder mehreren Aktivitäten besteht. Eine Aktivität ist eine autonome Aufgabeneinheit. Eine typische Aktivität beschreibt eine Aufgabe, welche in Kombination mit einer Ressource und/oder einer Computerapplikation ausgeführt wird.

XPDL unterscheidet fünf verschiedene Typen von Aktivitäten:



➤ **Keine Aktivität** (None Activity)

➤ **Untergeordnete Aktivität** (Sub flow Activity)

In diesem Fall ist die Aktivität ein Container für eine weitere Prozessdefinition, die ausgeführt werden muss bevor der eigentliche Prozess weiter laufen kann. Im BPMN ist das ein „Reusable subprocess“.

➤ **Geblockte Aktivität** (Block Activity)

Diese Aktivität ist ein Container für eine weitere Menge an Aktivitäten. Es entspricht dem BPMN „embedded subprocess“.

➤ **Verzweigungsaktivität** (Route Activity)

Diese Aktivität führt keine Aufgabe durch und dient rein zur Ablaufplanung. Bei der ersten Möglichkeit (join) führen zwei oder mehrere Aktivitäten zu einer Nachfolgenden und bei der zweiten Möglichkeit (split) folgen nach einer Aktivität zwei oder mehrere weitere Aktivitäten.

➤ **Ereignis Aktivität** (Event Activity)

Zu guter Letzt kann eine Aktivität auch ein BPMN Ereignis repräsentieren. Ein Ereignis beeinträchtigt den Prozessablauf und hat üblicherweise einen Auslöser und eine Auswirkung.

Für jede dieser Aktivitätstypen gibt es auch entsprechende Symbole in der BPM Notation (siehe Unterkapitel 3.2.3.1).

### 3.3.2.4 Zustandsübergänge

Zwischen zwei Aktivitäten befindet sich immer eine Übergangsinformation (transition information). Dieses Element beschreibt durch drei Attribute, wie der Prozessablauf von Aktivität A zur Aktivität B übergeht. Neben Anfangs- und der Endaktivität wird noch die Bedingung festgelegt. Folgende vier Bedingungstypen gibt es (Bartonitz, XPD - XML Process Definition Language, 2007):

➤ **Condition** ... die Transition wird bei Eintreten der Bedingung ausgeführt,

➤ **Otherwise** ... die Transition wird genau dann ausgeführt, wenn keine der angegebenen Bedingungen zutrifft,

➤ **Exception** ... die Transition wird ausgeführt, falls eine Ausnahmebedingung auftritt,

- **DefaultException** ... die Transaktion erfolgt, wenn alle anderen Exception-Bedingungen als nicht erfüllt sind.

### 3.3.2.5 Teilnehmer

Teilnehmer des Geschäftsprozesses sind Ressourcen, die eine bestimmte Aktivität ausführen. Es handelt sich dabei nicht zwangsläufig um eine Person, sondern auch um andere Systeme oder Rollen innerhalb einer Organisation. Die XPDL unterscheidet folgende Typen (Workflow Management Coalition, 2008):

- **RESOURCE\_SET** ... ist eine Menge von Ressourcen,
- **RESOURCE** ... beschreibt explizit eine Maschine,
- **ROLE** ... ist eine Funktion einer Person innerhalb einer Organisation,
- **ORGANIZATIONAL\_UNIT** ... ist eine Abteilung innerhalb einer Organisation,
- **HUMAN** ... beschreibt eine Person ohne besonderer Qualifikationen oder Funktionen,
- **SYSTEM** ... ist eine selbstständig arbeitende Maschine.

### 3.3.2.6 Weitere Elemente

Mit der Version 2.0 sind in der Absprache zwischen WfMC und der OMG die Entitäten Pool, Lane, Gateway und Event hinzugekommen, die den bidirektionalen Austausch zwischen XPDL und BPMN verbessern sollen.

Package	Workflow Process	Activity	Transition	Application	Data Field (Workflow Relevant Data)	Participant
- Id	- Id	- Id	- Id	- Id	- Id	- Id
- Name	- Name	- Name	- Name	- Name	- Name	- Name
- Description	- Description	- Description	- Description	- Description	- Description	- Description
- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes	- Extended Attributes
- XPDL Version	- Creation Date	- Automation Mode			- Data Type	- Participant Type
- Source Vendor ID	- Version	- Split				
- Creation Date	- Author	- Join				
- Version	- Codepage	- Priority				
- Author	- Country Key	- Limit				
- Codepage	- Publication Status	- Start Mode				
- Country Key	- Priority	- Finish Mode				
- Publication Status	- Limit	- Deadline				
- Conformance Class	- Valid From Date					
- Priority Unit	- Valid To Date					

**Tabelle 6: XPDL – Übersicht der Attribute der wichtigsten XPDL-Entitäten (Bartonitz, XPDL - XML Process Definition Language, 2007)**

Jede Entität verfügt noch über das Element *ExtendedAttribute*, welches zur freien Verwendung steht, womit auch die Zielsetzung der Flexibilität erreicht wird.

### 3.3.3 Business Process Execution Language (BPEL)

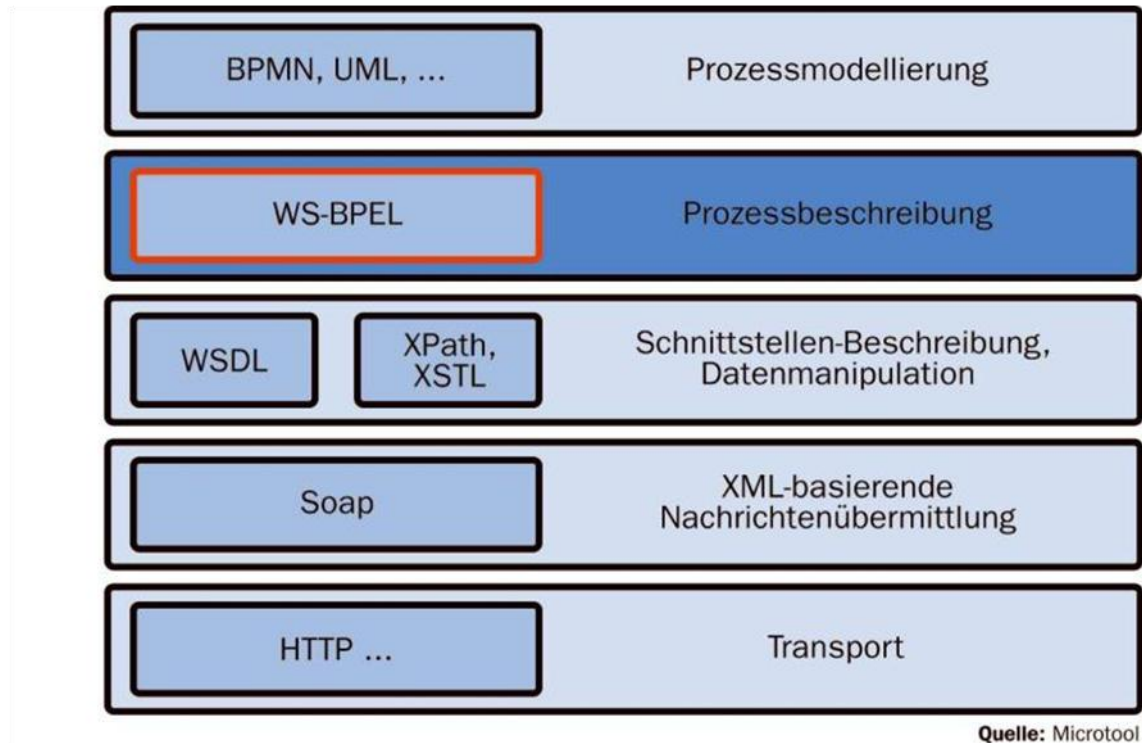
Die korrekte Bezeichnung des Standards, welcher von OASIS gepflegt wird, ist WS-BPEL und steht für Web-Service-BPEL<sup>6</sup>. Die Aktivitäten dieser Sprache werden durch Web-Services implementiert und der Nachrichtenaustausch erfolgt über XML Dokumente (Bartonitz, 2007).

BPEL wurde zunächst von IBM, BEA Systems und Microsoft im Jahr 2002 eingeführt, ehe es im April 2003 an OASIS zur Standardisierung übergang. Bevor WS-BPEL als Standard von OASIS bestätigt wurde, musste sich die Sprache erst gegen andere behaupten. Im gleichen Zeitraum wurde nämlich auch die Business Process Modeling Language (BPML)<sup>7</sup> zur Standardisierung eingereicht. Diese

<sup>6</sup> Wikipedia – WS-Business Process Execution Language: [http://de.wikipedia.org/wiki/WS-Business\\_Process\\_Execution\\_Language](http://de.wikipedia.org/wiki/WS-Business_Process_Execution_Language) (02.03.2010)

<sup>7</sup> Wikipedia – Business Process Modeling Language: [http://de.wikipedia.org/wiki/Business\\_Process\\_Modeling\\_Language](http://de.wikipedia.org/wiki/Business_Process_Modeling_Language) (02.03.2010)

Sprache stammt von BPMI, der Organisation, welche die BPMN Spezifikation erfolgreich standardisiert hatte (Bartonitz, 2007).



**Abbildung 22: WS-BPEL – Ebenen der Internet-Geschäftsprozesse (Nawrot, 2007)**

Die folgenden Unterkapitel beschreiben die Grundstrukturen von WS-BPEL.

### 3.3.3.1 XML-Struktur und Aufbau

Die Grundstruktur eines BPEL Dokuments ist ziemlich kompakt und beginnt mit dem Container `<process>`. Die zwei Attribute in der Abbildung 23 sind verpflichtend, aber zusätzlich existieren noch weitere, die definiert werden können. Das innere des Prozessbereichs wird in den Definitionsteil und die Aktivitäten geteilt.

```
<process name= "testProzess"
  targetName="anyURI">  <!-- Prozess -->

  <extensions />        <!-- Erweiterungen -->

  <imports />           <!-- externe Abhängigkeiten -->

  <partnerLinks />     <!-- Dienste -->

  <variables />        <!-- Daten -->

  <sequence name="main" >  <!-- Ablaufsequenz -->
    <receive />
    <assign />
    <reply />           <!-- Aktivitäten -->
    ...
  </sequence>
</process>
```

**Abbildung 23: WS-BPEL – Prozessdefinition (Stapf, 2005)**

Die wichtigsten Komponenten des Definitionsteils sind Verbindungen zu Partnerprozessen (partners, partnerLinks) und Variablenmengen (variables). Daneben existieren aber noch weitere, die hier für diese Arbeit aber nicht von Bedeutung sind (Stapf, 2005).

### **Abstrakte und ausführbare Prozesse**

Die abstrakten Prozesse werden als Schnittstellenbeschreibung zwischen Business Partnern verwendet. Man kann also sagen, dass abstrakte Prozesse eine Art Projektion der ausführbaren Prozesse sind, wobei die interne Business Logik der einzelnen Partnerunternehmen nicht gezeigt wird (Wagner, 2006).

### **Partner eines Prozesses**

Die Partnerprozesse werden benötigt, um andere Web-Services aufzurufen oder von anderen Web-Services aufgerufen zu werden.

```
<partnerLinks>
  <partnerLink name="Fluggesellschaft"
    partnerLinkType="wsdl:FLugPLT"
    myRole="VerfügbarkeitAbfragen"
    partnerRole="FlugService" />
</partnerLinks>
```

**Abbildung 24: WS-BPEL – PartnerLinks (Bachmann, 2009)**

Dieses Element trägt also zur Gruppierung und Strukturierung der Prozesse bei.

## Variablen

Die Variablen dienen entweder zum Nachrichtenaustausch zwischen Services oder aber sie enthalten Daten für den Prozess. Eine Variable besitzt einen eindeutigen Namen und einen der drei möglichen Typen (Bachmann, 2009):

- **messageType** ... WSDL-Nachrichtentyp,
- **type** ... einfacher oder komplexer XML-Schematype,
- **element** ... XML-Schema-Element.

```
<variables>
  <variable name="FlugAnfrage" messageType="FlAnNa" />
  <variable name="FlugAntwort" messageType="FlAwNa" />
  <variable name="FlugPreis" type="xsd:decimal" />
</variables>
```

**Abbildung 25: WS-BPEL – Variablen (Bachmann, 2009)**

Mittels XPath<sup>8</sup>, das in BPEL standardmäßig verwendet werden kann, können auch Variablen geändert werden.

---

<sup>8</sup> XML Path Language (XPath): <http://www.w3.org/TR/xpath> (20.03.2010)

## Weitere Deklarationen des Definitionsteils

Im Definitionsteil können noch weitere Deklarationen vorgenommen werden, wie (WSBPEL Technical Committee, 2007):

- Korrelationsmengen (*correlationSet*),
- Fehlerbehandlungen (*faultHandler*),
- Kompensationsroutinen (*compensationHandler*),
- Nachrichtenaustausch (*messageExchanges*),
- Ereignisbehandlungen (*eventHandler*),
- Erweiterungen zum Beispiel durch neue Attribute oder Elemente (*extensions*) oder
- Abhängigkeiten zu externen Prozessen, XML-Schemas oder WSDL-Definitionen (*imports*).

Die eigentlichen Aktivitäten werden nach dem Definitionsteil modelliert.

## Aktivitäten

Die Aktivitäten stellen den wichtigsten Teil der Prozessdefinition dar. Die elementaren Aktivitäten (Basic Activities) sind atomar und können nicht weiter verschachtelt werden. Die Tabelle 7 zeigt mögliche Aktivitäten auf (WSBPEL Technical Committee, 2007):

Aktivität	Beschreibung
<code>&lt;invoke&gt;</code>	ruft ein Service einen Partners auf (synchron oder asynchron)
<code>&lt;reply&gt;</code>	antwortet eine eingegangene Nachricht (Antwort auf <code>&lt;receive&gt;</code> )
<code>&lt;receive&gt;</code>	empfängt Nachrichten von anderen Services (nur synchron)
<code>&lt;assign&gt;</code>	kopiert, verändert oder erzeugt Daten
<code>&lt;throw&gt;</code> , <code>&lt;rethrow&gt;</code>	Fehlerbehandlung
<code>&lt;wait&gt;</code>	hältet den Prozess an (für eine bestimmte Zeit oder es

Aktivität	Beschreibung
	wird der genaue Zeitpunkt für das Weitermachen angegeben)
<code>&lt;empty&gt;</code>	Platzhalter für das spätere Einfügen von Aktivitäten
<code>&lt;extensionActivity&gt;</code>	Mittels dieses Elementes können Prozesse durch neuartige Aktivitäten, welche nicht im Standard vorgesehen sind, erweitert werden
<code>&lt;exit&gt;</code>	erzwingt das Beenden einer Prozessinstanz

**Tabelle 7: WS-BPEL – Elementare Aktivitäten (WSBPEL Technical Committee, 2007)**

Strukturierte Aktivitäten enthalten elementare Aktivitäten und beschreiben die Reihenfolge der Ausführung, den Datenfluss sowie das Fehlerhandling. Sie können rekursiv geschachtelt werden.

Aktivität	Beschreibung
<code>&lt;sequence&gt;</code>	definiert die Reihenfolge der Abarbeitung der Aktivitäten
<code>&lt;if&gt;</code>	Bedingte Ausführung von Aktivitäten
<code>&lt;while&gt;</code> , <code>&lt;repeatUntil&gt;</code>	Schleifenkonstrukte für mehrfache Ausführung der Aktivitäten
<code>&lt;pick&gt;</code>	Blockierendes warten auf eine Menge von Nachrichten
<code>&lt;flow&gt;</code>	Parallele Ausführung der Aktivitäten
<code>&lt;forEach&gt;</code>	Mehrfache (sequentielle oder parallele) Ausführung der Aktivitäten

**Tabelle 8: WS-BPEL – Strukturierte Aktivitäten (WSBPEL Technical Committee, 2007)**

Die folgende Abbildung zeigt ein vollständiges Beispiel eines WS-BPEL Prozesses.



```

<process name="Seminarteilnahme">
  <sequence>
    <pick>
      <onAlarm>
        <until>
          '2009-04-15T16:00:00+02:00'
        </until>
        <sequence>
          <!-- Kickoff -->
        </sequence>
      </onAlarm>
    </pick>
    <flow>
      <invoke name="Anmeldung"
        partnerLink="Prüfungsamt"
        operation="anmelden" />
      <repeatUntil>
        <!-- Besprechung des Themas -->
        <!-- Bearbeitung des Themas -->
        <condition>
          current - date () &lt; 2009-06-25
        </condition>
      </repeatUntil>
    </flow>
    <!-- Vortrag halten -->
    <!-- Ausarbeitung schreiben und abgeben -->
    <if>
      <condition>
        <!-- Ausarbeitung nicht ok -->
      </condition>
      <!-- Korrigieren und abgeben -->
    </if>
    <wait>
      <for>'P5DT10H' </for>
    </wait>
    <if>
      <condition>
        <!-- bestanden -->
      </condition>
      <!-- Scheinvergabe -->
      <else>
        <exit ...="" />
      </else>
    </if>
  </sequence>
</process>

```

**Abbildung 26: WS-BPEL – Prozessdefinition (Bachmann, 2009)**

### 3.3.4 Event-driven Process Chain Markup Language (EPML)

EPML („EPC Markup Language“) ist eine weitere auf XML basierte Metasprache zur Definition von Geschäftsprozessen. Wie der Name schon verrät, ist die Sprache ganz auf die „ereignisgesteuerte Prozesskette“ (EPK) abgestimmt.

Jan Mendling von der Wirtschaftsuniversität in Wien und Markus Nüttgens von der Universität des Saarlandes in Saarbrücken sind die Initiatoren und die Hauptmotoren dieser Entwicklung. Im November 2002 stellen sie erstmalig konkrete Anforderungen an die Definition des XML-Schemas für EPKs.

Mendling und Nüttgens konnten in ihrem Artikel (Mendling & Nüttgens, Event-Driven-Process-Chain-Markup-Language (EPML), 2002) bereits Bezug auf bestehende Standardisierungsbemühungen nehmen und entwickelten daraus folgende Anforderungsgruppen:

➤ **Methodisch**

Die Sprachelemente des EPKs sollen vollständig und explizit definiert werden können. Neben den Elementen des flachen EPK-Schemas sind auch Hierarchien und Zustände des Schemas zu modellieren.

➤ **Toolorientiert**

Beinhalten das Ablegen von Positionsdaten sowie Größen der Elemente. Verwaltung von hierarchischen Modellen sowie die Möglichkeit flexible Verweise herzustellen (um z.B. eine Funktion mehrfach verwendet zu können, ohne sie jedes Mal neu zu definieren).

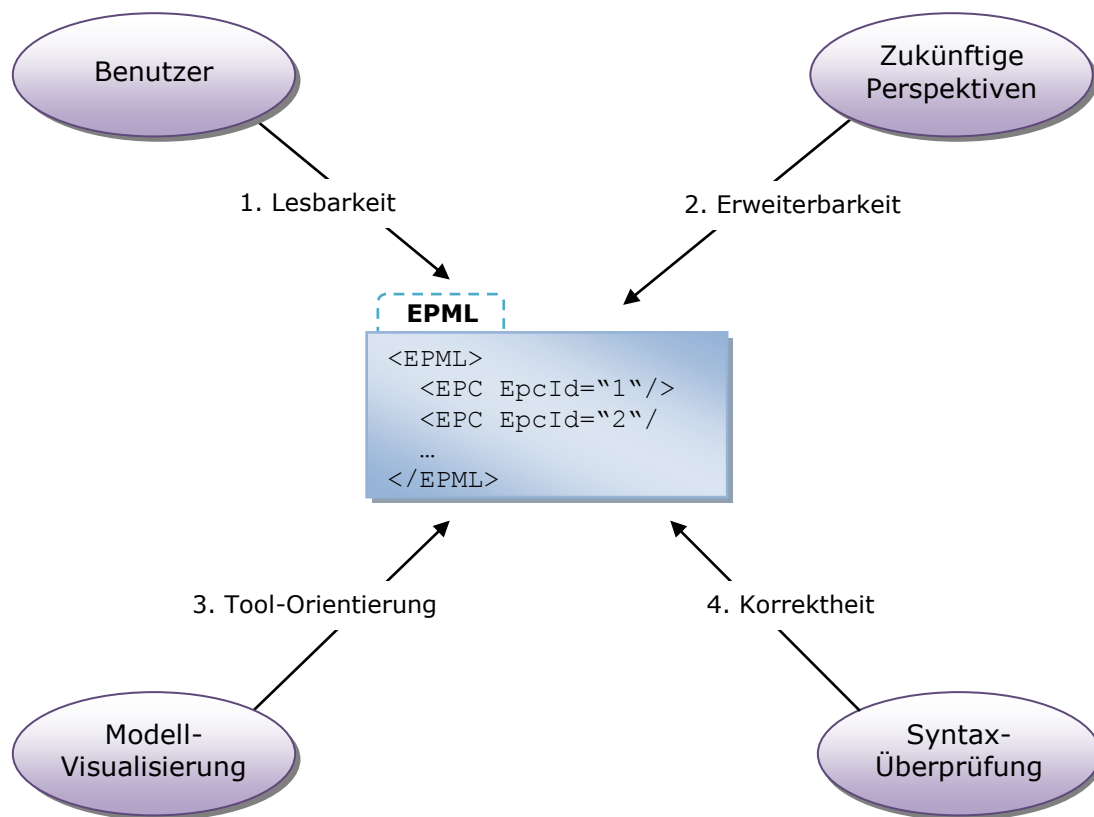
➤ **Implementierungsorientiert**

Hinzufügen von zusätzlichen Implementierungsdetails um beispielsweise Prozess-Zustände einer Prozess-Instanz verwalten und portieren zu können. Dadurch soll auch die Möglichkeit der Konvertierung in ein anderes Workflow-Modell wie BPML geschaffen werden.

➤ **Verifikationsorientiert**

Möglichkeit zur Erweiterung des Modells durch Integritätsbedingungen. Des Weiteren muss aus dem Format eine effiziente Darstellung zur Verifikation des Modells erzeugt werden können.

Die westlichen Ziele waren den Modelldatenaustausch sowie die Modellverifikation zu standardisieren. Aus diesen Anforderungen wurde im 2004 die Syntax von EPML basierend auf den Designprinzipien in Abbildung 27 vorgestellt.



**Abbildung 27: EPML – Design Prinzipien (Mendling & Nüttgens, 2004)**

Die letzte Aktualisierung der EPML Struktur fand im März 2005 statt. Der aktuelle Aufbau wird im Folgenden näher erläutert.

### 3.3.4.1 Struktur und Syntax

Das Hauptelement einer EPML Struktur ist `epml`.

Die Elemente `documentation` und `toolInfo` können Unterknoten eines beliebigen Elementes sein und enthalten anwendungsspezifische Informationen. Diese Daten können beliebigen Typs sein, beziehen sich aber meist auf Details des Speicherns einer Anwendung.

Für die Angaben zur grafischen Positionierung gibt es die Elemente `graphicsDefault` und `graphics`. Das erstere gilt für alle EPK Elemente. Mittels des zweiten Elements können die Standardeinstellungen überschrieben werden. Die entsprechenden grafischen Unterelemente können der Abbildung 28 entnommen werden.

```

epml
  documentation?
  toolInfo?
  graphicsDefault?
    fill? (color, image, gradient-color, gradient-rotation)
    line? (shape, width, color, style)
    font? (family, style, weight, size, decoration, color,
           verticalAlign, horizontalAlign, rotation)
  coordinates (xOrigin, yOrigin)
  definitions?
    definition* (defId)
      name
      description
  attributeTypes?
    attributeType* (typeId)
      description
  directory (name)
    directory* (name) ...
    epc* (epcId, name)
      attribute* (typeRef, value)
      event* (id, defRef) ...
      function* (id, defRef)
        name
        description?
        graphics?
          position? (x, y, width, height)
          fill? (color, image, gradient-color,
                gradient-rotation)
          line? (shape, width, color, style)
          font? (family, style, weight, size,
                decoration, color,
                verticalAlign, horizontalAlign,
                rotation)
          syntaxInfo? (implicitType)
          toProcess? (linkToEpcId)
          attribute* (typeRef, value)
      processInterface* (id, defRef) ...
    and* ...
    or* ...
    xor* ...
    arc* ...
    application* ...
    participant* ...
    dataField* ...
    relation* ...

```

**Abbildung 28: EPML – Elemente (Mendling & Nüttgens, 2005)**

Für die Abbildung von Hierarchien im EPK werden die `directory` Elemente angewendet. Diese können wiederum weitere `directory` Elemente beinhalten und werden mit dem Tag `name` beschrieben. Die Unterelemente `epc` entsprechen den

tatsächlichen Prozessen und werden mit einer eindeutigen Bezeichnung im Tag `epcID` identifiziert.

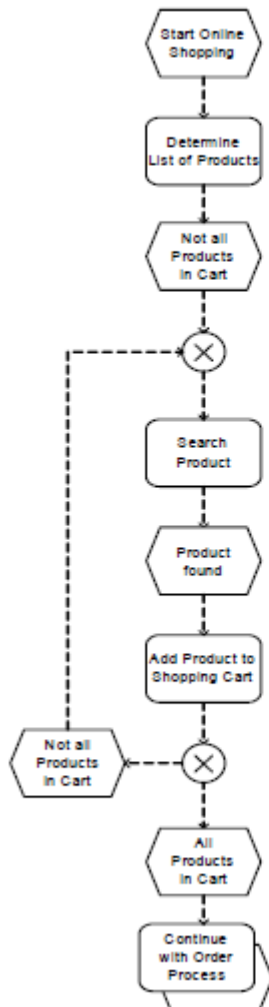
Um Redundanzen zu vermeiden und Definitionen in hierarchischen Modellen mehrfach zu referenzieren, können die Prozesse unter dem Element `definitions` deklariert werden. Informationen welche mit den Standardelementen nicht beschrieben werden können, enthalten die Elemente `attributeTypes` und `attributeType`.

#### *3.3.4.2 Anwendung von EPML auf flache EPKs*

Anhand des folgenden, einfachen Beispiels eines „Online Shopping-Prozesses“ (Mendling & Nüttgens, 2005) wird das Zusammenspiel zwischen EPK und EPML dargelegt.

Das Beispiel beginnt natürlich mit einem Ereignis: „Einkaufen online beginnen“. Danach entscheidet sich der Kunde für die Produkte, welche er kaufen möchte. Es folgt eine Schleife in der Produkte gesucht und anschließend in den Einkaufswagen gelegt werden. Die Schleife wird solange durchlaufen, bis alle Produkte von der Liste im Einkaufswagen sind. Sind alle Produkte gefunden worden, wird mit einem Subprozess fortgesetzt.

### Online Shopping



```

<?xml version = "1.0" encoding = "UTF-8"?>
<epml:epml xmlns:epml = "http://www.epml.de">

  <coordinates
    xOrigin = "leftToRight"
    yOrigin = "topToBottom" />

  <directory name = "Root">
    <epc epcId = "1"
      name = "Online Shopping">
      <event id = "1">
        <name>Start Online Shopping</name>
      </event>
      <arc id = "10">
        <flow source = "1" target = "2"/>
      </arc>
      <function id = "2">
        <name>Determine List of Products</name>
      </function>
      <arc id = "11">
        <flow source = "2" target = "3"/>
      </arc>
      <event id = "3">
        <name>Not all Products in Cart</name>
      </event>
      <arc id = "12">
        <flow source = "3" target = "4"/>
      </arc>
      <xor id = "4" />
      <arc id = "13">
        <flow source = "4" target = "5"/>
      </arc>
      ...
      <processInterface id = "111">
        <name>Continue with Order Process</name>
      ...
    </processInterface>
  </epc>
</directory>
</epml:epml>
  
```

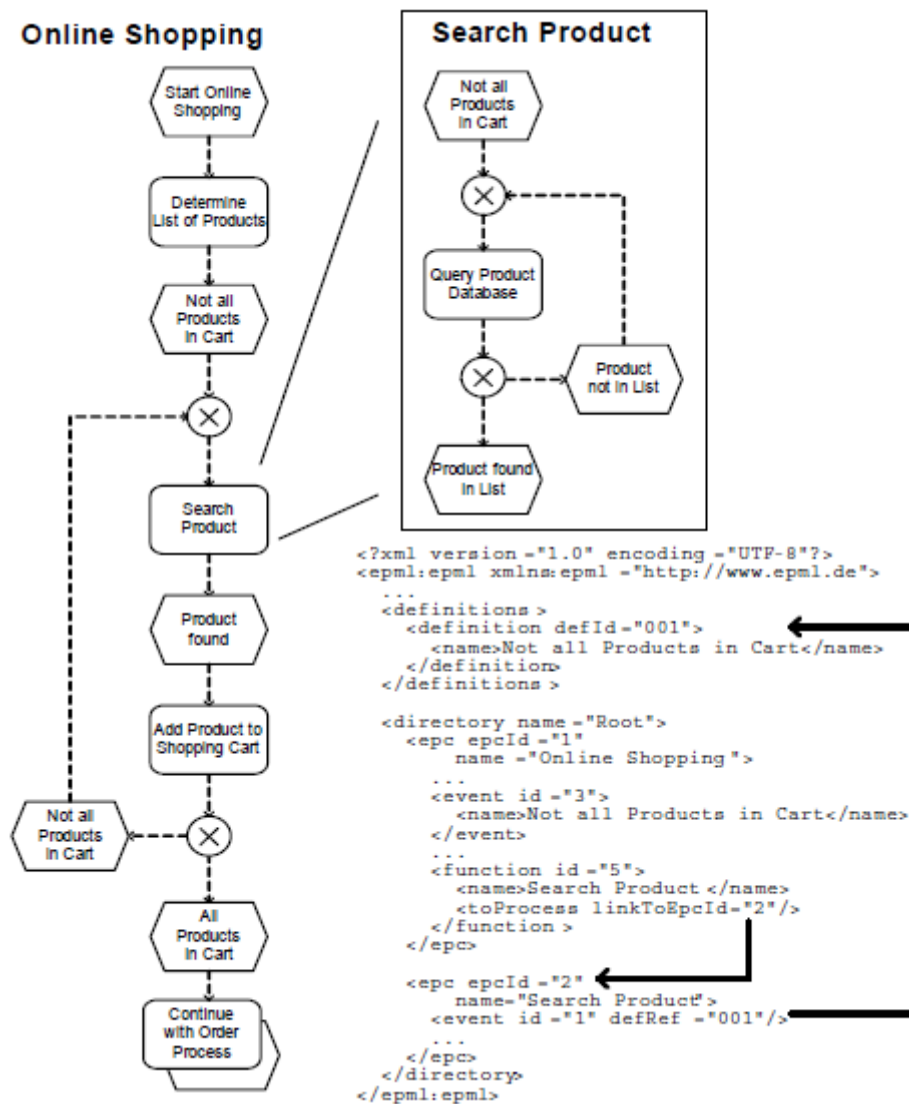
**Abbildung 29: EPML – Flache EPK mit EPML Repräsentation (Mendling & Nüttgens, EPC Markup Language (EPML), 2005)**

Das entsprechende EPML Dokument liest sich fast analog. Es gibt einen EPK Prozess, welcher die `epcID` 1 hat. Die Elemente `arc` wurden bisher nicht beschrieben. Sie stellen die Kanten zwischen den einzelnen EPK Elementen dar und haben einen Start- und einen Endpunkt.

#### 3.3.4.3 Anwendung von EPML auf hierarchische EPKs

Angenommen, das vorhergehende Beispielmodell wird erweitert. Die Funktion „Search Product“ wird zu einem Unterprozess. Im EPML Dokument existieren jetzt zwei Prozesse, die mit einander in Relation stehen und auch gemeinsame Elemente verwenden.

Das mehrfach verwendete Element „Not all Products in Cart“ wird jetzt im definitions Bereich deklariert und in weiterer Folge nur referenziert, um Redundanz zu vermeiden.



**Abbildung 30: EPML – Hierarchische EPK mit EPML Repräsentation**

Der neue Prozess wird im Hauptverzeichnis deklariert und mit Hilfe einer Funktion verlinkt. Das Verlinken geschieht mit dem Unterelement `toProcess` der Funktion, welches im Attribut `linkToEpcID` die eindeutige Identifikation des Unterprozesses angibt.

### 3.4 Zusammenfassung

Obwohl EPC und eEPC viel länger existieren als BPMN und diese Notationen in dem Gesamtkonzept des „ARIS-House of Business Engineering“ zu finden sind, ist BPMN sicherlich der Standard der Zukunft, wenn es um die Modellierung der Geschäftsprozesse geht.

Was die Definitionssprachen angeht, ist es wichtig zu erwähnen, dass diese sich nicht unbedingt ausschließen. Auf Grund der unterschiedlichen Anforderungen während der Entwicklungsphasen decken die Sprachen teilweise unterschiedliche Aufgabenstellungen. Es taucht oft die Frage auf, warum es im Bereich des Geschäftsprozess-Managements zwei Standards – XPDL und BPEL – für den Austausch braucht.

Dazu kann folgendes gesagt werden: XPDL gilt hier als reines Austauschformat, welches sich auf die Transformation zwischen unterschiedlichen Tools konzentriert. Dafür hat XPDL die Möglichkeit geschaffen, toolspezifische Informationen, wie die grafische Beschreibung der Elemente, abzuspeichern. BPEL hingegen ist für diesen Zweck nicht geeignet, da es keine Informationen zur grafischen Darstellung halten kann und außerdem viele Aspekte der modernen Geschäftsprozesse fehlen. BPEL ist eine Sprache zur Orchestrierung von Web-Services und ist demnach auch hauptsächlich bei Internetlösungen anwendbar (Workflow Management Coalition, 2010).

In den diversen Internetforen und –Communities existiert sowohl die Meinung, dass sich die beiden Formate ausschließen, als auch, dass sie komplementär wären.



## 4 Workflow Management Systeme in der Praxis

---

Nach dem Einblick in die aktuellen Technologien der Workflow Management Systeme verschafft der erste Teil dieses Kapitels etwas Übersicht über den Einsatz in der Praxis. Der zweite Teil des Kapitels zeigt die Entscheidungen und die zugrunde liegenden Kriterien, welche im Zusammenhang mit Wasaka getroffen wurden.

### 4.1 Implementierungen und Projekte

Ein wichtiger Indikator für die Durchsetzbarkeit eines Standards ist sicherlich die Anzahl der bestehenden sowie der geplanten Softwareprojekte, die den angesprochenen Standard verwenden bzw. implementieren. Bedeutend ist außerdem, welches Unternehmen hinter diesen Konzepten steht, damit nicht nur in der näheren Zukunft mit Unterstützung gerechnet werden kann.

#### 4.1.1 EPC und eEPC Projekte

Wie schon erwähnt, gehörten diese Modellierungssprachen zu einem kompletten Konzept<sup>9</sup>. Obwohl sie wahrscheinlich das älteste Modell für die Darstellung der Geschäftsprozesse sind, haben es die Notationen nicht zu einem allgemein anerkannten Standard geschafft. Das ist wahrscheinlich auch der Grund, dass sich die Implementierungen im Rahmen halten:

- **EPC Tools**<sup>10</sup> wurde entwickelt von Nicolas Cuntz und Ekkart Kindler an der Universität Paderborn. Die Software dient der Erstellung, Überprüfung und Simulation der EPC-Modelle.
- **ARIS Toolset** ist ein Produkt der Firma IDS Scheer AG. Es dient als Werkzeug zum Erstellen, Pflegen und Optimieren der Geschäftsprozesse

---

<sup>9</sup> ARIS-House of Business Engineering: <http://de.wikipedia.org/wiki/ARIS> (20.01.2010)

<sup>10</sup> EPC-Tools: <http://www2.cs.uni-paderborn.de/cs/kindler/Forschung/EPCTools> (20.01.2010)

welche auf dem ARIS-Konzept basieren. Die Software wird häufig bei den Implementierungen von SAP-Systemen genutzt.

Alle ARIS-bezogenen Produkte wurden mittlerweile in der ARIS Plattform<sup>11</sup> zusammengeführt, die auch andere als die hier erwähnten Standards wie BPMN und BPEL unterstützt.

## 4.1.2 BPMN Projekte

Der Homepage von BPMN nach gibt es derzeit 61 Produkte, welche die Notation implementiert haben und weitere vier sind in Planung (OMG). Viele wichtige Unternehmen, wie Fujitsu, IBM oder SAP, haben die BPM Notation in ihre Gesamtlösungen eingebaut.

Die BPM Notation ist allerdings sehr umfangreich. Es gibt Studien, welche belegen, dass weniger als zwanzig Prozent des Vokabulars der Sprache regelmäßig verwendet wird und einige der komplexeren Sprachkonstrukte gar nicht zur Anwendung kommen (zur Muehlen & Recker, 2008).

### 4.1.2.1 BPMN und .NET

Die Integration von BPMN ins Visual Studio oder die Verbindung mit .NET fällt recht mager aus. Im Hause Microsoft ist hier nichts zu finden. Es gibt lediglich einige Third-Party Unternehmen<sup>12</sup> welche Erweiterungen für Microsofts Visio zur Verfügung stellen.

Als Erweiterung zum Microsofts SharePoint Server gibt es noch die BPM Lösung aus dem Hause Skelta<sup>13</sup>.

---

<sup>11</sup> ARIS-Plattform: [http://www.ids-scheer.de/de/ARIS\\_Software\\_Software/7796.html](http://www.ids-scheer.de/de/ARIS_Software_Software/7796.html) (14.02.2010)

<sup>12</sup> Interfacing Technologies - Free BPMN Modeler: <http://interfacing.com/Products/business-process-modeling> (20.01.2010), bzw. ITpearls - Process Modeler for Visio: <http://www.itpearls.com/index.php?id=28&L=2> (20.01.2010)

<sup>13</sup> Skelta SharePoint Accelerator: <http://www.skelta.com/products/sps/sharepoint-workflow.aspx> (20.01.2010)

### 4.1.3 EPML Projekte

Die Sprache ist noch relativ jung im Vergleich mit anderen hier erwähnten. Genau wie die entsprechenden Modelle (EPC und eEPC) ist sie nicht weiter verbreitet. Folgende Liste ist ein Auszug der EPK-Community Homepage:

➤ **C-EPC-Validator**

Dieses Tool ist ein XSLT Validierungsprogramm, dass die Korrektheit der EPML Dateien in der Version 2.1 überprüft.

➤ **Semtalk**

Das Programm ist ein professionelles BPM Tool der Firma Semtation GmbH, welches als Aufsatz für das Microsoft Visio dient.

➤ **AML2EPML und EPML2AML**

Das sind Transformationsskripts zwischen der AML Sprache des ARIS Toolsets und EPML entwickelt von Jan Mendling.

### 4.1.4 BPEL Projekte

Die Prozesse, welche mit BPEL modelliert wurden, können mit sogenannten BPEL-Engines ausgeführt werden. Die Engine kann über verschiedene Schnittstellen mit Maschinen (Web-Services) und mit Menschen kommunizieren, um die Prozesse richtig durchzuführen, zu steuern und zu kontrollieren (Freud & Götzer, 2008).

Zu beachten ist, dass BPEL selbst keine Unterstützung für die Interaktion mit Menschen bereithält. Diese Funktionalität kann nur mit Erweiterungen, wie beispielsweise BPEL4People, hinzugefügt werden. Das stellt einen der größten Stolpersteine dar, wenn es um die Herstellerunabhängigkeit der Prozessdefinition geht, da jeder Hersteller dieses Problem anders behandelt.

Die erste offizielle BPEL-Engine auf dem Markt war Collaxa von Oracle. Die Engine basierte auf Win NT und beinhaltete Debugging und Visualisierung von Prozessen sowie ein Web-Interface (Wagner, 2006). Zu den derzeit wichtigsten kommerziellen Engines zählen IBM's Web-Sphere, BizTalk von Microsoft und BPEL Process Manager aus dem Hause Oracle. All diese Engines bieten auch eine grafische Unterstützung beim Erstellen der Prozesse an (Bachmann, 2009). Unter den Open Source Projekten ist Open ESB von Sun Microsystems ganz oben anzusetzen. Für einen

ersten Überblick, hält Wikipedia<sup>14</sup> eine Übersicht der bedeutendsten Lösungen bereit.

#### 4.1.4.1 BPEL und .NET

Die Windows Workflow Foundation umfasst auch eine vollständige Abbildung von BPEL. Der Designer ist im Visual Studio 2008 inkludiert. Die Entwickler werden mit dem „BPEL for Windows Workflow Foundation March CTP“ unterstützt. Es ist ein Add-On für die Workflow Foundation, welches das Importieren und Exportieren von BPEL Workflows ermöglicht.

Das Postfix CTP steht hier für „Community Technology Preview“.

#### 4.1.5 XPDL Projekte

Generell können Anwendungen, die XPDL unterstützen, in drei Kategorien unterteilt werden. Die erste Kategorie unterstützt allein die Dokumentation von Geschäftsprozessen. Die zweite Art der Anwendungen inkludiert eine Workflow Engine für die Ausführung der definierten Prozesse. Die dritte Kategorie simuliert und analysiert Geschäftsprozesse (Bartonitz, XPDL - XML Process Definition Language, 2007):

##### **Workflow-Anwendung:**

###### ➤ **COSA BPM**

Die Prozessdiagramme werden BPMN-nah grafisch erstellt, in XPDL 2.0 gespeichert und beim Übertragen in die Workflow-Engine in Petri-Netze<sup>15</sup> konvertiert. Der Simulator lädt die XPDL und die aktuellen Prozessdaten zur Analyse.

###### ➤ **ECM Documentum Process Suite**

---

<sup>14</sup> Wikipedia - Comparison of BPEL Engines:  
[http://en.wikipedia.org/wiki/Comparison\\_of\\_BPEL\\_engines](http://en.wikipedia.org/wiki/Comparison_of_BPEL_engines) (27. 02 2010)

<sup>15</sup> Definition: „Petri-Netz ist ein formales Modell zur Beschreibung von Systemen mit dem Schwerpunkt der Modellierung von asynchronen, nebenläufigen Vorgängen. Petri-Netze werden sowohl zur Modellierung von Systemen im Hardwareentwurf als auch im Softwareentwurf eingesetzt.“ (Teich & Haubelt, 2007)

Die Anwendung ermöglicht Modellierung von BPMN-Diagrammen mit Export nach XPDL und BPEL. XPDL kann von dem eigenen Analyzer oder auch von der Engine verarbeitet werden.

➤ **IBM FileNet Business Process Manager**

Der Manager bietet vollen Support von XPDL Version 1.0 und 2.0 als auch BPMN.

**Dokumentations- und Simulationsanwendungen:**

➤ **BOC Adonis**

Die Software unterstützt Modellierung mit BPMN, Speicherung in XPDL und Analyse von Geschäftsprozessen.

➤ **ITP Commerce Process Modeler for Microsoft Visio**

Die Erweiterung für Visio bietet die Darstellung in BPMN, Export nach XPDL oder BPEL, Dokumentation, sowie direkte Unterstützung von Workflow Engines wie Microsoft BizzTalk-Server, Carnot oder Oracle Process Manager.

WfMOpen<sup>16</sup> und Enhydra Shark<sup>17</sup> sind zwei Open Source Softwareprodukte. Beide unterstützen die OMG und WfMC Standards, basieren auf der J2EE Technologie und sind weit verbreitet. Diese Liste der Workflow-Anwendungen ist bei weitem nicht vollständig (Pierre, 2007).

#### 4.1.5.1 XPDL und .NET

Der WfMC zeigt auf seiner Homepage eine Reihe von Produkten, welche XPDL bereits implementieren oder unterstützen. Im Zusammenhang mit der .NET Technologie ist die Auswahl aber sehr begrenzt (Workflow Management Coalition):

- **Ascentn AgilePoint Server** ist ein auf .NET basierter BPMS, welches XPDL unterstützt.
- **Aspose's Aspose.Workflow** ist eine .NET basierte Workflow Engine, welche XPDL verwendet. Allerdings wurde die Unterstützung für dieses Produkt seit 19.01.2010 abgebrochen.

---

<sup>16</sup> WfMOpen: <http://wfmopen.sourceforge.net> (25.03.2010)

<sup>17</sup> Enhydra Shark: <http://www.enhydra.org/workflow/shark/index.html> (25.03.2010)

Somit war die Suche nach der .NET Unterstützung für XPDL auch schon wieder vorbei.

#### **4.1.6 Übersetzungs- und Transformationstools**

Die Entwicklung des Business-Process-Managements dauert seit Beginn der 90'er Jahre, dennoch konnten sich nicht immer klare und eindeutige Standards durchsetzen. Gerade bei XPDL und BPEL erleben wir in vieler Hinsicht eine Überschneidung der Aufgaben, wobei jede Sprache noch Stärken mit sich bringt, die die andere nicht hat.

Um die Schwächen der einen Sprache mit einer anderen zu beheben, entstanden verschiedene Übersetzungs- bzw. Transformationstools.

### **4.2 Fazit der Recherche**

Nach der Einsicht in die zur Verfügung stehenden Technologien, gilt es abzuwägen welche und unter welchen Bedingungen für unser Projekt Wasaka angewendet werden könnte. Dazu werden die wichtigsten Kriterien in diesem Zusammenhang noch einmal aufgelistet:

#### **Aus der Sicht des Softwareherstellers, der Firma ASE:**

- Das WfM System solle einfach mit Diagnosekomponenten (.NET) erweitert werden können.
- Eine einfache Wartung und Erweiterung des Systems ist ebenfalls eine Voraussetzung.
- Die Kosten der zusätzlichen Komponenten und/oder der neuen Hardware dürfen zehn Prozent der Projektkosten nicht übersteigen.

#### **Aus der Anwendersicht, des Unternehmens BMW:**

- Das Schreiben der Workflows muss auch für Nicht-Informatiker einfach und verständlich sein.
- Eine leichte Durchführung der Änderungen und Erweiterungen sowie klardefinierte Fehlerbehandlung soll unterstützt werden.
- Die Lizenzkosten dürfen zehn Prozent der gesamten Kosten des Projektes nicht übersteigen.

### 4.2.1 Komponenten einer möglichen Lösung

Es stellt sich die Fragen, wie eine mögliche Lösung aussehen könnte und welche Komponenten dafür notwendig sind.

1. Es braucht eine Komponente, um Diagnoseabläufe mittels grafischer Unterstützung festzulegen bzw. kombinieren zu können. Dazu müssen Eingangs- und Ausgangsparameter, sowie Abhängigkeiten zwischen Diagnosejobs bestimmt werden können.
2. Für spezielle Abläufe (zum Beispiel: das Implementieren von mathematischen Formeln oder das Bearbeiten von Zeichenketten) braucht es eine Möglichkeit diese mittels Schreiben von Quellcode oder mittels Funktionen zu implementieren.
3. Desweiteren wird eine Komponente zur Speicherung, Verwaltung und Verteilung der einzelnen Definitionen verwendet.
4. Abschließend ist eine Ausführungskomponente, welche die Kommunikation mit dem Anwender und dem Steuergerät übernimmt und die Diagnosejobs ausführt, notwendig.

Wie aus den einzelnen Produktbeschreibungen hervorgeht, erfüllen viele der im Unterkapitel 4.1 angesprochenen Gesamtsysteme die Punkte 1 und 3. Die Unterstützung für die Anforderungen 2 und 4 lassen sich aber sehr schwer nachweisen bzw. dementieren, da die Systeme detaillierter geprüft werden müssten, und dass aus Zeit- und Kostengründen nicht möglich ist.

### 4.2.2 Gegenüberstellung der Lösungen

Die folgende Tabelle vergleicht die Lösungen anhand für das Projekt Wasaka relevanter Anforderungen.

	<i>Oracle BPEL Process Manager</i>	<i>IBM WebSphere</i>	<i>Microsoft BizTalk</i>	<i>Sun Open ESB</i>
	<b>Hersteller (ASE)</b>			
.NET Anbindung	<b>X</b>	<b>X</b>	<b>OK</b>	<b>X</b>
Einfache Wartung und Erweiterung	<b>?</b>	<b>?</b>	<b>?</b>	<b>?</b>
Kosten für Komponente	<b>X</b>	<b>X</b>	<b>X</b>	<b>OK</b>

	<i>Oracle BPEL Process Manager</i>	<i>IBM WebSphere</i>	<i>Microsoft BizTalk</i>	<i>Sun Open ESB</i>
Kosten für Support	X	X	X	X
	<b>Anwender (BMW)</b>			
Erstellen neuer Diagnosevorgänge	?	?	?	?
Änderungen und Erweiterungen	?	?	?	?
Fehlerbehandlung	?	?	?	?
Lizenzkosten	X	X	X	OK

**Tabelle 9: ASE Kriterien vs. Workflow Management Systeme**

Die tabellarische Übersicht zeigt deutlich, dass für das Problem der Diagnose, diese Art der Lösung nicht wirklich zusagt. Die Kosten für die Gesamtsysteme der bekannten Unternehmen wie Oracle, Microsoft oder IBM belaufen sich zwischen 8.499,- \$ (Microsoft BizTalk) und 43.050,- € (Oracle BPEL Process Manager), was ein eindeutiges K.O. Kriterium darstellt. Ferner ist es schwierig die Mächtigkeit solcher Systeme nur mit Hilfe der im Web vorhandenen Dokumente zu bestimmen.

Ein weiteres Problem stellt die .NET Anbindung dar. Diese kann lediglich beim BizTalk von Microsoft angenommen werden, da alle anderen Systeme auf Java basieren. Die Interoperabilität dieser beiden Plattformen<sup>18</sup> ist zwar immer besser, durch das fehlende Know-How auf der Seite der Firma ASE würde aber weitere Verzögerungen und damit Kosten entstehen, welche die angebotene Lösung verteuern würden.

Bei den angebotenen Open-Source Lösungen, wie bei Open ESB von Sun Microsystems entstehen zwar keine Anschaffungskosten, dennoch ist die Komplexität der einzelnen Systeme für das Problem der Diagnose zu groß. Die Notationen und Sprachen sind sehr umfangreich und einem Prüfstandfahrer bei BMW nicht zumutbar.

Für das neue Projekt wurden daher andere, hausinterne Lösungen, die im folgenden Kapitel vorgestellt werden, gewählt.

---

<sup>18</sup> Microsoft J+N-Initiative: <http://www.microsoft.com/windowsserversystem/jplusn/default.aspx> (17.01.2010)



## 5 Wasaka – Die neue Diagnosesoftware

---

Unter dem Entwicklungsnamen Wasaka wurde bzw. wird immer noch die neue Applikation für Diagnoseaufgaben programmiert. Es handelt sich um ein modular aufgebautes, auf .NET basiertes System.

Neben Standardfunktionen wie den Fehlerspeicher-Explorer gibt es auch etliche Steuergerät-Spezifische Module. Der Funktionsbaum auf der linken Seite der Applikation wird nach dem Initialisieren mit dem Steuergerät aufgebaut und bietet verschiedenste Diagnosefunktionen, welche vom Steuergerät unterstützt werden.

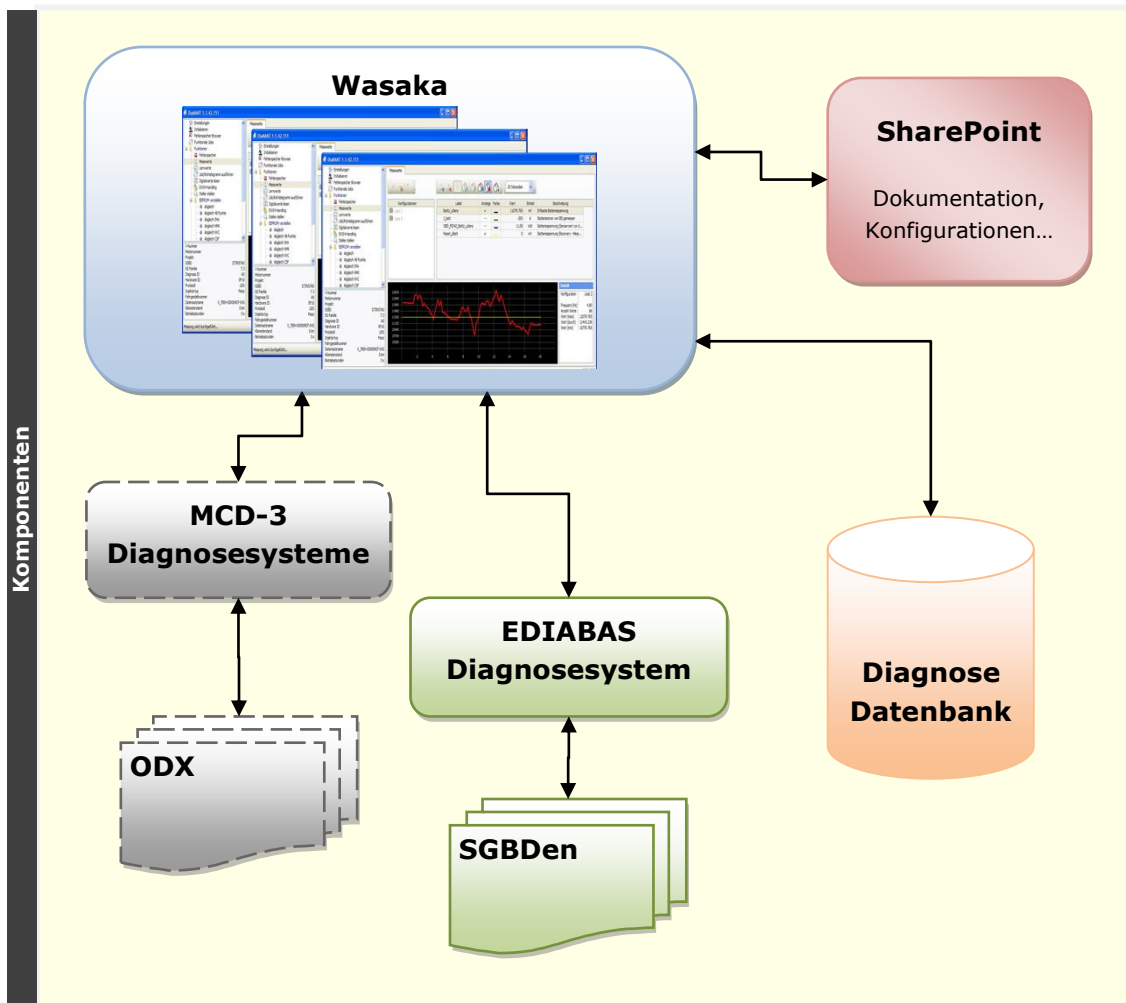
Wasaka ist mit einer Diagnose-Datenbank verbunden, die für die Verwaltung von:

- Fehlerspeicherinhalten,
- Einmal-Messwerten und
- verschiedenen Lernwerten (NMK Schnell-Lernen, etc.) zuständig ist.

Mit Wasaka können zu jeder Zeit Fehlerspeicherinhalte ausgelesen oder in die Datenbank hochgeladen werden. Des Weiteren existiert eine Vielzahl der Darstellungsmöglichkeiten der Diagnosedaten wie HTML oder XML.

### 5.1 Komponenten der Diagnose

Wasaka ist sowohl für die Arbeit mit dem Diagnosesystem EDIABAS als auch mit Diagnosesystemen, welche auf MCD-3 basieren, ausgelegt. Für die Arbeit im Fahrzeug bzw. bei fehlendem Netzzugang gibt es auch eine lokale Datenbank, in welche die Daten abgelegt und gesucht werden können. Ist der Netzzugang wieder möglich, können die Daten in die gemeinsame Diagnose-Datenbank hochgeladen werden.



**Abbildung 31: Wasaka – Zusammenspiel der Diagnosekomponenten**

Die Abbildung 31 zeigt die Komponenten der neuen Lösung. Dazu folgt eine Kurzbeschreibung der einzelnen Teile:

- **Wasaka** ist die neue Applikation für die Visualisierung und Verwaltung der Diagnosedaten. Das Tool bildet Diagnoseabläufe ab und bietet einen transparenten Zugriff auf wichtigste Diagnosefunktionen.
- **MCD-3** stellt Diagnosesysteme nach dem ASAM MCD-3 Standard dar (siehe Unterkapitel 2.2.2).

- **ODX<sup>19</sup>** ist ein weiterer ASAM Standard und die international genormte Beschreibungssprache für die Fahrzeug- und Steuergeräteinformationen. Die Definition des Datenaustauschformates für Diagnosebeschreibung, sowie die Definition der Semantik und Syntax von Diagnosebeschreibungen, zählen zu den dessen Hauptaufgaben (Wallentowitz & Reif, 2006).
- **EDIABAS** steht für die Diagnosesoftware Elektronik Diagnose Basissystem der Firma Softing AG.
  - **SGBD** ist ein Kürzel für Steuergerätebeschreibungsdatei. Diese Dateien werden vom Unternehmen geschrieben und gewartet.
- Die **Diagnosedatenbank** ist eine gemeinsam genutzte Oracle-Datenbank zur Speicherung und Verwaltung der Diagnosedaten.
- Der **Microsoft SharePoint Server** hält und verwaltet einen Teil der Konfigurationsdateien der Applikation. Außerdem unterstützt er die Fehler-, Dokumentations- und Lizenzierungsverwaltung.

## 5.2 Die Funktionalität

Startet die Applikation, verbindet sie sich automatisch mit der Diagnosesoftware. Ist der Mitarbeiter mit dem Fahrzeug oder HIL<sup>20</sup> verbunden, kann die Software das angebundene Motorsteuergerät identifizieren und entsprechende Diagnosemodule anbieten.

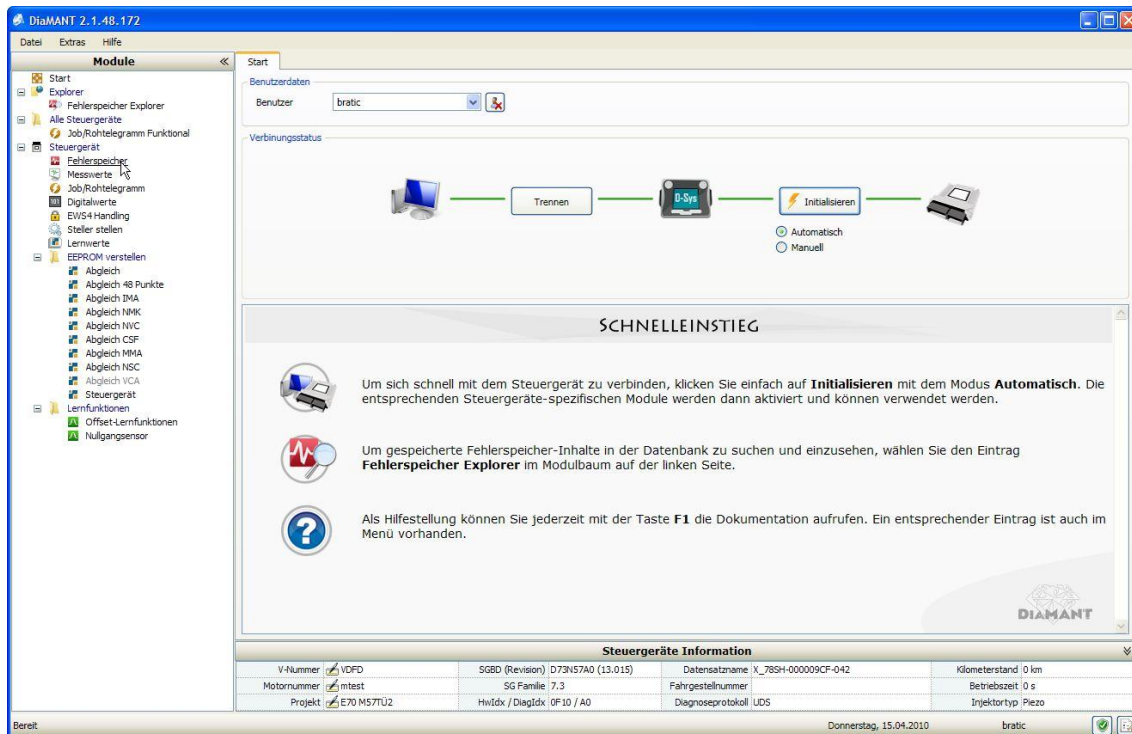
Die Abbildung 32 zeigt die Ansicht nach der erfolgreicher Identifizierung des Steuergerätes bzw. Ermitteln der Steuergerätebeschreibungsdatei. Der Baum auf der linken Seite der Applikation listet dabei die Möglichkeiten der Benutzerlizenz entsprechend auf. Wird ein Eintrag ausgewählt, zeigt die rechte Seite den Dialog des Moduls an.

---

<sup>19</sup> ODX (ASAM MCD-2 D):

[http://www.asam.net/index.php?option=com\\_content&task=view&id=124&Itemid=206](http://www.asam.net/index.php?option=com_content&task=view&id=124&Itemid=206)  
(24.03.2010)

<sup>20</sup> HIL (Hardware in the Loop): Nachbildung des realen Fahrzeugs zu Simulationszwecken

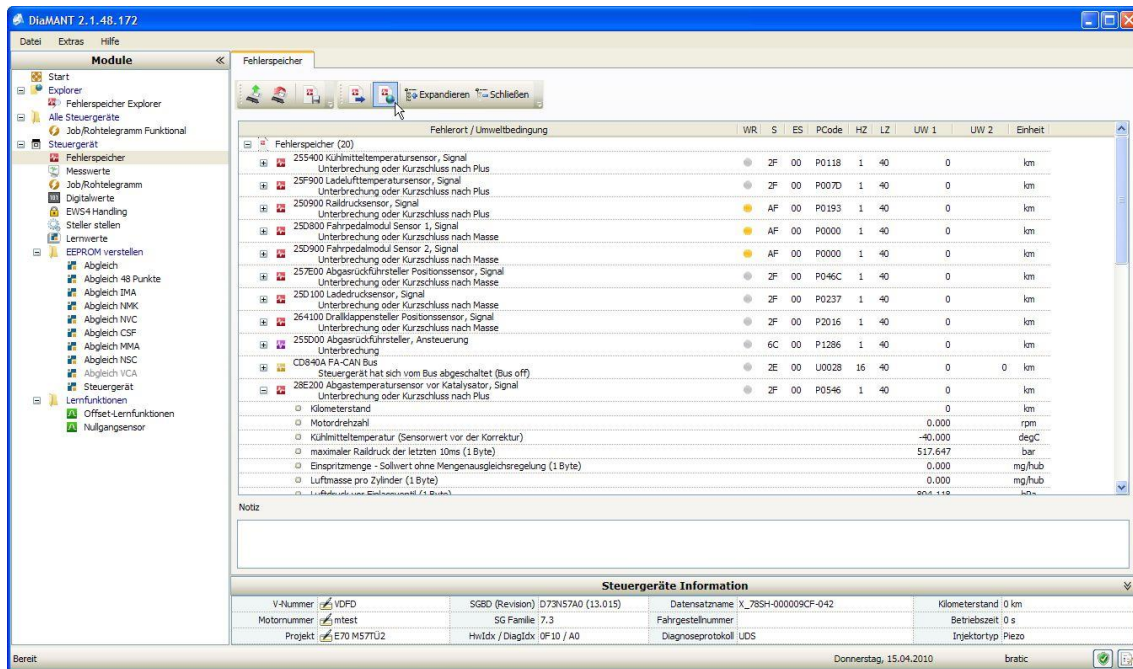


**Abbildung 32: Wasaka – Nach erfolgreicher Initialisierung des Steuergerätes**

Neben etablierten und im Unternehmen BMW standardisierten Diagnosefunktionen, welche nahezu von jedem Steuergerät unterstützt werden, gibt es auch eine Vielzahl von Diagnosejobs, die abhängig von Diagnoseprotokoll, Zylinderanzahl oder der Steuergerätefamilie angewendet werden kann.

### 5.2.1 Das Lesen des Fehlerspeichers

Das Auslesen des Fehlerspeichers ist einerseits eine Standarddiagnoseaufgabe und andererseits ein sehr spezieller Diagnosejob. Dadurch, dass die Jobs für das Auslesen in allen Steuergeräten absolut identisch sind – lediglich die Ergebnisse können leicht variieren, ist die Automatisierung dieser Aufgabe eher einfach. Das Erzeugen der gewünschten Darstellung nimmt da schon etwas mehr Zeit in Anspruch.



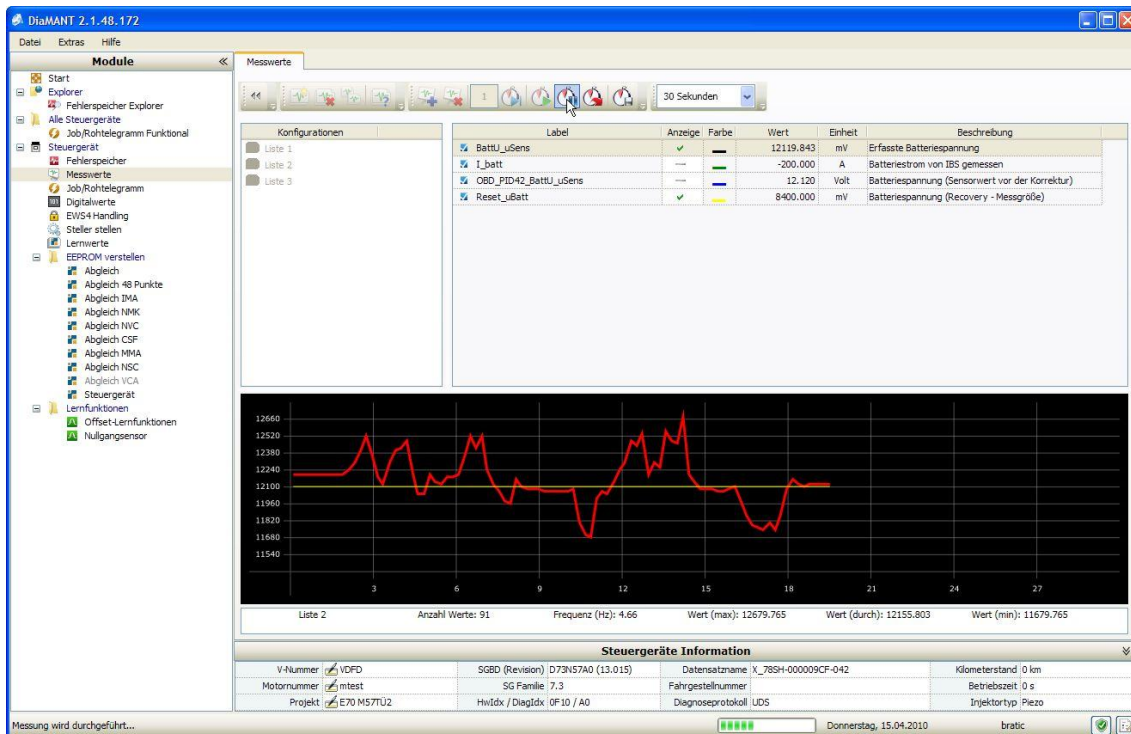
**Abbildung 33: Wasaka – Fehlerspeicher Modul**

Der Fehlerspeicherinhalt wird samt Umweltbedingungen und Statusinformationen übersichtlich dargestellt. In diesem Modul kann der Fehlerspeicher

- ausgelesen,
- gelöscht,
- kommentiert,
- oder in verschiedene andere Ansichten (XML, HTML...) transformiert werden.

### 5.2.2 Die Durchführen von Messungen

Das Messmodul ist ein gutes Beispiel für eine einfache, aber vielseitig einsetzbare Diagnosefunktion. Um Werte aus dem Steuergerät zu messen, wird der entsprechende Job mit einer Reihe von Labels aufgerufen. Das Steuergerät sendet dann zu jedem Label den Wert, die in der Tabelle der Steuergerätebeschreibungsdatei festgelegte Einheit und eine Info bzw. Beschreibung dazu.



**Abbildung 34: Wasaka - Modul zum Lesen der Messwerte**

Das einzig spezielle an diesem Job ist dass er nicht wie die meisten anderen Diagnosefunktionen eine vorgegebene fixe Anzahl an Ergebnissen hat, sondern dass diese Anzahl von den Aufrufparametern abhängt. Für diesen recht simplen Job entstand aber ein sehr komplexes und vielseitig einsetzbares Modul.

### 5.2.3 Weitere Diagnosemodule

Neben den oben dargestellten Modulen existieren noch viele weitere, wie: Abgleichfunktionen, Lernwerte, Lernfunktionen, Abfrage der Digitalwerte, Verstellen der Steller und viele andere mehr. Die folgenden Abbildungen zeigen die grafische Oberfläche zweier weiterer Module.

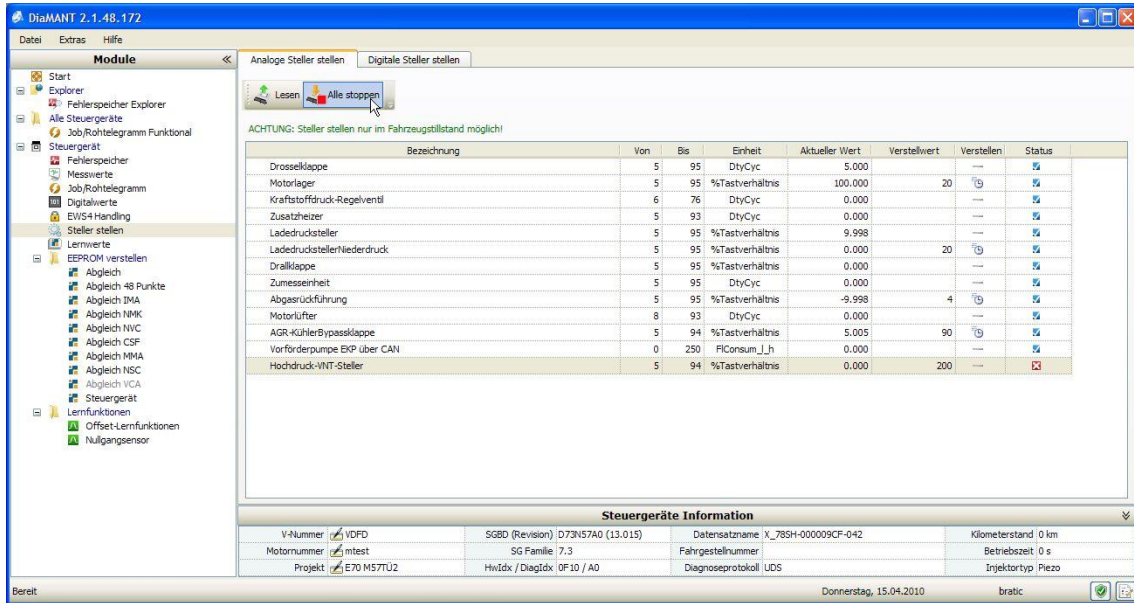


Abbildung 35: Wasaka - Steller stellen

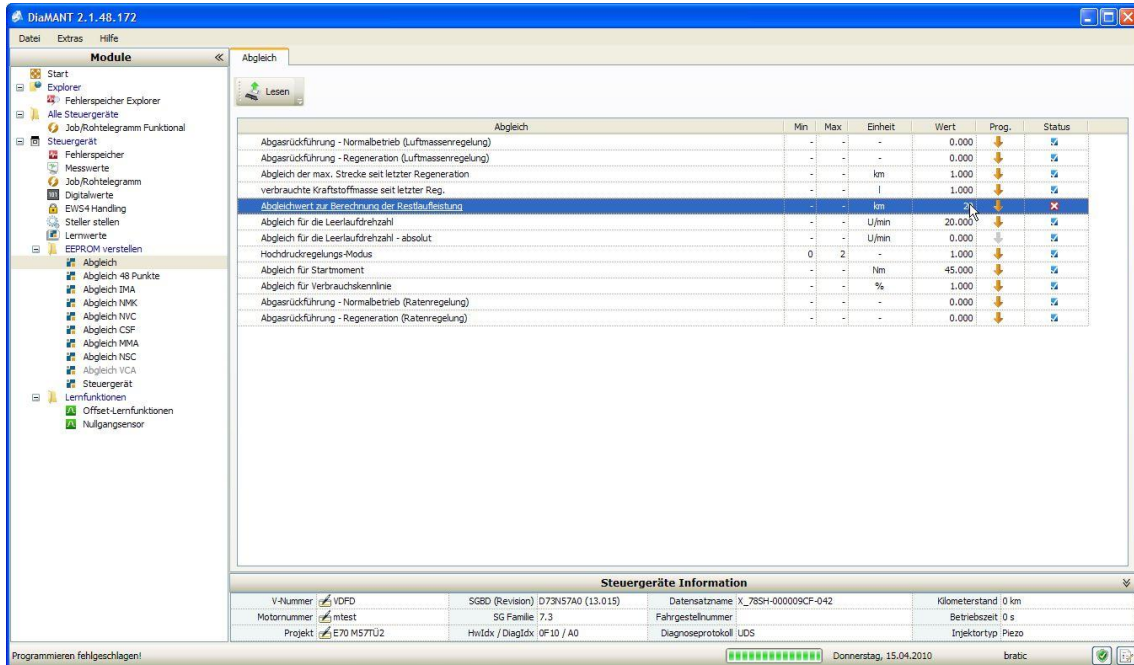
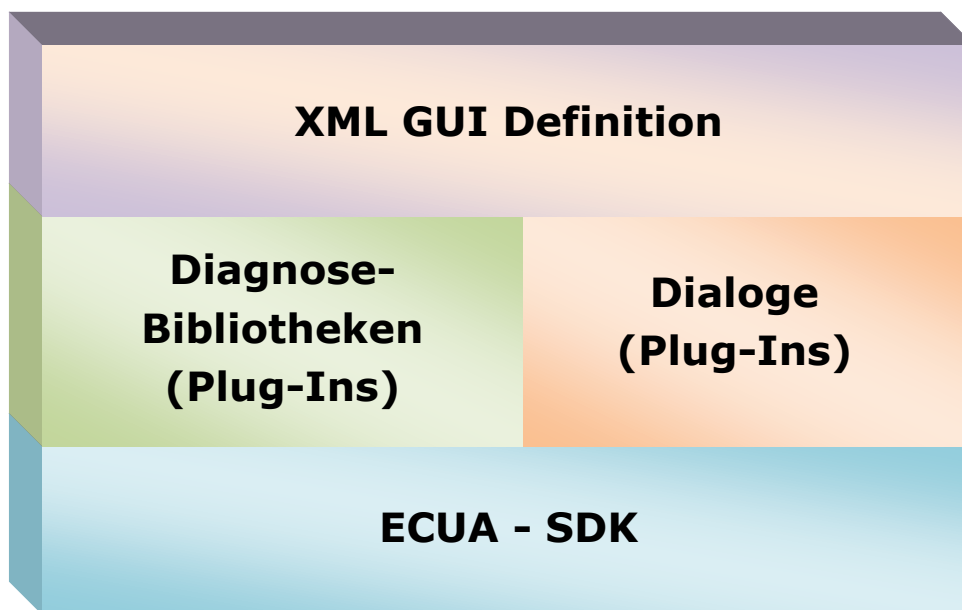


Abbildung 36: Wasaka - Modul EEPROM Abgleich

Obwohl viele dieser Module gleiche Steuergerätefunktionen verwenden, sind die Abläufe sehr auf die jeweilige Aufgabe spezialisiert und können kaum verallgemeinert werden.

### 5.3 Architektur und Aufbau

Wasaka ist in zwei essentielle Schichten zu teilen. Das SDK ist ein hauseigenes Framework basierend auf der .NET Technologie. Sie ist der Grund, weshalb Neuentwicklungen oder zugekaufte Softwareprodukte die .NET Technologie unterstützten müssen. Der zweite Teil sind die Wasaka Plug-Ins und die dazugehörige XML Definition der GUI, welche die eigentlichen Dialog- und Diagnosefunktionalitäten beschreiben.



**Abbildung 37: Wasaka – Architektur der Lösung**

Das SDK ermöglicht eine sehr einfache und schnelle Entwicklung von Dialogen. Die einzelnen Dialoge werden mittels XML definiert und mit einem Plug-In verbunden. Das Prinzip ist dem XAML aus der Microsoft Windows Presentation Foundation sehr ähnlich.

Warum ASE aber ein eigenes Framework verwendet, ist sowohl historisch als auch praktisch bedingt. Ein SDK auf Basis von Visual Basic existierte bereits und im Laufe des Winters 2008/2009 wurde dieses SDK für die .NET Plattform neu



konzipiert und implementiert. Die XML Definitionen haben sich dabei nicht verändert, um die Abwärtskompatibilität zu wahren.

Das ASE SDK übernimmt viele Aufgaben, wie die Visualisierung, automatische Größenanpassung, aber auch Objektverwaltung. Dabei bauen wir auch eigene GUI-Objekte und verwenden einige Nicht-WPF Komponenten.

## 5.4 Die Entwicklungsphase der Abläufe

Die Informationen und den einzelnen Diagnoseabläufen waren recht schwer zu ermitteln. Die zur Verfügung gestellten Informationen des Unternehmens BMW haben oft nicht ausgereicht um bestehende Prozesse von DiaAccess zu verstehen.

Um schnell und effektiv voranzukommen, wurde in vielen kleinen Iterationsschritten entwickelt. Die entsprechenden Funktionalitäten für einfache und schnelle Aktualisierungen waren durch die ClickOnce-Bereitstellung<sup>21</sup> gegeben. Zu bestimmten Zeiten fand fast eine Art Extrem-Programming via Telefon statt.

Die Ermittlung der Gemeinsamkeiten und Abhängigkeiten in der Diagnose waren zum Teil sehr mühsam. Es dauerte etwa vier bis fünf Monate, um eine stabile Grundfunktionalität auf einem hohem Abstraktionsniveau für die Durchführung der Diagnoseaufgaben auf die Beine zu stellen.

## 5.5 Abläufe im Wasaka

Wie aus den vorhergehenden Kapiteln ersichtlich, ist die Diagnose relativ oft Veränderungen unterlegen, und diese Veränderungen müssen einen leichten und schnellen Weg ins Wasaka finden.

Im vorhergehenden Hauptkapitel konnte festgestellt werden, dass die gängigen Standards für unsere Anforderungen nicht wirklich tauglich sind. Um Wasaka auch von der Benutzerseite konfigurierbar bzw. erweiterbar zu gestalten, wurden gleich mehrere Wege gewählt.

---

<sup>21</sup> Mittels ClickOnce-Bereitstellung können Windows-basierte Anwendungen über einen Webserver für eine einfache Installation veröffentlicht werden: <http://msdn.microsoft.com/de-de/library/t71a733d%28v=VS.80%29.aspx> (09.04.2010)

### 5.5.1 Erweiterung durch Skriptsprachen

Die Erweiterung der Funktionalität durch Skriptsprachen hat bereits in vielen Vorgängerapplikationen erfolgreich die fehlenden Funktionalitäten ersetzt bzw. diese erweitert. Die BMW Mitarbeiter haben mit Hilfe von „Visual Basic Script“ (VBS) eigene Diagnoseaufgaben programmiert und damit die Arbeit automatisiert. Diese Möglichkeit sollte auch mit dem neuen Framework erhalten bleiben und noch stärker unterstützt werden.

Vorteile der Skriptsprachen:

- Bei BMW bereits bekannt und gelebt durch jahrelange Erfahrung
- Viele vorhanden Diagnose-Skripts, welche weiter verwendet werden können
- Unterstützung der Programmierung durch viele Beispiele, kostenlose Anleitungen und Tools aus dem Internet
- Vergleichsweise schnelle Ergebnisse, da während der Entwicklung wenig Overhead entsteht
- Kauf von vergleichsweise teureren IDE Lösungen wie Visual Studio oder anderen Lizenzen entfällt, da die entsprechenden Engines bereits in Windows integriert sind

Nachteile der Skriptsprachen:

- Schwache Unterstützung der Fehlerbehandlung, daher fehleranfälliger als Programmierung mit Compiler
- Vergleichsweise schwache IDE-Unterstützung (wie IntelliSense, etc.)

### 5.5.2 Erweiterung durch Konfiguration und Codegenerierung

Durch Codegenerierung wird die Erstellung von Scripts unterstützt, um die Diagnosefunktionen besser und mit weniger Fehler zu schreiben. Diese Art der Unterstützung wurde zunächst mit dem Initialprojekt Diagnose Parameter Assistent erprobt.

Was generiert wird, entscheidet der Skriptersteller mittels einer XML Deklaration. In dieser können Diagnosejobs samt Eingabe- und Ausgabeparameter festgelegt werden. Der Zugriff auf die Jobs des Steuergerätes und die Fehlerbehandlung kann dadurch vereinheitlicht werden.

```
<Workflow AUID="wf_001" N="Abgleichwert lesen" LL="D73N57B0">

  <Job AUID="job_001" N="FS_LESEN" R="1">
    <RESP AUID="req_001" N="F_ORT_NR" />
  </Job>

  <Job AUID="job_002" N="FS_LESEN_DETAIL" R="2">
    <REQ N="F_CODE" ST="R" V="req_001" />
  </Job>

  <Job AUID="job_003" N="STATUS_MESSWERTBLOCK_LESEN" R="3">
    <REQ N="MODE" ST="C" V="5" />
    <REQ N="MESSWERT" ST="V" V="label_name" />
  </Job>

  <Job AUID="job_004" N="ABGLEICH_LESEN" R="4">
    <REQ N="IDENTIFIER_ABGLEICH" ST="C" V="AGR" />
    <RESP N="ABGLEICH_LESEN_WERT" IN="eeprom_value" />
    <RESP AUID="resp_003" N="ABGLEICH_LESEN_WERT2" />
  </Job>

</Workflow>
```

**Abbildung 38: Wasaka – Beispielkonfiguration eines Diagnoseablaufs**

Die Eingangsparameter einer Funktion können auf verschiedene Wege festgelegt werden. Mit diesen Möglichkeiten bekommen die recht statischen Definitionen einen dynamischen Aspekt:

- Konstante Werte (siehe Job 3, Parameter 1)
- Variable Werte
  - Durch Zuweisung zur einer Skriptvariable (siehe Job 3, Parameter 2)
  - Durch Zuweisung eines Ergebnisses eines anderen Jobs (siehe Job 2)

Komplizierte Vorgänge können so vom Skriptersteller mittels Quellcode implementiert werden.

### **Grafische Unterstützung**

Das nötige XML muss natürlich nicht von Hand geschrieben werden. Wasaka bietet ein Modul zur Erstellung solcher Workflow Definitionen an und hilft dem Benutzer zum Beispiel mit dem Zugriff auf die Tabellen der SGBD.

### 5.5.3 Erweiterung durch Plug-In Codierung

Die früheren Entwickler der alten DiaAccess Software werden mit der Arbeit mit .NET und der neuen Bibliotheken geschult und können selbständig Module programmieren. Die Plug-In Entwicklung für Wasaka geht recht schnell.

```

<Dialog K="dlg-fehlerspeicher" N="" AXLanguage=".NET" Img="" Style="2"
  AXPlugin="Wasaka.PluginFehlerspeicher"
  AXAssembly="Wasaka.Plugins.dll">
  <AXAction Action="OnInit" Args="1" Method="DlgInit" />
  <AXAction Action="OnActivate" Args="0" Method="DlgActivate" />
  <AXAction Action="OnUnload" Args="0" Method="DlgClose" />
  <Table>
    <TR>
      <TD>
        <Control T="button-list" K="bl-5544" N="" H="" State="1">
          ...
        </Control>
      </TD>
    </TR>
    <TR>
      <TD>
        <Control T="sft-tree" K="sft-values" N="" H="12" W="30"
          GridStyle="5" State="1"
          Style="treelines-none generic-view no-production">
          <AXAction Action="OnMouseUp" Method="Sft_OnMouseUp" Args="1" />
          <AXAction Action="OnItemClick" Method="Sft_OnItemClick" Args="1"/>
          <AXAction Action="OnSelectionChange" Method="SftOnSChange"
            Args="1"/>
          <Columns>
            <Column N="Fehlerort / Umweltbedingung" RW="0.8" Fixed="1">
              <CT T="0" A="N"/>
            </Column>
            ...
          </Columns>
        </Control>
        <Control T="label" K="lbl-comment" N="Notiz" H="1" W="20" State="1"
          Style="fixed-height" />
        <Control T="multiline-text-box" K="mltb-comment" N="" H="3" W="30"
          State="1" Style="fixed-height" >
          <AXAction Action="OnChange" Args="1" Method="CommentOnChange" />
        </Control>
      </TD>
    </TR>
  </Table>
</Dialog>

```

**Abbildung 39: Wasaka – XML Definition des Dialogs Fehlerspeicher**

Das erste Element definiert den Dialog und das entsprechende .NET Plug-In. Das besondere am SDK ist aber auch die Möglichkeit die Plug-Ins mittels Skriptsprachen zu realisieren. So könnte der Code auch gleich in das XML mittels CDATA-Sektionen eingebettet werden.

Die so entstandenen Zusatzprogramme und -funktionen werden zur Laufzeit geladen, kompiliert und ausgeführt.

```
[PluginSignature("dlg-fehlerspeicher")]
public class PluginFehlerspeicher : GENPlugin
{
    private IGENAXTreeControl c_sft_values;

    public PluginFehlerspeicher(IGENAXEngine engine)
        : base(engine)
    {
        // Initialisierung der grafischen Komponenten
        c_sft_values = engine["sft-values"] as IGENAXTreeControl;
    }

    public void DlgInit(GENAXCancelEventArgs event_args)
    {
        base.DlgInit(event_args);
    }

    public void DlgActivate()
    {
        c_sft_values.VEVolumeExchange.SetVEList(...);
        c_sft_values.Reload();
    }

    public void DlgClose()
    {
    }

    public void Buttons_OnClick(GENAXClickEventArgs event_args)
    {
    }
}
```

**Abbildung 40: Wasaka – Minimalles Skelett eines Plug-Ins**

Die klaren Vorteile hier sind die Flexibilität und die Ausdrucksstärke durch Verwendung der bereits stark entwickelten Diagnose- bzw. Visualisierungsstrukturen. Das Plug-In ist jederzeit einsetzbar und muss lediglich ins entsprechende Verzeichnis nachkopiert werden.

Diese Art der Weiterentwicklung erfordert eine intensive Auseinandersetzung mit dem SDK und den damit verbundenen Bibliotheken und Strukturen. Die Programmierkenntnisse müssen an dieser Ebene recht hoch sein, damit die Entwicklung gewissen Qualitätskriterien entspricht und der Code leicht gewartet

werden kann. Zwar existieren viele Hilfsklassen für die Diagnose, aber die richtige Anwendung muss erst gelernt werden.

Bei den Mitarbeitern von BMW besteht daher die Gefahr, dass die Funktionalitäten nach gewisser Zeit erst recht von Experten neuprogrammiert werden müssen.

## 5.6 Zusammenfassung und Ausblick

Beim erstellen komplexerer Diagnosevorgänge führt noch immer kein Weg am eigentlich Codieren vorbei. Sowohl der Entwickler als auch der Benutzer, welcher die Softwarefunktionalität erweitern will, soll dabei aber bestmöglich unterstützt werden.

In der nächsten Zeit wird vorwiegend an dem Ausbau der Erweiterbarkeit und Konfigurierbarkeit von Wasaka gearbeitet. Dazu werden vor allem zwei Projekte genauer unter die Lupe genommen und auf Tauglichkeit überprüft:

### ➤ **Microsoft Windows Workflow Foundation**

Dieses erweiterbare Framework aus dem Hause Microsoft bietet Unterstützung in der Entwicklung von Workflow-Lösungen auf Windows Plattformen. Das Framework stellt sowohl eine API als auch Tools zum Entwickeln und Ausführen von workflow-gesteuerten Anwendungen zur Verfügung (Esposito, 2005).

### ➤ **NetBpm**

NetBpm ist die .NET Portierung des JBpm<sup>22</sup>. Das NetBpm ist ein freiverfügbares Framework und dient zur Ausführung und Verwaltung von Workflows. Es ist einfach zu verwenden und kann problemlos in andere .NET Applikationen integriert werden (Bolle, 2007).

Die Schnittstelle zum Scripting hat sich bereits mehrfach bewährt und wird noch durch Editoren und Codegenerierung verbessert.

Da die Firma ASE auch ein Know-How im Umgang mit Microsofts SharePoint Server besitzt, bekommen neu entdeckte SharePoint Server Erweiterungen, wie die des Unternehmens Skelta (Unterkapitel 4.1.2.1 BPMN und .NET) ganz neue Bedeutung und werden in ihrer Entwicklung weiter verfolgt.

---

<sup>22</sup> *JBpm ist eine flexible und mächtige BPM Engine geschrieben in Java:*  
<http://www.jboss.org/jbpm> (30.03.2010)

Die zukünftige Entwicklung des SDKs wird außerdem mehr in Richtung Rich-Internet-Application gehen, wobei die Microsoft Silverlight<sup>23</sup> Plattform zum Einsatz kommt. Die derzeit aktuelle Silverlight Version 4.0 (Beta) verspricht einige Verbesserungen vor allem im Bezug auf das Dateihandling, Drucksystem und Geschwindigkeit gegenüber der Version 3.0.

---

<sup>23</sup> *Silverlight ist ein kostenloses Browser Plug-In, das moderne Rich Internet Applications ermöglicht: <http://msdn.microsoft.com/de-de/silverlight/default.aspx> (10.04.2010)*

## 6 Literaturverzeichnis

---

Bachmann, J. (16. Juli 2009). BPEL - Wie werden meine Prozesse ausgeführt? Jena.

Bartonitz, M. (Juli 2007). BPEL - Business Process Execution Language.

Bartonitz, M. (Juli 2007). XPDL - XML Process Definition Language.

Bolle, J.-P. (14. Jänner 2007). *NetBpm*. Abgerufen am 06. November 2009 von <http://www.netbpm.org/>

BPMN. (kein Datum). *Business Process Modelling Notation*. Abgerufen am 16. Oktober 2009 von <http://www.bpmn.org/>

Day, M.-T., Bonati, F., Büttiker, P., & Lee, D. (Mai 2006). *Prozessmodellierung mit EPK*. Abgerufen am 05. Januar 2010 von <http://www.leed.ch/history/eepk/>

ebXML. (2006). *OASIS - ebXML*. Abgerufen am 07. Januar 2010 von <http://www.ebxml.org/geninfo.htm>

ebXML Requirements Team. (8. Mai 2001). ebXML Requirements Specification Version 1.06.

ebXML Technical Architecture Project Team. (16. Februar 2001). ebXML Technical Architecture Specification v1.0.4.

Esposito, D. (22. November 2005). *MSDN - Microsoft Windows Workflow Foundation*. Abgerufen am 10. März 2009 von <http://msdn.microsoft.com/de-de/library/cc431274.aspx>

Europäisches ebXML-Informationszentrum. (02. März 2009). *Die ebXML-Technologie*. Abgerufen am 06. Dezember 2009 von <http://www.ebxml.eu.org/Deutsch/Die%20ebXML-Technologie.htm>

Fischer, P. D. (26. November 2009). *Wiki der Fakultät Informatik und Wirtschaftsinformatik*. Abgerufen am 15. Januar 2010 von [http://www.iwiki.de/wiki/index.php/Erweiterte\\_ereignisgesteuerte\\_Prozesskette\\_%28eEPK%29](http://www.iwiki.de/wiki/index.php/Erweiterte_ereignisgesteuerte_Prozesskette_%28eEPK%29)

Freud, J., & Götzer, K. (2008). *Vom Geschäftsprozess zum Workflow: Ein Leitfaden für die Praxis*. München: Carl Hanser Verlag.



Keller, G., Nüttgens, M., & Scheer, A.-W. (1992). *Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“*. D - 66123 Saarbrücken.

Löschnigg, M. (2005). *Magisterarbeit: "XML in Workflow-Management-Systemen"*. Graz, Austria.

Mendling, J., & Nüttgens, M. (10. März 2005). EPC Markup Language (EPML). Wirtschaftsuniversität Wien.

Mendling, J., & Nüttgens, M. (November 2002). Event-Driven-Process-Chain-Markup-Language (EPML). Universität Trier, Deutschland.

Mendling, J., & Nüttgens, M. (März 2004). Exchanging EPC Business Process Models with EPML. Marburg, Deutschland.

Mendling, J., & Nüttgens, M. (März 2004). XML-based Reference Modelling: Foundations of an EPC Markup Language. Essen, Deutschland.

Nawrot, B. (20. April 2007). *BPEL 2.0 vereinfacht SOA-Entwicklung*. Abgerufen am 21. Januar 2010 von Computerwoche: <http://www.computerwoche.de/software/soa-bpm/591750/>

Nissen, V., Petsch, M., & Schorcht, H. (2008). *Service-orientierte Architekturen*.

OMG. (kein Datum). *BPMN Implementors And Quotes*. Abgerufen am 08. Februar 2010 von [http://www.bpmn.org/BPMN\\_Supporters.htm](http://www.bpmn.org/BPMN_Supporters.htm)

OMG. (2010). *Object Management Group*. Abgerufen am 16. Oktober 2009 von <http://www.omg.org/>

Owen, M., & Raj, J. (September 2003). *BPMN and Business Process Management: Introduction to the New Business Process Modeling Standard*. New York City, USA.

Pierre, S. (2007). *E-Learning Networked Environments and Architectures*. Springer-Verlag London.

RosettaNet. (2010). <http://www.rosettanet.org/>. Retrieved September 26, 2009, from [http://www.rosettanet.org/dnn\\_rose/Standards/RosettaNetStandards/tabid/473/Default.aspx](http://www.rosettanet.org/dnn_rose/Standards/RosettaNetStandards/tabid/473/Default.aspx)

Rossi, D., & Turrini, E. (September 2007). *EPML: Executable Process Modeling Language*. Bologna, Italy.

Scheer, A.-W. (September 1996). ARIS-House of Business Engineering. Saarbrücken, Deutschland.

Stapf, M. (Januar 2005). BPEL, das "SQL" der Prozesse. *Javaspektrum*, S. 54-57.

Teich, J., & Haubelt, C. (2007). Digitale Hardware/Software-Systeme.

Wagner, F. (28. Juni 2006). Proseminararbeit: BPEL - Business Process Executable Language. Martin-Luther-Universität Halle-Wittenberg, Deutschland.

Wallentowitz, H., & Reif, K. (2006). Handbuch Kraftfahrzeugelektronik: Grundlagen, Komponenten, Systeme, Anwendungen. Wiesbaden: Friedr. Vieweg & Sohn Verlag | GWV Fachverlage GmbH.

White, S. A. (Mai 2004). Introduction to BPMN. IBM Corporation.

White, S. A. (Januar 2004). Process Modeling Notations and Workflow Patterns. IBM Corporation, USA.

White, S. A. (Februar 2005). Using BPMN to Model a BPEL Process. IBM Corporation, USA.

Workflow Management Coalition. (Februar 1999). *Terminology and Glossary*. Abgerufen am 16. September 2009 von [http://www.wfmc.org/standards/docs/TC-1011\\_term\\_glossary\\_v3.pdf](http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf)

Workflow Management Coalition. (kein Datum). *Terminology and Glossary German*. Abgerufen am 9. März 2010 von <http://wfmc.org/Download-document/Terminology-and-Glossary-German.html>

Workflow Management Coalition. (2010). *Workflow Management Coalition - Homepage*. Retrieved Oktober 15, 2009, from <http://www.wfmc.org/>

Workflow Management Coalition. (10. Oktober 2008). *XML Process Definition Language Version 2.1a*. Abgerufen am 15. Oktober 2009 von <http://www.wfmc.org/xpdl-developers-center.html>

Workflow Management Coalition. (kein Datum). *XPDL Implementations*. Abgerufen am 02. Oktober 2009 von <http://www.wfmc.org/xpdl-implementations.html>

WSBPEL Technical Committee. (11. April 2007). *Web Services Business Process Execution Language Version 2.0: OASIS Standard*. Abgerufen am 10. Oktober 2009 von <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

zur Muehlen, M., & Recker, J. (16-20. Juni 2008). How Much Language is Enough? Theoretical and Practical Use of the Business Process Modeling Notation. 20th

International Conference on Advanced Information Systems Engineering (CAiSE 2008), Montpellier, France.