6

6 Converter

6.1 Grundlegender Aufbau eines Converters

Abbildung 6.1 Die Konversion als Informations-Transformation

Die Konversion (lat. convertere = umwenden, verwandeln) von Informationen ist im weiteren Sinne die Umsetzung von Informationen, die in einem Ausgangsformat vorliegen, in ein Zielformat welches wiederum eine Information darstellt (**Abbildung 6.1**). Converter übernehmen diese Aufgabe. Converter werden als Hardwarekomponenten und als Software realisiert.

Sehr oft werden Software Converter verwendet, um Daten von einem proprietären Format in ein standardisiertes Format zu übersetzten, wie zum Beispiel in die genormten EDI (Electronic Data Interchange) Formate.

Ein Compiler garantiert die Sematik des zu konvertierenden Inhalts. Er sorgt dafür, dass die Informationen, die in einer bestimmten Programmiersprache formuliert worden sind, korrekt und vollständig in die Zielsprache übersetzt werden. Der Konverter hingegen führt einen Umwandlungsprozess durch, bei dem es sowohl zu Informationsverlust als auch zur Informationsanreicherung kommen kann.

6.1.1 Der Converter als Hardware

Hardware Converter konvertieren Informationen aus der Umwelt zu Informationen, die von Computern verarbeitet werden können und umgekehrt. Ohne Konversion ist keine Interaktion eines Computers mit der Umwelt möglich. Da diese Konversion in Real-Time erfolgen muss, werden diese Konverter in den meisten Fällen als spezialisierte Hardware realisiert.

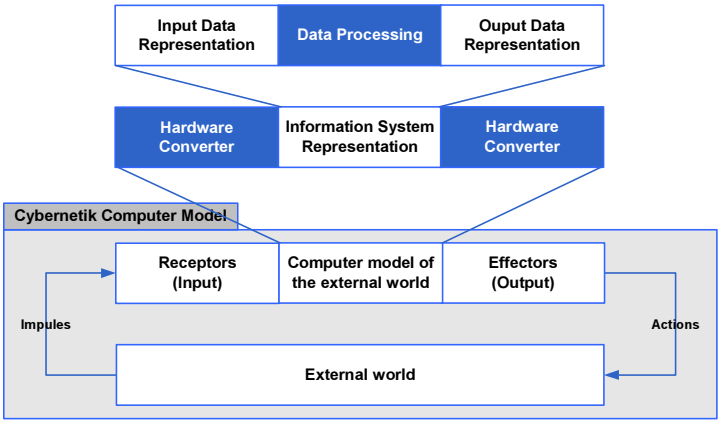

Abbildung 6.2 Der Hardware Converter als Teil des Kybernetischen Systems Computer

Der Hardware Converter kann als Teil des Kybernetischen Systems Computer, welches wiederum dem Modell des Menschen als Informationsverarbeitende Maschine entspricht, wie in **Abbildung 6.2** dargestellt werden (siehe Kapitel 1).

6.1.2 Der Converter als Software

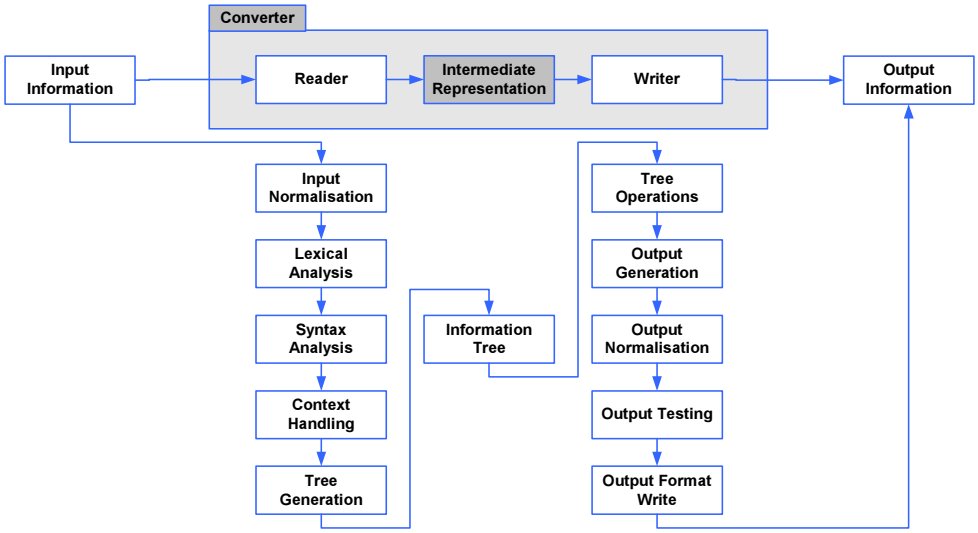

Abbildung 6.3 Generalisiertes Modell eines Software Converter

Der generalisierte Aufbau eines Software Converters ist demjenigen eines Compilers sehr ähnlich, wie in **Abbildung 6.3** dargestellt. Die Analyse findet im Reader des Converters statt, während die Synthese im Writer umgesetzt wird. Trotzdem wird der Compiler nicht

als eine spezialisierte Form eines Converters betrachtet, da er lediglich zur Übersetzung von Programmiersprachen eingesetzt werden kann, während der Software Converter vor allem für die Datentransformation eingesetzt wird.

- Der *Reader* liest die *Input Information* ein.
- Die *Input Normalisation* entfernt für die Konversion unwesentliche Informationen.
- Die *Lexical Analysis* prüft, ob die Input Information das vom Konverter akzeptierten Alphabet "erfüllt".
- Die *Syntax Analysis* prüft den Inhalt der Information.
- Das *Context Handling* reichert die Input Information mit zusätzlichen Informationen an.
- Die *Tree Generation* ist für den Aufbau der *Intermediate Representation* der Information zuständig.
- Die *Tree Operations* setzten die Konversion der Information um.
- Die konvertierte Information wird mittels *Output Generation* in das Zielformat umgewandelt.
- Die *Output Normalisation* entfernt für die Output Information unwesentliche Informationen.
- Mittels *Output Testing* wird die Korrektheit der *Output Information* geprüft.
- Die *Output Information* wird mit *Output Format Write* erzeugt.

6.2 Hardware Converter

Hardware Converter werden vor allem in Embedded Systems eingesetzt. Embedded Systems (Eingebettete System) sind Rechnersysteme, die integraler Bestandteil von Geräten, wie beispielsweise eines Videogerätes oder einer Kaffeemaschine, sind. Die wichtigsten Eigenschaften solcher Systeme sind der geringe Herstellungspreis, der niedrige Stromverbrauch, die Echtzeitfähigkeit und die Robustheit. Neben den einfachen Prozessoren, die für diese Systeme eingesetzt werden, sind Hardware Converter die wichtigsten Prozessorkomponenten.

Zwei Beispiele solcher Hardwarekomponenten sind die AD und DA Wandler und die DSP’s.

- *Analog/Digital* und *Digital/Analog Converter* als die zentrale Schnittstelle aller Informationsverarbeitenden Systeme zur Aussenwelt.
- *Digitale Signalprozessoren* als Grundbaustein der meisten Hardware Converter.

6.2.1 Analog/Digital Converter (ADC)

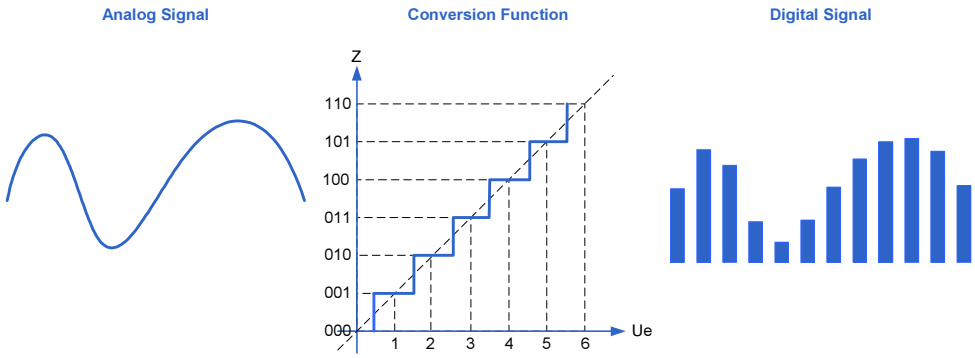

Abbildung 6.4 Grundsätzlicher Ablauf Analog/Digital Konversion

Analog Digital Converter sind elektronische Komponenten, die analoge Signale in entsprechende digitale Informationen umwandeln können (**Abbildung 6.4**). A/D-Wandler sind oft Bestandteil einer Interface-Elektronik.

Die Kenngrössen eines ADC sind [Biethan, 2003]:

- Die *Eingangsspannung* Ue, die das analoge Signal repräsentiert.
- Die *digitale Ausgangsgrösse* Z, die das digital umgesetzte Signal darstellt.
- Die *Conversion Function* f, welches die Umwandlung Z = f (Ue) erlaubt.

Es existieren drei "klassische" Konversionsverfahren für die Analog / Digital Umwandlung, das Parallelverfahren, das Wägeverfahren und das Zählverfahren.

- *Word at a Time (Parallelverfahren):* Kennzeichen des Parallelverfahrens, auch direkte Methode genannt, ist der gleichzeitige Vergleich des analogen Eingangssignals mit n verschiedenen Referenzwerten. Die Konversion erfolgt durch diesen Vergleich in einem einzigen Schritt und ist aus diesem Grund sehr schnell.
- *Digit at a Time (*Wägeverfahren*):* Dieses Verfahren, auch sukzessive Approximation (Successive Approximation Register) genannt, baut die digitale Abstufung des Ausgangssignals schrittweise auf. Die Anzahl der Vergleichsschritte entspricht der Auflösung des digitalen Ausgangs und ist aus diesem Grunde ein langsames Verfahren.
- *Dual Slope (Zählverfahren):* Das Zählverfahren zählt, wie oft eine Quantisierungsspannung Uref als Referenzgrösse aufsummiert werden muss, um dem Eingangssignal Ue zu entsprechen.

6.2.1.1 Successive Approximation Register Converter (SAR)

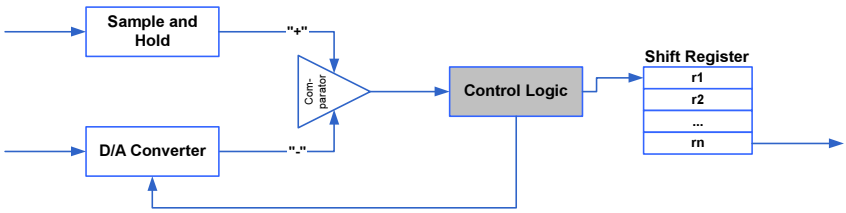

Abbildung 6.5 Successive Approximation Register Converter

Haupteinsatzgebiet von Successive Approximation Register Converter (SAR - **Abbildung 6.5**) sind Seismische Rekorder und andere Anwender, die einen sehr schnellen Durchsatz verlangen. SAR Converter arbeiten mit dem Wägeverfahren. Schnelle SAR Converter werden meist als Pipeline hintereinander geschaltet, um den gewünschten Durchsatz zu erreichen

6.2.1.2 Dual Slope Converter

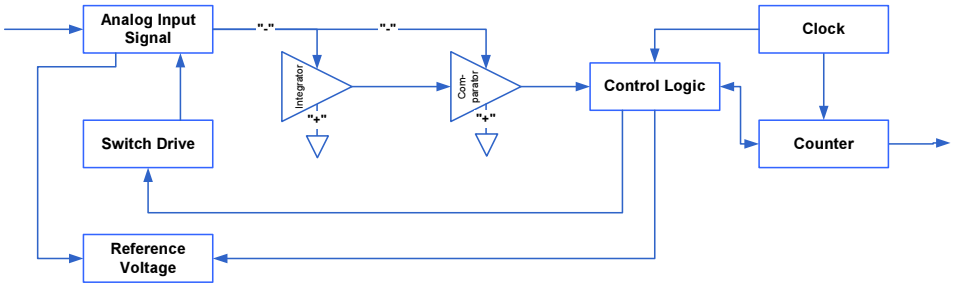

Abbildung 6.6 Dual Slope Converter

Der Dual Slope Converter wird vor allem in der Sprachdigitalisierung eingesetzt (**Abbildung 6.6**). Dual Slope Converter arbeiten mit dem Zählverfahren [Microchip 2004]. Sie sind relativ langsam, erlauben jedoch eine sehr hohe Auflösung.

6.2.1.3 Dynamische Effekte von AD Convertern

Alle A/D Converter sind mit einer Reihe von Hilfskomponenten ausgerüstet, die sich mit den dynamischen Effekten der Konversion auseinandersetzten müssen. Diese Effekte sind Störsignale infolge der Quantisierung, Fehler aufgrund der Unterabtastung (Aliasing) und Fehler aufgrund der Signaländerung während der Konversion.

6.2.1.4 Signal Noise Ratio (SNR)

Der Signal-Rausch-Abstand entsteht aufgrund der Treppenhöhe der Quantisierungsstufen. Wird ein Signal digitalisiert und anschliessend wieder in ein analoges Signal zurückverwandelt, so werden diejenigen Bereiche, die zwischen den einzelnen Treppenstufen des Samplings liegen, durch solche ersetzt, die exakt einer Treppenstufe entsprechen. Dieser Effekt wird Quantisierungsrauschen genannt und durch den Rauschabstand SNR bestimmt.

6.2.1.5 Effective Number Of Bits

Der Signal-Rauschabstand SNR hängt von der in Bit angegeben Auflösung n ab. Dieser Effekt verstärkt sich bei geringer Aussteuerung eines Signals. Dieser Effekt wird ENOB (Effective Number Of Bits) genannt.

Digital / Analog Converter (DAC)

Ein Digital Analog Converter (DAC) hat die Aufgabe, digital vorliegende Informationen, in eine analoge Grösse umzusetzen. Der Grundaufbau eines DAC entspricht demjenigen eines ADC.

Es existieren vier "klassische" Arten von Convertern zur Digital / Analog Umwandlung, die Converter mit Pulsbreitenmodulation, der Delta-Sigma DAC, der Binary Weighted DAC und der Segmented DAC.

- *Pulse Width Modulator (Pulsbreitenmodulation):* Das digitale Signal wird durch einen Zeitraum abgebildet. Ein stabiles Ausgangsignal wird also eine bestimmte Zeit (entspricht dem digitalen Eingang) lang als analoges Ausgangsignal gesendet. Diese Technik wird vor allem für Motorensteuerung und HiFi Anlagen verwendet.
- *Delta-Sigma DAC:* Delta-Sigma Converter setzen die "Pulsdichte" als Konversionstechnik ein. Dabei werden Abfolgen von zeitlich genau bestimmten Signalen mit einer bestimmten Dichte, basierend auf dem digitalen Inputsignal, erzeugt. Die meisten hochauflösenden DAC verwenden diese Technik.
- *Binary Weighted DAC:* Diese Converter verwenden eine 1:1 Abbildung des digitalen Eingangssignals durch einen einzigen analogen Ausgangswert. Diese Converter sind schnell und teuer, da sie sehr genau jedes einzelne Bit des Eingangs in eine Ausgangsspannung abbilden.
- *Segmented DAC:* Der Segmented DAC, besteht aus je einem Widerstand (current source segment) für jeden möglichen Ausgangswert. Ein "8 bit binary segmented DAC" besteht aus 256 Segmenten, ein "16 bit binary segmented DAC" besteht aus 65536 Segmenten.

6.2.1.6 Delta-Sigma DAC

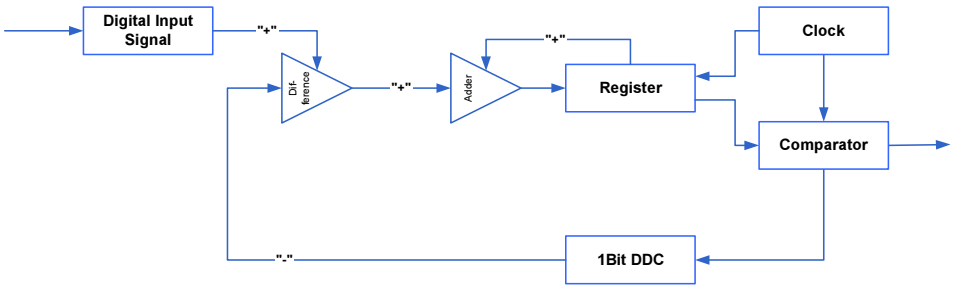

Abbildung 6.7 Delta-Sigma Converter

Ein Delta-Sigma DAC erzeugt aus der Pulsdichte des Digitalen Inputs die entsprechende Analoge Umsetzung (**Abbildung 6.7**). Delta-Sigma DAC's werden in digitalisierten Leistungsverstärkern eingesetzt [Beis 2007].

6.2.2 Kombinierte Systeme

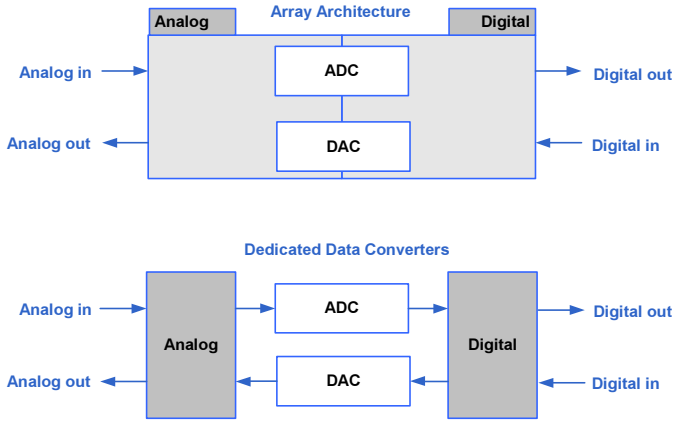

Abbildung 6.8 Kombinierter und Universeller Converter

DAC und ADC werden heute als einzelner Chip in Massen produziert und verkauft. Neue Ansätze versuchen sogar symmetrische Architekturen auf einem einzigen Chip zu realisieren, wie in **Abbildung 6.8** dargestellt. Dabei werden so genannte FPMA (Field-Programmable Mixed-Analog-Digital Arrays), das heisst Chips, deren Layout sich programmieren lässt, eingesetzt [Chow, Gulak 2005].

6.2.3 Digital Signal Processors

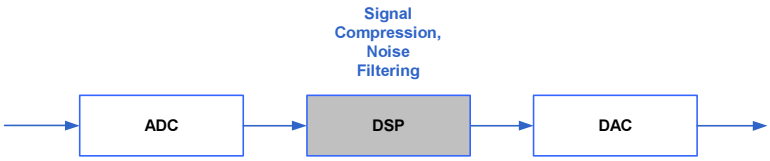

Abbildung 6.9 Grundlegende DSP Architektur

DSP (Digital Signal Processor - **Abbildung 6.9**) spielen im Bereich Hardware Converter eine zentrale Rolle. Die traditionelle Rolle der Signalverarbeitung als Basisfunktion für Modems, Musik Synthese und Rauschunterdrückung ist in den letzten Jahren durch neue Anwendungen erweitert worden. Diese Erweiterung des Einsatzgebietes von DSP's erfolgte durch die zunehmende Bedeutung moderner Applikationen im Bereich Wireless Technology und Audio/Video Elektronik.

Die wichtigsten Eigenschaften eines DSP sind:

- Ein DSP verarbeitet digital dargestellte Signale.
- Er weist eine spezielle Architektur zur Beschleunigung sich wiederholender numerisch intensiver Berechnungen auf.
- Er verfügt über eine Multiplikation-Akkumulation-Operation zur Beschleunigung von Signalverarbeitungsalgorithmen.
- Zur Berechnung des Kreuzprodukts wie es in digitalen Filtern zu berechnen ist.
- Ein DSP hat eine spezielle Speicher-Architektur für eine hohe Speicherbandbreite.
- Der DSP kennt spezielle Speicheradressierungsarten und spezielle Kontrollflussoperationen zur Beschleunigung von Schleifen.
- Er verfügt über spezielle On-Chip periphere Funktionen und Schnittstellen.

DSP’s gehören zur Familie der ASIP (Application Specific Instruction Set Processors), das heisst, sie verfügen über einen spezialisierten Instruktionsset, der auf eine bestimmte Anwendung zugeschnitten werden kann.

6.2.3.1 DSP Positionierung

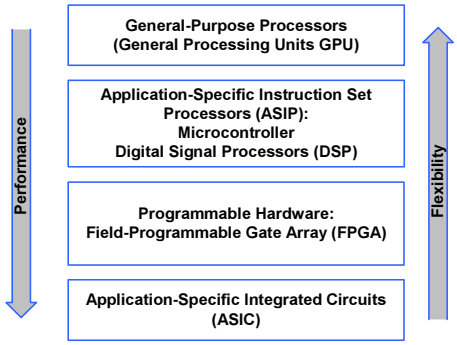

Abbildung 6.10 Die Positionierung eines DSP's in der Klassierung von Prozessortypen

Ein DSP kann im Rahmen einer Klassifizierung verschiedener Prozessortypen anhand den Kriterien Performance und Flexibilität wie in **Abbildung 6.10** dargestellt positioniert werden. Die Klassifizierung unterscheidet zwischen GPU, ASIP, FPGA und ASIC.

GPU

Eigenschaften der Klasse General-Purpose Prozessoren (GPU) sind:

- Hohe Performance durch Nutzung führender Technologien (gate count, clock rate)
- Nutzung von Parallelität
- Superskalar: dynamisches Scheduling von Instruktionen
- Superpipelining: Instruktionspipelining,branch prediction
- Mehrstufige Speicherhierarchie (caches)

Ein GPU hat jedoch Probleme mit Echtzeitanwendungen da die Ausführungszeiten nur schlecht vorhersagbar durch dynamische Effekte (scheduling, branch prediction, caching).

ASIP und Microcontroller

Eigenschaften der Klasse Application-Specific Instruction Set Processors (ASIP):

- Spezialisierung des Instruktionssatzes (z.b. durch Operatorverkettung "multiply-accummulate Instruktion")
- Spezialisierung der Funktionseinheiten (z.b. durch Pixel-Operationen,1/sqrt(x))
- Spezialisierung der Speicherarchitektur (z.b. durch mehrere Speicherbänke mit parallelem Zugriff)

Die Vorteile gegenüber GPU sind die höhere Performance, die niedrigeren Kosten (kleinere Chipfläche, weniger Pins), die geringere Codegrösse und die geringere Leistungsaufnahme

Die wichtigsten Eigenschaften von Microcontroller sind:

- Eigenschaften von steuerungsdominaten Anwendungen

- kontrollfluss-dominanter Code (Verzweigungen,Sprünge)
- Viele logische Operationen, wenige arithmetische Operationen
- Geringer Datendurchsatz
- Multitasking

Microcontroller sind optimiert für:

- Bit-und Logikoperationen
- Register sind oft im RAM realisiert: Kontextswitch durch Pointeroperation, dadurch minimale Interruptlatenz, schneller Kontextwechsel
- Periphere Einheiten integriert (ADC, DAC,Timerfunktionen)

6.2.4 DSP Grundlagen

6.2.4.1 Definition Digital Signal Processor

Ein DSP ist ein Prozessor, der digitale Signale verarbeitet. Generell sind DSP Funktionen mathematische Operationen auf Signalen, die in Echtzeit verarbeitet werden müssen.

6.2.4.2 Definition Signal

Das Signal ist eine Funktion, die eine Information über Zustand oder Verhalten eines physikalischen Systems wiedergibt. Die Darstellung eines Signals erfolgt als mathematische Funktion von einer oder mehreren unabhängigen Variablen.

Beispiele:

- amplitude(t): Sprache
- satturation(x, y): Bild

6.2.4.3 Definition zeitkontinuerliches Signal

Ein zeitkontinuerliches Signal ist Definiert im Zeitkontinuum. Die Darstellung eines zeitkontinuerlichen Signals erfolgt durch zeitkontinuierliche Variablen ('analoge' Signale).

6.2.4.4 Definition zeitdiskretes Signal

Ein zeitdiskretes Signal ist definiert zu diskreten Zeitpunkten. Die Darstellung eines zeitdiskreten Signals erfolgt durch eine Zahlenfolge (Sequenz)

Ein digitales Signal ist zeitdiskret und wertdiskret.

6.2.5 Harvard Architektur

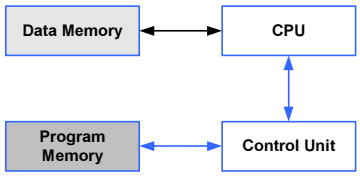

Abbildung 6.11 Harvard Architecture

Die Harvard Architektur umgeht das von Neumann Bottleneck durch die Trennung von Data Memory und Program Memory, wie in **Abbildung 6.11** dargestellt.

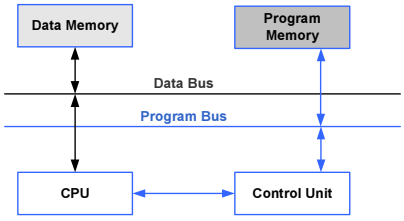

Abbildung 6.12 Advanced Harvard Architecture

Digital Signal Processors verwenden eine modifizierte Harvard Architektur, die einen direkten Zugriff auf Daten aus dem Programmspeicher heraus ermöglicht (**Abbildung 6.12**). Dies ist vor allem bei Koeffiziententabellen, die oft in sehr schnelle „nicht flüchtige" Speicher abgelegt werden, sehr nützlich.

6.2.6 Generelle Anforderungen an einen DSP

Die wichtigsten Anforderungen an einen DSP sind [Wagner 2002]:

- *Schnelle Multiplikation:* Als erstes fällt ins Auge, das sich das Ergebnis ergibt als Summe von Produkten. Jede zweite Operation ist also eine Multiplikation. Diese muss daher besonders effizient realisiert werden.
- *Schnelle Verfügbarkeit der Daten:* Weiterhin werden in jedem Summand zwei jeweils verschiedene Daten miteinander multipliziert. Diese müssen daher schnell zur Verfügung gestellt werden können.
- *Akkumulation und Multiplikation in einem Zyklus:* Innerhalb der Summe werden die Summanden stets zum vorherigen aufaddiert. Es wäre also wünschenswert, diese Akkumulation mit der Multiplikation in einem Schritt zusammen durchführen zu können.
- *Effiziente Schleifen:* Die Schleifengrenzen zum Berechnen der Summe stehen meist schon vor dem Start der Schleife fest. Daher wäre ebenfalls eine effiziente Schleifenbehandlung ohne überflüssigen Overhead wünschenswert.

- *Leistungsfähige Eingabe-/Augabeeinheit:* Da dauernd Daten mit der Peripherie ausgetauscht werden müssen, ist oft auch eine leistungsfähige Ein-/Ausgabe von nöten, die den restlichen Prozessorbetrieb auch nach Möglichkeit sehr wenig behindern soll.
- *Hohe Genauigkeit:* Es werden hohe Anforderungen an die Genauigkeit der Berechnungen gestellt. Daher ist z.B. eine hardwaremäßige Überlaufkontrolle oft sinnvoll.
- *Geringer Stromverbrauch/Preis:* In Hinsicht auf viele Anwendungen sind oft auch geringer Stromverbrauch und Preis wichtige Schlüsselqualifikationen, die ein DSP erfüllen muss.

6.2.6.1 DSP Architekturen der ersten Generation

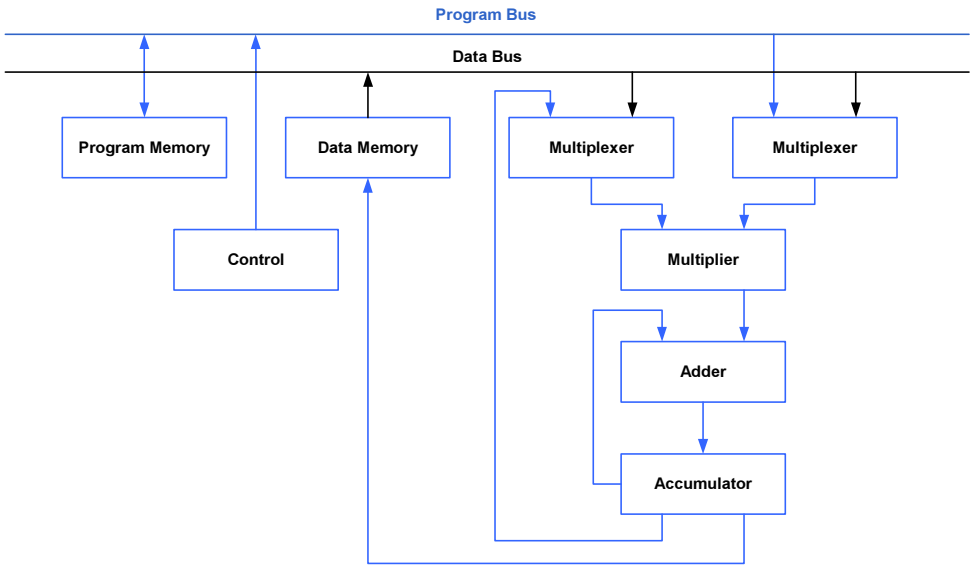

Abbildung 6.13 Digitaler Signalprozessor – die erste Generation [Tan, Heinzelmann 2005]

Die erste Generation der DSP Architekturen sind in den späten 70er Jahren des letzten Jahrhunderts entwickelt worden und weisen einen sehr einfachen Aufbau auf (**Abbildung 6.13**).

- Die schnelle Multiplikation ist die wichtigste Operation in der digitalen Signalverarbeitung. Beispiel: FIR (Kreuzprodukt zweier Vektoren), Konvolution, IIR Filter, Fourier Transformationen.
- Häufig in Verbindung mit Akkumulation der Produkte.
- Erster kommerziell erfolgreicher DSP: TI TMS32010 (1980).
- Spezielle Hardware zur Ausführung der Multiplikation in einem Taktzyklus.
- Nahezu alle DSPs enthalten Spezialhardware zur Ausführung der Multiplikation in einem Zyklus, vielfach kombiniert mit einer Multiply-Accumulate (MAC) Einheit.

6.2.6.2 DSP Architekturen der zweiten Generation

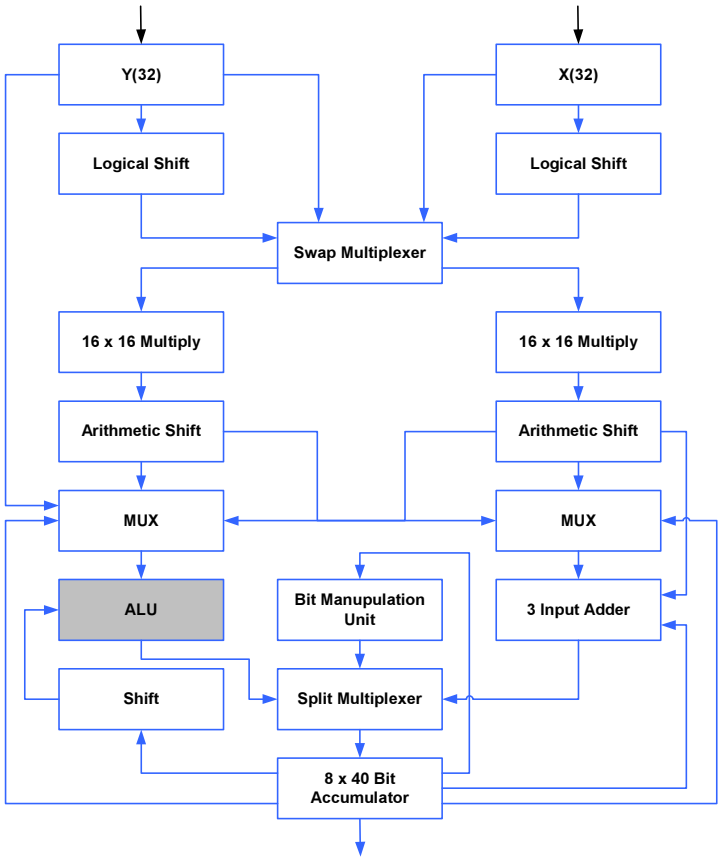

Abbildung 6.14 Digitaler Signalprozessor – die zweite Generation Lucent DSP 16 Reihe

Die DSP Prozessoren der zweiten Generation wurden Ende der 80er Jahre entwickelt und dienen noch heute als Basisarchitektur für die moderne Generation dieser Chips (**Abbildung 6.14**). Wesentliche Neuerungen waren Instruction Piplining, eine oder mehrere ALU (Arithmetic Logical Unit) und Akkumulatoren, um die Geschwindigkeit zu steigern.

Die Funktionsmerkmale der Lucent DSP 16 Reihe sind:

- Harvard-Architektur mit zwei Adress- und Datenbussen
- Dreistufige Instruktions-Pipeline
- Hardware-Multiplizierer für 16x16 Bit Operationen
- Zwei 36 Bit Akkumulatoren

- Instruction-Cache speziell für schnelle und effiziente Schleifenabarbeitung ohne Überhang durch Pipelining (zero-overhead looping)
- Gleichzeitiges Addieren vorher multiplizierter Werte, Ausführung einer 16x16 Bit Multiplikation und Holen des nächsten Wertepaares aus dem Speicher in einem Instruktions-Zyklus
- Gleichzeitiges Holen einer Programminstruktion und eines Datenworts aus dem internen RAM durch zwei separate Speicher-Steuereinheiten
- Flexible Konfiguration des internen ROM und internen Dual-Port RAM
- Serielle I/O-Ports für Multiprocessing

6.2.7 Vergleich DSP – GPU

Eine General Processing Unit, wie beispielsweise ein CISC oder RISC Prozessor, und ein DSP unterscheiden sich in folgenden Merkmalen [Falk 2004]:

Tabelle 6.1 Vergleich GPU mit DSP

General Processing Unit	Digital Signal Processor
Von-Neumann-Architecture	Advanced Harward Architecture
Allgemeine Adressierungsarten	Spezielle anwendungsbezogene Adressierungsarten
Control Unit, ALU und Register, zur allgemeinen Verwendung	Spezielle Control Units und Register zur Adress- und Datenmanipulation neben der ALU
Nur ein interner Datenbus	Mehrere Datenbusse zum parallelen Transport von Daten

6.3 Software Converter

Software Converter sind ein sehr verbreitetes Instrument in der Informatik. Die Anwendungen sind derart vielfältig, dass davon ausgegangen werden kann, dass in jedem grösseren System mindestens ein Converter existieren muss.

Klassische Anwendungsgebiete dieser Converter sind die Sprachübersetzung, der Schnittstellenbau und die Formatumwandlung.

6.3.1 Datenkonversion

Abbildung 6.15 Data Converter

Die Datenkonversion spielt vor allem im Bereich Business Intelligence und Data Warehousing eine wichtige Rolle. Zunehmen gewinnen Datenkonversionen im Rahmen unternehmensweiter Informations-Architekturen an Bedeutung. So sind in jedem System, welches basierend auf einer SOA (Service Oriented Architecture) realisiert wird, solche Komponenten zu finden.

6.3.1.1 Der schematische Prozess der Datenkonversion

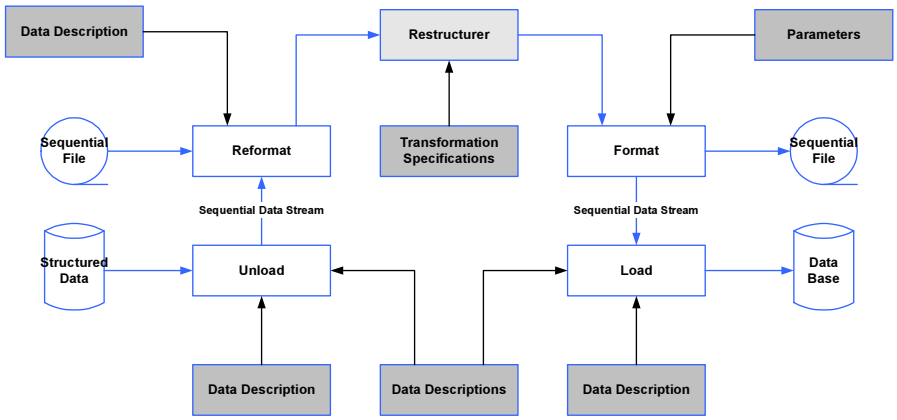

Abbildung 6.16 Schematischer Prozess der Datenkonversion

Bereits 1981 wurde im Rahmen einer Arbeitsgruppe des IEEE ein Assessment der verschiedenen Datenkonversions-Methodiken durchgeführt [Fry et al. 1981]. Die Vertreter und Vertreterinnen der wichtigsten großen Informatikfirmen (IBM, HP, Bell Labs, Honeywell) und Universitäten haben ein generelles Modell zur Datenkonversion entwickelt, das heute noch in den meisten ETL (Extract Transform Load)-Softwarekomponenten Verwendung findet. Der Prozess ist in **Abbildung 6.16** dargestellt.

Dieses Modell ist heute erweitert durch das Konzept der Metadaten – Daten, die Daten beschreiben. Selbstverständlich gibt es in komplexeren unternehmensweiten Informations-Modellierungen eine zusätzliche Erweiterung: die Meta-Meta Daten, d.h., Daten die die Beschreibung von Daten beschreiben. Diese Informationen sind außerordentlich nützlich, wenn ein unternehmensübergreifender Informationsaustausch stattfinden soll.

6.3.1.2 Beispiel: ETL Prozess für die Datenübernahme

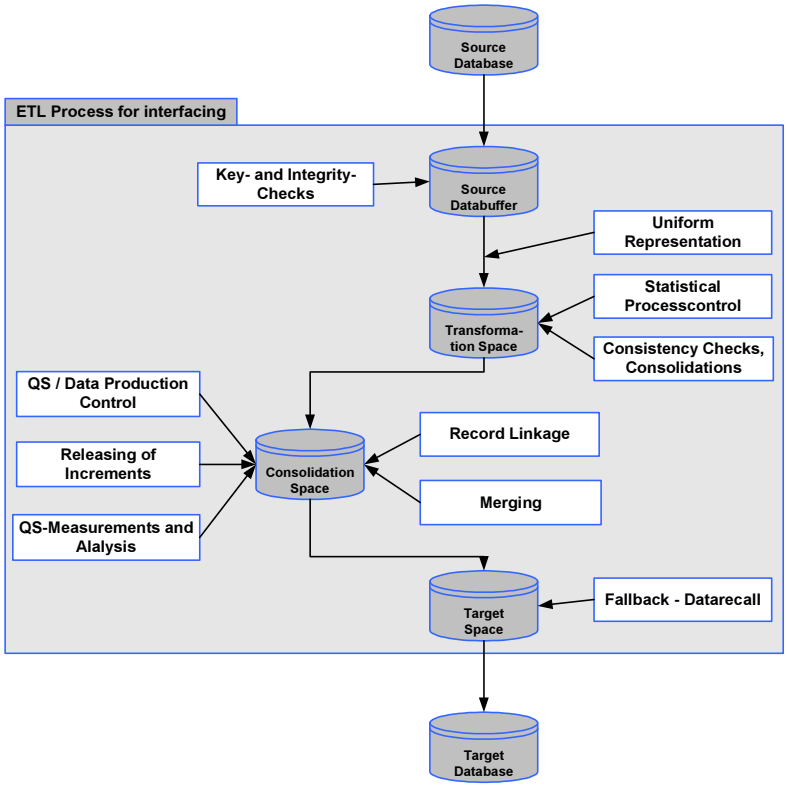

Abbildung 6.17 ETL Prozess für eine Datenübernahme

Die Übernahme von Daten ist in diesem Beispiel mit einer Konvertierung der Daten und einer Neustrukturierung der einzelnen Tabellen der Datenbank verbunden, da das Datenbankschema der Source Database sich vom Schema der Target Database unterscheidet. ETL Prozesse verarbeiten in den meisten Fällen sehr grosse Datenmengen, wie zu Beispiel sämtliche Bestellungen eines Versandhauses oder auch sämtliche Telephone aller Kunden eines Telekommunikationsunternehmens.

Ein ETL Prozess für die Datenübernahme arbeitet mit 4 logischen Datenbereichen (**Abbildung 6.17**):

- *Source Databuffer:* Die Daten werden unverändert aus der Source Database übernommen. Auf diesen Daten werden Integritätsprüfungen ausgeführt. Die Daten werden, falls inkonsistent in eine uniforme Repräsentation überführt.
- *Transformation Space:* Die Transformation wird in diesem Bereich vorgenommen. Dabei werden die Transformationsprozesse überwacht durchgeführt und immer wieder Konsistenzprüfungen und Konsolidierungsschritte ausgeführt.

- *Consolidation Space:* Die Transformierten Daten werden in diesem Bereich konsolidiert und in das Datenbankschema der Target Database überführt. Das Zusammenführen einzelner Records ist dabei der wichtigste Schritt.
- *Target Space:* Dieser Datenbereich ist ein Spiegel der Target Database. Letzte Prüfungen stoppen allenfalls den ganzen Prozess, oder starten die Durchführung des ETL-Laufs neu.

6.3.2 Transformational Data Integration

Transformational Data Integration ist eine dynamische Variante der traditionellen ETL (Extract Transform Load) Technik zur Datenintegration zwischen einer Datensource (typischerweise eine Datenbank) und einem Datentarget (typischerweise ein Data Mart oder ein Data Warehouse). Die zentrale Idee hinter Transformational Data Integration ist die dynamische Übertragung ganz bestimmter Teile der Daten, die durch definierte Events gesteuert wird. Die Technik wird auch CTF (Capture, Transform and Flow) genannt.

Die Vorteile von CTF sind:

- *Mehrstufige Ad-Hoc Query Unterstützung:* Ziel-Systeme können dynamisch neue Datenbestände anfordern.
- *Aktualität:* Datenlieferanten können nur die geänderten Datenbestände nachliefern, respektive Zielsysteme können Aktualisierungen verlangen.
- *Robust:* Der asynchrone Datenaustausch ist stabiler als die in Batch-Läufen durchgeführte ETL Technik, da im Fehlerfall nicht ganze Datenbestände nocheinmal bearbeitet werden müssen.

Eine andere Anwendung von CTF ist das so genannte „Continuous Mirroring“, das heisst die schnelle Nachführung von parallelen Datenbeständen.

6.3.3 Modell Driven Conversion

Model-Driven Conversion basiert auf einer abstrakten Sicht der Unternehmensdaten und versucht die Konversion und damit auch die Integration von Datenarchitekturen durch Modellierung zu erreichen. Ein wichtiger Ansatz zur Modellierung kommt von der OMG (Object Management Group). Die OMG präsentiert mit der Meta Object Facility (MOF) einen Standard zur Modellierung so genannter PIM’s (Plattform Independent Model) [OMG 2002]. Dabei wird ein Meta-Metamodel, also ein Modell, welches Metamodelle beschreibt, definiert. Der Vorteil dieser auf UML aufbauenden Technik ist die Unabhängigkeit der Modellierung von der Realisierungstechnologie. So eignet sich das objektorientierte Metamodell (Klassen-Attribute) nun mal nicht zur Modellierung von relationalen Daten, die basierend auf dem Metamodell Tabelle-Schlüssel definiert werden.

Ein interessantes Beispiel einer Model-Driven Integration ist die Federal Enterprise Architecture (FEA) der amerikanischen Behörden, respektive des Office of Management and Budget (OMB). Dieses seit Februar 2002 entwickelte mehrstufige Modell erlaubt die Kon-

version und Integration von Datenarchitekturen zwischen verschiedenen amerikanischen Behörden [OBM 2007].

6.3.4 Beispiel 1: Converter Engine

Die Erstellung von Schnittstellen zwischen Systemen ist eine zentrale Aufgabe der Informatik und der wichtigste Bestandteil jeder Integrationslösung. DRaCoMaP (Data Rendering and Converting for Mapping and Processing) ist ein Framework zur Konvertierung und zum Mapping von Daten.

Die Eigenschaften diesers Frameworks sind:

- DRaCoMaP ist ein modulares Framework. Dieses Framework ist die ideale Basis für Integrationsprojekte, die Daten aus einem System beziehen, diese umwandeln und abbilden, um sie schliesslich in ein Zielsystem anzuliefern.
- DRaCoMaP ist ein universelles Framework.
- DRaCoMaP ist Plattformunabhängig. Es wird auf UNIX-Systemen (AIX, Linux, Solaris, HP-UX), auf Windowsumgebungen (NT, 2000, XP, .NET) und auf Hostrechnern eingesetzt.
- DRaCoMaP ist Herstellerunabhängig. Es kann nahtlos in Standardprodukte integriert werden (CORBA, mqSeries,...).
- DRaCoMaP unterstützt standardisierte Datenformate. Heute werden XML, SQL, COBOL und CSV unterstützt, weitere Formate sind vorgesehen.
- DRaCoMaP basiert auf dem "Pipes & Filters" Architekturpattern der "Gang of Five" (Buschmann, Meunier, Rohnert, Sommerlad, Stal).
- DRaCoMaP ist flexibel dank der internen Datenrepräsentation in einer Baumstruktur, auf die transparent über RDN (Relative Distinguished Name) zugegriffen werden kann.

6.3.4.1 Grundmodule

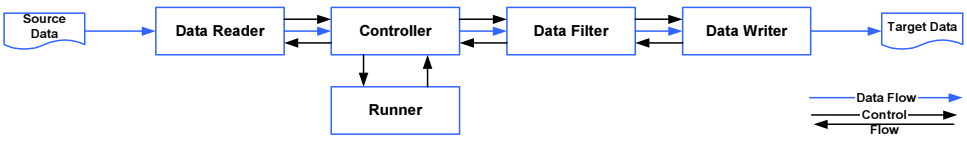

Abbildung 6.18 Einfaches Pipeline Beispiel

Die Funktionsweise einer Pipe-and-Filter-Interface-Architektur stellt **Abbildung 6.18** dar. Der Ablauf einer Schnittstelle geht folgendermaßen vor sich:

- Der *Data Reader* liest Daten ein (aus einer Datei oder einer Datenbank) und baut die interne Datenrepräsentation auf.
- Der *Controller* koordiniert die anderen Module und garantiert den geordneten Ablauf der Pipeline.

- Über den *Runner* wird eine Pipeline gesteuert. Er stellt die Schnittstelle nach aussen dar. Sämtliche Konvertierungs- und Mappingregeln werden als Konfigurationsdateien eingelesen und an die anderen Module weitergegeben.
- Der *Data Filter* manipuliert die interne Datenrepräsentation und führt damit alle notwendigen Mappings und Konvertierungen durch.
- Der *Data Writer* schreibt die konvertierten Daten in ein Zielsystem (Datei oder Datenbank).

6.3.4.2 Schnittstellen

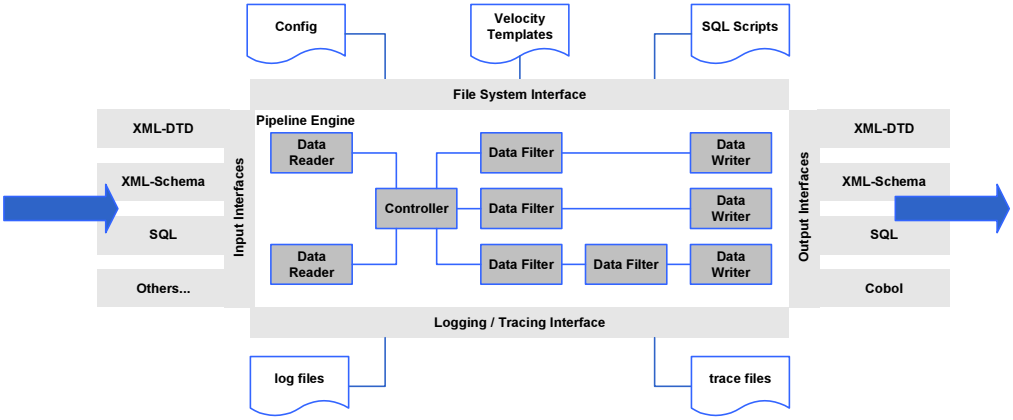

Abbildung 6.19 Interfaces der Converting Engine

Die verschiedenen Schnittstellen des Frameworks sind in **Abbildung 6.19** dargestellt

6.3.4.3 Baukastenprinzip

- Beliebige Pipelines können wie in einem Baukasten aus einzelnen Modulen zusammengesteckt werden.

6.3.4.4 Konfiguration

DRaCoMaP ist auf zwei Ebenen konfigurierbar:

- Die Konfiguration jedes einzelnen Moduls erlaubt Anpassungen an eine bestehende Umgebung, ohne die Komponenten zu ändern.
- Der Aufbau einer Pipeline Engine wird konfiguriert, um die Komplexität der Konvertierung und des Mappings auf einzelne Module zu verteilen. So werden einzelne Schritte spezifizierbar und kontrollierbar.

Die Konfiguration erfolgt im XML-Format.

6.3.4.5 Einsatzgebiet

- Anbindung von Legacy-Systemen an Online-Plattformen.
- Verbindung verschiedener Legacy-Systeme.
- Konsolidierung von unternehmensweiten Datenbeständen.
- Umwandlung von Standardformaten (EDI, FpML, EBXML, IATA, SWIFT) in Proprietäre Formate (SAP, unternehmens-spezifisches XML).
- Erstellung von Technologiebrücken (Legacy - Middleware, Legacy - .NET, J2EE - Legacy, COBOL - CORBA, hierarchische Datenbank - relationale Datenbank).
- Erweiterung bestehender EAI-Infrastrukturen.

Die Erweiterbarkeit der Grundarchitektur von DRaCoMaP erlaubt eine kundenspezifische Umsetzung weiterer Einsatzgebiete.

6.3.5 Beispiel 2: Eine Adaptor Factory

Das vorgestellte System beschreibt die Ergänzung der Standardprodukte " INSURANCE - Provision" und eventuell " INSURANCE - Organisation " um Adapterkomponenten (Konversion) und eine Produktionssteuerung (Orchestrierung).

6.3.5.1 Adapterfactory

- Die Adapterfactory ermöglicht für die schnelle und kostengünstige Realisierung der Datenkommunikation zwischen „INSURANCE-Provision“ („INSURANCE-Organisation“) und Zuliefersystemen (Primärsysteme), respektive Zielsystemen (Vertriebssysteme).
- Die Adapterfactory setzt die XML-Technologie zur Spezifikation der Datenkommunikation ein.
- Die Adapterfactory ermöglicht die unternehmensweite Standardisierung des Datenmanagements zwischen Systemen und den INSURANCE-Produkten durch die formale Definition des XML Transportformates.
- Die Adapterfactory garantiert den fehlerfreien Transport von Daten über verschiedenste Plattformen und Netzwerke. Der Datentransport kann auch über Standardprodukte wie mqSeries, entireX, etc. erfolgen.

6.3.5.2 Produktionssteuerung

- Die Produktionssteuerung steuert und überwacht den gesamten Prozess der Provisionierung.
- Die Produktionssteuerung fügt sich nahtlos in bestehende Überwachunngssysteme wie beispielsweise HP Open-View, BMC und Tivoli ein.

6.3.5.3 Ausgangslage

Die meisten Betriebe haben heute eine heterogene Systemumgebung. Neben Legacy Systemen (Hosts) sind verschiedenste Lösungen mit verschiedenen Architekturen und Datenbeständen nebeneinander im Einsatz. Die Produkte "INSURANCE-Provision" und "INSURANCE-Organisation" müssen sich nahtlos in diese bestehenden Systeme einfügen.

Der reibungslose Datenverkehr zwischen diesen IT-Systemen und den INSURANCE-Produkten ist eine wichtige Voraussetzung für die Abwicklung bestehender und zukünftiger Geschäfte.

Laut einer Analyse von Forrester Research umfasst die Entwicklung von Schnittstellen 35% des Gesamtaufwandes der Programmierung. Bei Punkt-zu-Punkt-Integrationen entfallen sogar 70% des Gesamtaufwandes für den Informa-tionsaustausch.

Die Steuerung und Überwachung des Gesamtsystems (INSURANCE-Produkte und Zuliefer- respektive Abnehmersysteme) ist eine wichtige Voraussetzung für den reibungslosen und kostengünstigen Betrieb der INSURANCE-Produkte.

Die TCO (Total Cost of Ownership) kann durch eine Produktionssteuerung erheblich gesenkt werden. Die INSURANCE-Produkte werden durch die Produktionssteuerungs-Komponente überwachbar, verwaltbar und robuster.

6.3.5.4 Zielsetzung

Anforderungen an die Adapterfactory:

- Das generische Format aller Daten, die zwischen verschiedenen Systemen und „INSURANCE-Provision“ und „INSURANCE-Organisation" ausgetauscht werden, ist XML
- Die Daten müssen fehlerfrei und zeitgerecht angeliefert werden
- Jedes System, welches Daten erhält, muss diese Daten in einem definierten Zielformat zur Verfügung gestellt bekommen
- Sämtliche Daten werden aus einem oder mehreren Systemen exportiert
- Verschiedene Plattformen sind zu unterstützen (Windows NT, Linux, Solaris, AIX, HP-Unix, MVS, OS-X)
- Die Lösung muss skalierbar, wartbar, robust und fehlertolerant sein
- Gegebenenfalls muss die Kommunikation zwischen den verschiedenen Systemen durch ein Messaging System oder ein entsprechendes Standardprodukt (Middleware) ersetzt werden können
- Anforderungen an die Produktionssteuerung:
- Das System muss sich nahtlos in eine bestehende Systemmanagement-Umgebung einfügen können (HP-OpenView, Tivoli, BMC, etc.)
- Sämtliche Prozesse müssen "remotely" verwaltbar sein (start, stop)

- Tests müssen zur Produktionszeit durchgeführt werden können, ohne das laufende System zu beeinträchtigen
- Alle Aktivitäten müssen überwacht und gegebenenfalls rapportiert werden können

6.3.5.5 Systemübersicht

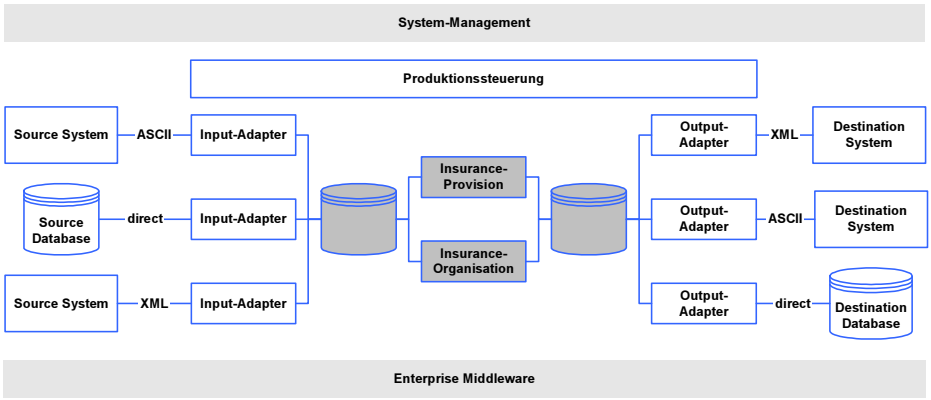

Abbildung 6.20 Systemübersicht Adaptor Factory

Die Adaptor Factory besteht aus folgenden Komponenten (**Abbildung 6.20**):

- *Source System:* Ein Primärsystem, welches Daten an „INSURANCE-Provision“ oder „INSURANCE-Organisation" liefert. Das Datenformat kann beliebig sein.
- *Source Database:* Die Daten werden direkt aus einer Datenbank ausgelesen. Mögliche Datenbanken sind: DB2, Oracle 7.1 oder höher, MS SQL Server, Informix, Sybase oder IMS.
- *Input-Adapter:* Einzelne Ausprägungen der Adapterfactory, die Daten konvertieren und an die INSURANCE-Datenbank anliefern.
- *Output-Adapter:* Einzelne Ausprägungen der Adapterfactory, die Daten aus der INSURANCE-Datenbank ausle-sen konvertieren und an die Zielsysteme anliefern.
- *Destination System:* Zielsystem des Kunden oder des Geschäftspartners des Kunden. Das Datenformat kann beliebig sein.
- *Destination Database:* Die Daten werden direkt an eine Datenbank angeliefert. Mögliche Datenbanken sind: DB2, Oracle 10g oder höher, MS SQL Server 2005, MySQL oder IMS.
- *Produktionssteuerung:* Steuerung, Überwachung, Reporting und Online Testing der Adapter und der INSURANCE-Systeme.
- *Enterprise Middleware:* Bestehende Middleware des Kunden. Beispielsweise mqSeries, entireX, Tibco, SeeBeyond, EJB-Server, etc.

- *System-Management:* Systeme zur Überwachung, die der Kunde einsetzt. Beispielsweise HP OpenView, Tivoli, BMC, Nimbus etc...

6.3.5.6 Komponenten der Adaptorfactory

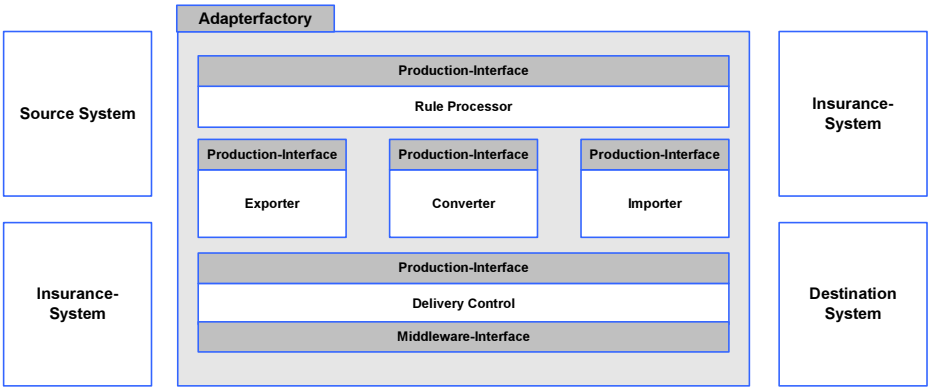

Abbildung 6.21 Logische Komponenten der Adaptor Factory

Die Adaptorfactory als Ganzes konvertiert nicht nur Daten und Informationen, sondern enthält auch eine Reihe von Utilities, die als Unterstützung der produktiven Umgebung dienen. Diese Komponente, das Delivery Control überwacht den Ablauf der Konversion zur Laufzeit (**Abbildung 6.21**).

- Der *Exporter* liest Daten aus einem Primärsystem oder aus „INSURANCE-Provision“, respektive „INSURANCE-Organisation" aus und liefert Sie über die Delivery Control Komponente an den Converter.
- Der *Converter* wandelt die vom Exporter erhaltenen Daten in ein Zielformat um und gibt sie über die Delivery Control Komponente an den Importer weiter.
- Der *Importer* liefert die Daten an das Zielssystem oder an "INSURANCE-Provision", respektive "INSURANCE-Organisation" an.
- Der *Rule-Processor* dient zur Konfiguration des Exporters, des Converters und des Importers. Die Konfiguration erfolgt über XML-DTD's, Templates zur Konversion und SQL-Scripts zur Interaktion mit den Datenbanken.
- Das *Production-Interface* ist die Schnittstelle für die Produktionssteuerung. Über diese Schnittstelle werden Control- und Monitorinformationen, Logs und Traces sowie Testdaten kommuniziert.
- Das *Middleware-Interface* ist die Schnittstelle zur unternehmensweiten Middlewarekomponente.

6.3.5.7 Komponenten der Produktionssteuerung

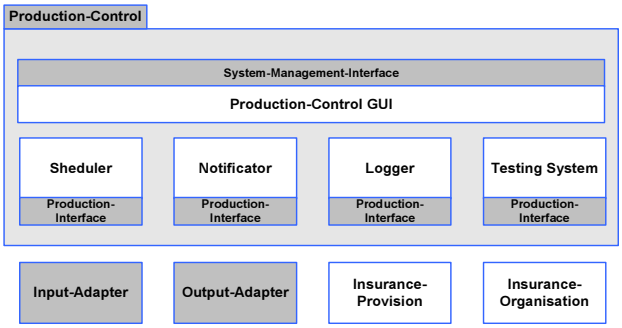

Abbildung 6.22 Logische Komponenten der Produktionssteuerung

Die Produktionssteuerung ist als Service realisiert (**Abbildung 6.22**).

- Der *Scheduler* steuert die einzelnen Komponenten. Er startet beispielsweise eine Komponente zu einem definierten Zeitpunkt. Er kann jedoch auch auf einzelne Events reagieren.
- Der *Notificator* dient zur Systeminformation im Ausnahmefall. So kann eine Mail abgesetzt werden, falls eine Komponente ein bestimmtes Signal sendet (z.b. Connection-Unterbruch).
- Der *Logger* steuert das Tracing und das Logging der einzelnen Komponenten. So können zur Laufzeit gezielt einzelne Komponenten überprüft werden. Der Logger steuert ausserdem das Reporting des Systems.
- Das *Testing-System* erlaubt eine Überprüfung des laufenden Systems. Es können während der normalen Produktion Testdaten gesendet und ausgewertet werden.
- Das *Production-Control GUI* erlaubt die interaktive Konfiguration und Steuerung aller Komponenten der Produktionssteuerung
- Das *System-Management Interface* interagiert mit Standard System-Management Software über SNMP V2 Meldungen.
- Das *Production-Interface* ist für die Kommunikation mit allen Komponenten der Adapterfactory und der "INSURANCE-Provision" und "INSURANCE-Organisation" Software zuständig.

6.4 Fragen zum Kapitel

Nr	Frage
1	Was ist der Unterschied zwischen dem Generallisierten Modell eines Software Modells und dem konzeptionellen Aufbau eines Compilers?
2	Was sind die wichtigsten Vorteile eines ASIP gegenüber einen GPU?
3	Was ist das so genannte von Neumann Bottleneck und wie kann es umgangen werden?
4	Was genau sind Metadaten und wie verändern sie den schematischen Prozess der Datenkonversion?
5	Welches ist das Einsatzgebiet von ETL Prozessen?
6	Welche Alternativen gibt es zum konventionellen ETL Prozess?
7	Die im Kapitel beschrieben Converter Engine ist auf dem Pipe & Filter Prinzip aufgebaut. Welche Alternative würden Sie wählen, um die Maschine mit einer Process oder Workflow Engine umzusetzen?
8	Die im Kapitel beschriebenen Komponenten der Produktionssteuerung unterstützen klassische System-Management Funktionen. Welche sind das?
9	Nennen sie die gebräuchlichen Verfahren für AD und DA Wandler.
10	Welches sind die Anwendungsgebiete der Converter Technologie?