Automatisierungstechnik in der industriellen Automation - Zusammenfassung

Daniel Belles

1 Vorwort

Dieses Dokument diente lediglich zur eigenen Prüfungsvorbereitung. Es besteht kein Anspruch auf Korrektheit und Vollständigkeit. Dennoch kann es den ein oder anderen vielleicht bei der Prüfungsvorbereitung unterstützen.

2 Einführung

2.1 Information

Unterrichtung über Sachverhalte, Ereignisse oder Abläufe. Stellt eine eigene Entität neben Masse oder Energie dar.

2.2 Automatisierung

Mensch braucht weder ständig noch in erzwungenem Rythmus für den Ablauf tätig werden.

2.3 Technischer Prozess

- Stückprozesse — eher Fertigungstechnik
- Kontinuierliche — eher Verfahrenstechnik
- Chargenprozesse (Abgegrenzte Stoffmengen, diskontinuierlich)

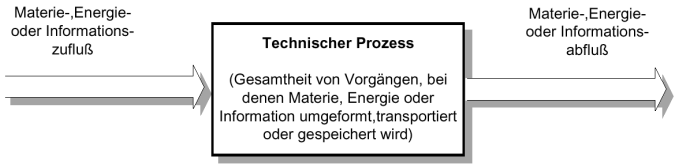


Figure 1: technischer Prozess

2.4 Automatisierungspyramide

- **Feldebene:** Sensoren, Aktoren hängen über Feldbus zusammen
- **(Steuerungsebene:** SPS - Steuerung und Regelung

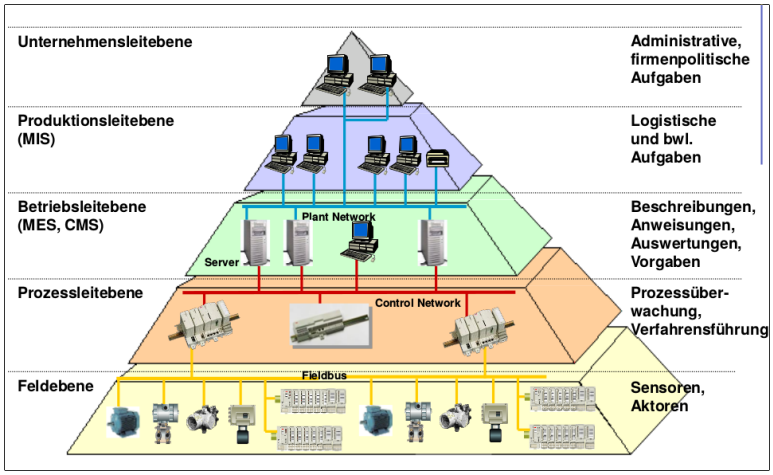


Figure 2: Informationsstrukturen in der AT

- **Prozessleitebene:** Bedienen und Beobachten, Rezeptverwaltung und Ausführung, Messwertarchivierung
  MES: Manufacturing Execution System - Betriebs-, Maschinen- und Personaldatenerfassung

- **Betriebsleitebene:** Betriebswirtschaftliche Aufgaben, Produktionsdatenerfassung, Logistik, ...

- **Unternehmensleitebene:** Administrative, Firmenpolitische Aufgaben, zB. Bestellabwicklung

3 Leitsysteme und SPS

3.1 Prozessleittechnik (PLT)

Messen, Steuern, Regeln, Leitung von verfahrenstechnischen Produktionsprogrammen Es bestehen 2 Systeme je nach Einsatzgebiet. Die Grenzen zwischen diesen verschwimmen heutzutage immer mehr.

3.1.1 Speicherprogrammierbare Steuerung (SPS)

Flexibel einsetzbar. Programmierung durch Sprachen der IEC 61131-3 (IEC: International Electrotechnical Commission)

3.1.2 Prozessleitsystem (PLS)

komplexe propritäre Systeme für verfahrenstechnische Prozesse, deutlich Leistungsstärker.

3.2 Steuerungstechnik

3.2.1 Steuern

Die Eingangsgrößen beeinflussen durch bestimmte Gesetzmäßigkeiten die Ausgangsgröße.
Steuerung nimmt zu jedem Zeitpunkt einen bestimmten Zustand ein - Endlicher Automat / Zusatandsautomat.

3.2.2 Regeln

Die Ausgangsgröße wird ständig gemessen, mit einem Sollwert verglichen und entsprechend korregiert.

3.2.3 Verknüpfungssteuerung

Steuerung, bei der den Signalzuständen der Eingangssignale bestimmte Signalzustände der Ausgangssignale im Sinne von boole'schen Verknüpfungen zugeordnet werden. Eingesetzt wenn Bedingungen unabhängig vom zeitlichen Ablauf sind.

3.2.4 Ablaufsteuerung

Steuerung läuft schrittweise ab, es existieren Weiterschaltbedingungen (Transitionen), die für das Weiterschalten zum nächsten Schritt erfüllt sein müssen.

3.3 Zyklischer / Azyklischer Programmablauf

Beim Zyklischen Programmablauf, werden zu Anfang alle Eingangsgrößen gespeichert, dann verarbeitet und anschließen in einem Vorgang die Ausgangsgrößen geschrieben. Beim Azyklischen Programmablauf, werden parallel zu den zyklischen Automatisierungsprogrammen auch azyklische Tasks direkt vom Systemteil verwaltet (Eingangsgrößen werden direkt Abgefragt - zB. Interrupts).

3.4 Abtasttheorem von Shannon

Die Zyklenfrequenz (Abtastfrequenz mit der die Eingänge in den Speicher gelesen und verarbeitet werden) der SPS muss min. zwei mal so hoch sein, wie die Signalfrequenz der Eingänge.

3.5 Bauformen SPS

- klassische SPS
- Steuerungsintegriert (Computerized Numberical Control / Robot-Control)
- Soft-SPS mit PC
- Slot-SPS im PC (Karte)

3.6 Aufgaben / Anforderungen PLT

- langfristig gute Wartbarkeit
- vermeidung negativer Einflüsse auf die Umwelt
- sichere Prozessführung
- Erfassung und Protokolierung aller Prozessschritte und Umweltwerte (besonders Pharma- und Lebensmittelindustrie, Kraftwerke)

3.7 Sicherheitsmaßnahmen - Redundanz

3.7.1 Redundanz

zusätzliches Vorhandensein von funktional gleicher Resourcen eines technischen Systems, die im störungsfreien Betrieb überflüssig sind.

- **Hot-Standby:** Mitlaufen einer redundanten Komponente mit stoßfreier Übernahme im Fehlerfall
- **Hot-Swap:** Austausch einer defekten redundanten Komponente im laufenden Betrieb

3.7.2 Arten von Redundanz

- **fail-safe:** System fährt in sicheren Zustand sobald ein Fehler erkannt wird. Billige Realisierung, falls ein Produktionsstop keine großen Verluste mit sich bringt.
- **Hot-Standby:** Bei Fehlererkennung wird auf parallele Komponente umgeschaltet.
- **2oo3 - two out of three:** Es wird der Ausgabe vertraut, die bei min. 2 von 3 parallelen Systemen übereinstimmt.

3.7.3 Vorhalten von Redundanz:

- **Workby:** Beide Komponenten laufen mit, error-detection über komparator.
- **Hot-Standby:** Redundante komponente rechnet nicht mit sondern wird erst zugeschaltet, falls ein Fehler vorliegt.
- **Cold-Standby:** Reservekomponente ist nicht integriert, sondern wird nach Fehler ausgetauscht (Bsp. Glühbirne)

3.8 Leitsysteme

3.8.1 Aufgaben Leitsystem

- bestimmungsgemäßes Verhalten eines Prozesses
- keinerzeit eine Gefährdung für Mensch und Umwelt
- bei Störungen muss sicherer Betrieb gewährleistet sein

3.8.2 Leittechnik

Alle nötigen technischen Komponenten die nötig sind um den Prozess entsprechend zu führen.

- Prozess Komponenten
  - Sensoren
  - SPS
  - Aktoren
- Anzeige- und Bedienelemente

4 Leitsysteme und HMI

4.1 Human-Machine-Interface

Interdisziplinares Arbeitsgebiet - Informatik, Psychologie, Soziologie, Ingenieurwissenschaften.

4.1.1 Menschliches Gedächtnis

- Sensorischer Speicher:
  - vollständiges Abbild der Realität
  - sehr kurze Speicherdauer
- Kurzzeitgedächtnis:
  - bereits interpretierte Ereignisse -¿ komprimierung
  - zeitspanne von 10..15 Sekunden
  - beschränkung auf 7 +/- 2 Einheiten
  - Verdrängungsspeicher - neue Ereignisse verdrängen alte
- Langzeitgedächtnis:
  - geringe Detaildichte
  - enthält autobiografische Informationen aus dem Leben
  - speichert Wissensbausteine

4.1.2 Menschliches Handeln

Der Mensch handelt aufgrund von Wissen und Erfahrung. Es wird nach bereits bekannten Mustern gesucht und anhand von gespeicherten Regeln für Aufgaben das Handeln eingeleitet.

4.1.3 Handlungsfehler

- Fehler bei bewussten Handlungen
  - **Denkfehler:**
    Zielkonflikte werden in der Planung nicht erkannt bzw. Berücksichtigt.
    Fehler wird erst nach Abschluss der Handlung festgestellt.

  - **Merk- und Vergessensfehler:**
    Handlungsplan wird nicht so lange im Gedächtnis gehalten, bis die Handlung vollständig abgeschlossen ist. Das kann dazu führen, dass bestimmte Teile eines Handlungsplans nicht umgesetzt werden.
  - **Urteilsfehler:**
    Fehlendes Wissen um auf Rückmeldungen entsprechend zu reagieren.
  - **Wissensfehler:**
    Die gespeicherten Handlungsmuster entsprechen nicht den tatsächlichen Gegebenheiten.

- Fehler bei Routinehandlungen
  - **Gewohnheitsfehler:**
    Handlungsmuster wird in der falschen Situtation ausgeführt.
  - **Unterlassensfehler:**
    Handlungsschritte werden übersprungen.
  - **Erkennensfehler:**
    Rückmeldung wird nicht erkannt obwohl das Erkennungsmuster erlernt und gut beherrscht wird.

4.2 Usability

Maß das angibt, wie "gebrauchstauglich" ein bestimmtes Produkt im Nutzungskontext durch bestimmte Benutzer verwendet werden kann. Es sollen bestimmte Ziele effektiv, effizient und zufriedenstellend erreicht werden.
Dazu sollten die späteren Anwender stark in die Systemgestaltung mit eingezogen werden. (Benutzerbefragungen, Usability-Testing, Expertenmeinungen)

4.3 Anzeigengestaltung

- Abstände zwischen Elementen gleich
- Gesamtbild trägt zur Nutzung des räumlichen Erinnerungs- / Orientierungsvermögens bei
- dezente Verwendung von Farben
- Rot-/Grünschwäche berücksichtigen
- Farbe nie alleine als Codierung, nur zur Unterstützung verwenden.

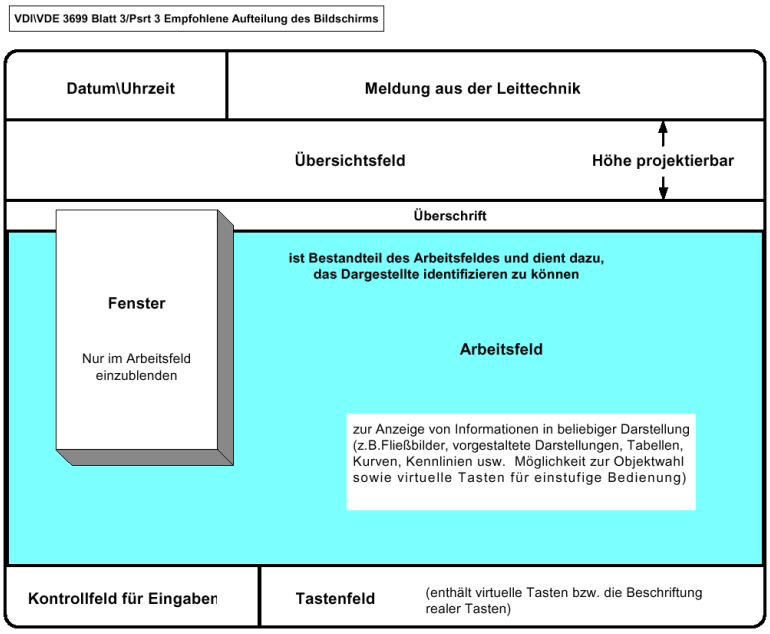


Figure 3: Bildschirmaufteilung nach VDI/VDE 3699 Blatt 3

Es sind bestimmte Symbole und Diagramme in der VDI Norm festgelegt, die zur gestaltung der Leitwarten- Anzeige verwendet werden sollen (zB. Lampen, Balkenanzeigen, ...).

4.4 Alarmmanagement

Zu viele Alarme führen dazu, dass die wichtigen Alarme nicht wahrgenommen werden. Andererseits ist jeder Alarm hilfreich für die Prozessführung. Es muss ein kompromiss für die richtige Anzahl von Alarmen gefunden werden. Alarmmanagement ist ein **kontinuierlicher** Prozess.

4.4.1 Eigenschaften eines Alarms

- **relevant**
- **eindeutig**
- **zeitgerecht**
- **priorisiert**

- **verständlich**
- diagnostisch
- hinweisend
- fokusierend

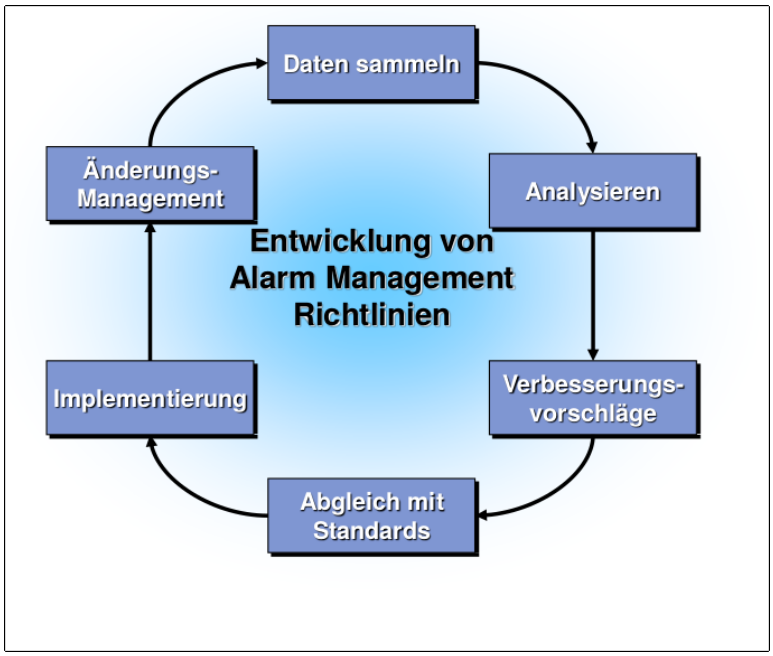


Figure 4: kontinuierlicher Verbesserungsprozess

Empfohlene Priorisierung: hoch-mittel-niedrig: 5%-15%-80%. Priorität wird aus Schadenshöhe und zeitlicher Relevanz kalkuliert. Ein Ziel sollte die Beseitigung unnötiger Alarme sein durch zB.:

- Beseitigung von Hardwarefehlern
- Reglertuning
- Anpassung Alarmgrenzen
- Vermeidung Redundanter Alarme
- Vermeidung Flatteralarme durch Filter oder Totzonen

5 Feldbusse

5.1 Interface

Festgelegte Datenübertragungsverbindung zwischen zwei Systemen. Umfasst Protokolle die festlegen, über welche Art und Weise die Information ausgetauscht wird.

5.1.1 Eigenschaften einer Schnittstelle

- mechanische Eigenschaften
  - Steckerart
  - Steckerbelegung
- elektrische Eigenschaften
  - Signalpegel (Strom, Spannung)
  - Frequenzen
- funktionelle Eigenschaften
  - Kodierung, d.h. Bedeutung der Signale
- Protokoll, d.h. Abfolge der Signale

5.2 Bus

Festgelegte Datenübertragungsverbindung zwischen mindestens zwei Geräten. Alle Komponenten eines Datenverarbeitungssystems hängen an einer Sammelleitung. Der Datenaustausch erfolgt im Multiplexbetrieb.

5.3 Bussysteme

- **Systembus:** Verbindung von Chips zu Systemen/Rechner
- **Peripheriebus:** Verbindung Rechner zu Ein-/Ausgabegeräten
- **Feldbus:** Verbindung von Fahrzeugkomponenten
- **LAN (Local-Area-Network):** Verbindung von Einheiten in Firmen, Gemeinden
- **WAN (Wide-Area-Network):** Wie LAN nur weitläufiger

5.4 Parallele Bussysteme

Besitzen mehrere Leitungen, auf denen zeitgleich unterschiedliche funktionen des Systems realisiert werden. Das gerät kann als Master oder Slave fungieren.

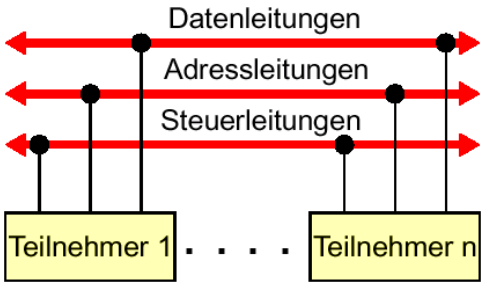


Figure 5: paralleles Bussystem

- **Datenbus:**
  Eigentliche Datenübertragung
- **Adressbus:**
  Auswahl einzelner Geräte und Adressen innerhalb der Geräte
- **Steuerbus:**
  Busanforderung, Interrupts, Handshaking
- **Versorgungsbus:**
  Stromversorgung und Taktleitung

5.5 Serielle Bussysteme

Besitzen nur eine Übertragungsleitung. Die parallelen Funktionalitäten beim Parallelbus werden seriell durch Softwareprotokolle realisiert. Sie sind in der Regel flexibler in Bezug auf Änderungen. In der Regel werden anstatt Datenwörter wie beim Parallelbus Datenframes mit Header übertragen.

5.6 Erwartungen an Feldbus

Der frühere Standard für industrielle Übertragungssysteme war 4-20mA. Mit Bussystemen spart man dem gegenüber bis zu 25% Kosten ein.

- Raue Bedingungen

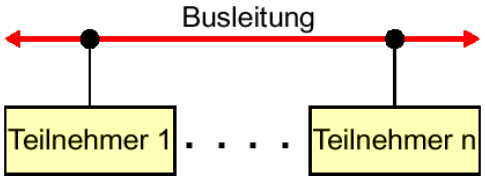


Figure 6: serielles Bussystem

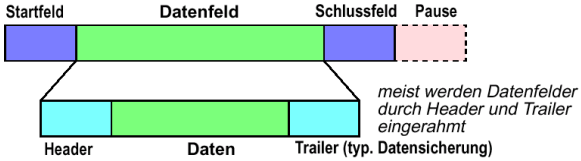


Figure 7: datenblock

- Einfache Installation
- Hohe Ausfall- und Übertragungssicherheit (Integrität)
- Geringe Erweiterungskosten
- Mittlere Übertragungsraten (50kbit/s .. 5Mbit/s)
- Hohe Übertragungsdistanzen
- Erhöht die Modularität der Anlage
- Kostenreduzierung

5.7 Feldbustypen

5.7.1 Steuerungs-Feldbusse

- Überwiegend firmenspezifische Lösungen oder Ethernet
- Meldungsbasierter Datentransfer
- TCP/IP gewinnt an Bedeutung
- Übertragungsraten bis 100 Mbit/s
- Entfernungen von einigen m bis in km-Bereich

5.7.2 Standard I/O-Feldbusse

- Mehrere defacto Standards
- Übertragungsraten bis 12 Mbit/s
- Entfernungen von einigen m bis in km-Bereich
- Reichhaltiges Angebot an Komponenten

5.7.3 Prozess-Feldbusse

- Zuverlässigkeit und Verfügbarkeit stehen im Vordergrund
- Entfernung km-Bereich und mehr
- Auslegung für aggressive/sensible Umgebungen
- I/O orientierter Transfer mit komplexen Parametriermöglichkeiten
- Spezielles Peripherieangebot für die Prozesstechnik

5.8 Dezentrale Automatisierung

Der Trend geht weg von einer zentralen Steuerungsintelligenz hin zu verteilter dezentraler Intelligenz direkt in Prozessnähe. Das spart Verkabelungsaufwand und hält das ganze System modularer und Wartbarer.

Vorteile der Dezentralisierung

- weniger Verdrahtung
- einfachere Modularisierung
- höhere Widerverwendbarkeit von Software
- einfachere Erweiterbarkeit
- offene Kommunikation, d.h. einfachere Kombinierbarkeit heterogener Systeme

Nachteile der Dezentralisierung

- Softwarestruktur wird komplexer
- Notwendigkeit von Kommunikations- und Schnittstellenoverhead
- Synchronisation und Datenableich

5.9 HART - Highway Addressable Remote Transducer

Digitale Bus-Kommunikation über bereits vorhandene 4-20mA Verkabelung. Dem analogen 4-20mA Signal wird ein FSK (Frequency-Shift-Keying) Digitalsignal aufgeprägt. Es können dadurch beide Systeme (analog / digital) parallel genutzt werden.

5.10 OSI-Schichtenmodell

OSI (Open System Interconnection) ist ein Schichtenmodell das versucht die einzelnen Komponenten einer Netzwerkkommunikation zu abtrahieren. Es ist aus insgesamt 7 Schichten aufgebaut, wobei nicht immer alle Schichten in einer Kommunikation tatsächlich notwendig sind.

Figure 8: OSI Schichtenmodell

5.10.1 Anwendungsschicht / Application Layer [VII]

Interaktion mit Anwendungen, die Netzwerkzugriff benötigen, Server-Client-Anwendungen. Bereitstellung von Diensten für Endanwender. Dateiübertragung, Verzeichnisverwaltung, ...

5.10.2 Darstellungsschicht / Presentation Layer [VI]

Umwandlung der Datenstruktur, Kodierung, Konvertierung, Kompression z.B. MPEG, ...

5.10.3 Sitzungsschicht / Session Layer [V]

Anforderung von Sitzungen und Datenströmen, Verschaffen von Zugang zu anderem Rechnersystem, Steuerung der Kommunikationsrichtung, Einbau von Synchronisations- und Wiederanlaufpunkten.

5.10.4 Transportschicht / Transport Layer [IV]

Flusskontrolle, Verbindungsauf- und -abbau, TCP- und UDP-Protokoll, Datenaufbereitung für die Vermittlungsschicht: Zerlegung/Zusammensetzung in Datenpackete, Zusammenlegen von mehreren Transportverbindungen auf eine Netzwerkverbindung.

5.10.5 Vermittlungsschicht / Network Layer [III]

Routing, logische-Adressierung, IP-Protokoll

5.10.6 Sicherungsschicht / Data Link Layer [II]

Flusssteuerung, Zugriffssteuerung, MAC-Adressen, Quittierungsmechanismen, Fehlererkennung

5.10.7 physikalische Schicht / Physical Layer [I]

Übertragungsmedium (Kuper- und Glasfaserkabel, Funk), Stecker, Signalformen, Funkfrequenzen, ...

5.11 Netzwerkkomponenten

5.11.1 Repeater / Hub (OSI I)

Kopplung von Bussegmenten zur Verlängerung

5.11.2 Bridge (OSI II)

Werden zur Verbindung von Subnetzen eingesetzt, die auf der Sicherungsschicht mit denselben Protokollen arbeiten. Layer I der beiden Subnetze kann verschieden sein. Datenverkehr wird gefiltert und nur weitergeleitet, falls die Datenpakete ihr Ziel auch erreichen können.

5.11.3 Switch (OSI II)

Kollisionsfreier Netzwerkknoten. Vergleichbar mit einem Hub, mit jeweils einer Bridge an jedem Port. Wird ein Hub an einen Port des Switches angeschlossen, muss sich die interne Bride an diesem Port alle am Hub angeschlossenen MAC-Adressen merken.

5.11.4 Router (OSI III)

Verbinden Subnetze die sich in Schicht I und II unterscheiden. Verwendet Protokoll aus Schicht III (IP-Protokoll). Kein Broadcast möglich. Routing steuert die Nachrichten durch das Netz. Wegentscheidung aufgrund von Routingtabellen.

5.11.5 Gateway

Kopplung zweier Netze unabhängig vom OSI Layer. Je nach Layer in dem die Kopplung geschieht, sieht die Umsetzung des Gateways sehr unterschiedlich aus.

5.12 Profibus

Implementiert normalerweise die OSI Schichten I, II, (III), VII.

5.12.1 Profibus FMS (Field Message Specification)

war vor allem für den Einsatz in komplexen Maschinen und Anlagen gedacht. Diese Protokollvariante wurde von DP abgelöst und ist heute nicht mehr Bestandteil der Internationalen Feldbusnorm.

5.12.2 Profibus DP (Dezentrale Peripherie)

Am häufigsten eingesetzte Variante des Profibus. Zur Ansteuerung von Sensoren und Aktoren durch eine zentrale Steuerung und Vernetzung mehrerer Steuerungen untereinander.

5.12.3 Profibus PA (Prozess-Automation)

In der Prozessleit- und Verfahrenstechnik eingesetzte Variante des Profibus. Geringere Übertragungsraten als DP aber dafür deutlich robuster und sicherheitsunkritischer. Die Energieversorgung der Teilnehmer erfolgt direkt über die Busleitungen (Digitale Alternative zu 4-20mA)

5.13 CAN - Controller-Area-Network

Robuster Bus für den Einsatz im Kraftfahrzeug. Definiert die Layer I und II im OSI-Schichtenmodell.

5.13.1 Arbitrierung

Es können alle Teilnehmer gleichzeitig auf dem Bus senden, bis sich die CAN-ID in einem Bit unterscheidet. Die 0 ist dabei dominant und überschreibt die 1 der anderen Teilnehmer. In dem Moment wo das eigene Bit nicht-mehr mit dem tatsächlichen Wert auf dem Bus übereinstimmt weiß der Teilnehmer, dass ein anderer mit einer höheren Priorität sendet und bricht seine Übertragung ab. Je kleiner die CAN-ID des Frames ist, desto höher wird dieser also priorisiert.

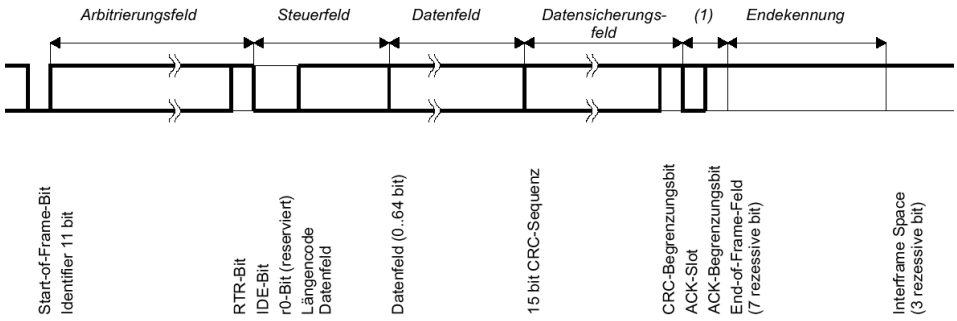


Figure 9: CAN Frame 11-Bit

6 Programmiersprachen nach IEC 61131

Es sind in der DIN EN 61131 verschiedene Grundbausteine definiert, die von den Compilern zur Verfügung gestellt werden. Dazu zählen zum Beispiel Aufwärts-/Abwärtszähler, Timer, Einschaltverzögerungen, Echtzeituhr, ...

6.1 Programmiersprachen

- Programmierung der SPS
- definierte Datentypen (Standarddatentypen) und Variablenattribute (const, retain, input, output, ...)
- textuelle und grafische Programmierung möglich

- Standardisierung ermöglicht teilweise konvertierung von Code in andere Sprachen

DIN EN 61131-3		Ältere Sprachen nach DIN 19239 bzw. VDI 2880

AWL – Anweisungsliste
IL – Instruction List
	`LD eingang_1`
	`AND eingang_2`
	`ST ausgang_0`
		AWL – Anweisungsliste

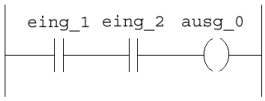
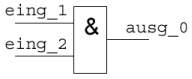
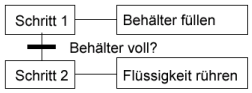
ST – Strukturierter Text ST – Structured Text	`C := A AND NOT B`	
KOP – Kontaktplan LD – Ladder Diagram		KOP – Kontaktplan
FBS – Funktionsbaustein-Sprache FBD – Function Block Diagram		FUP – Funktionsplan
AS – Ablaufsprache SFC – Sequential Function Chart		Funktionsplan nach DIN 40719, Teil 7

Figure 10: Programmiersprachen nach DIN 6-1131

6.1.1 Ladder Logic (Kontaktplan)

Angelehnt an Kontaktplan von klassischer Schütz/Relai-Verdrahtung. Es gibt Verbindungen, Kontakte, Spulen, Sprünge, Funktionsaufrufe, ... Es existiert eine Vertikale Stromschiene links und rechts. Verbindungen werden durch horizontale Linien realisiert. Zustand des Verbindungselementes wird gemäß dem "Stromfluss" bezeichnet (EIN/AUS). Zustände werden immer von links nach rechts übertragen.

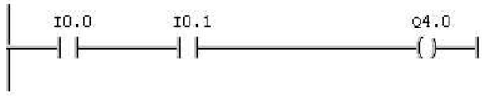


Figure 11: Kontaktplan AND-Verknüpfung

Figure 12: Kontaktplan OR-Verknüpfung

6.1.2 Function Block Diagram (FBD)

Besonders in Europa etabliert. Bestehend aus Verbindungen, Elementen zur Ausführungssteuerung (Sprünge), Funktionsaufrufen, ... Logikgatter statt einzelnen Schaltern (Kontaktplan).

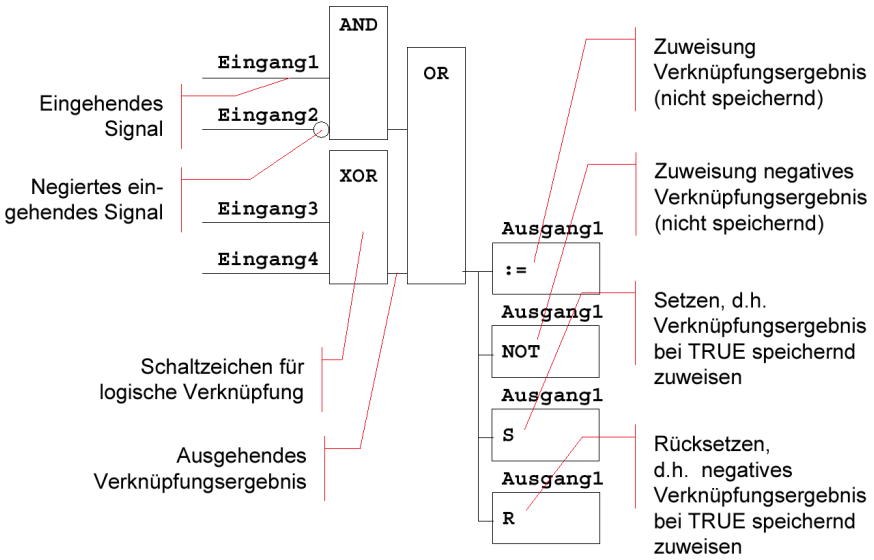


Figure 13: FBD Grundelemente

6.1.3 Instruction List (IL) — Anweisungsliste (AWL)

Am weitesten verbreitet. IL bietet den Vollen Funktionsumfang der SPS. Vergleichbar mit komfortablem Assembler. Befehler werden der Reihe nach abgearbeitet.

Vorteile:

- Am meisten verbreitet

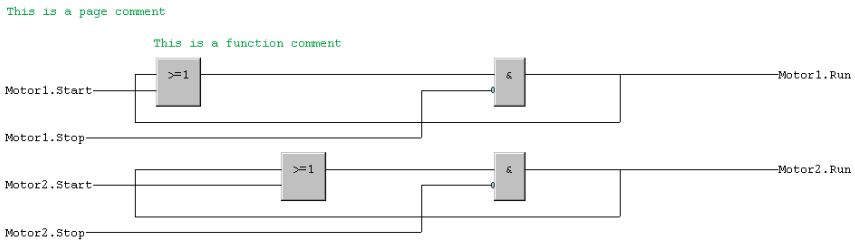


Figure 14: FBD Beispiel Motorsteuerung

- Für alle Steuerungssysteme Verfügbar
- Sehr hohe felxibilität
- Hohe Effizienz durch Maschinen- und Steuerungsorientierung
- Vollständiger Befehlsumfang der SPS nutzbar
- Kann andere grafische Sprachen abbilden

Nachteile:

- Unübersichtlich und fehleranfällig (begründet durch Flexibilität)
- Nicht als alleinige Dokumentation geeignet
- Sprachmittel zur strukturierten Programmierung fehlen
- Keine Datenstrukturen verfügbar

6.1.4 Structured Text (ST)

Kann mit höheren Programmiersprachen wie PASCAL oder C verglichen werden. Die mächtigen Sprachkonstrukte eignen sich besonders für umfangreiche und komplexe Berechnungen und Ablaufsteuerungen. Dazu gehören Konstrukte wie: IF, SWITCH, WHILE, FOR, REPEAT. Auserdem bringt der Compiler einige Standardfunktionen wie: ABS, SQRT, LOG, EXP, SIN, COS, ... mit sich. Es gibt ein Deklarationsteil für die Variablendeklarationen und einen Anweisungsteil für den eigentlichen Code.

Vorteile gegenüber IL:

- Kompakte und gut verständliche Formulierung des Programms

- Gut Strukturierte Anweisungsblöcke
- Gute Möglichkeit zur Steuerung von Programmflüssen

Nachteile gegenüber IL:

- Nicht so performant wie IL (langsamer)
- Benötigt mehr Speicherplatz

6.1.5 Sequence Function Chart (SFC) — Ablaufsprache (AS)

SFC gibt es sowohl in einer grafischen als auch in einer textuellen Variante. Allerdings ist nur die textuelle Variante genormt. Die grafische ist Herstellerabhängig. Besonders geeignet für Sequenzielle Ablaufsteuerungen (zB. Waschmaschine). Es gibt **Schritte** und **Aktionen**. Jeder schritt kann mehrere Aktionen enthalten. Sobald der Schritt aktiv ist, werden die Aktionen der Reihe nach ausgeführt. Einzelne Funktionsbausteine können in LL; FBD, IL, ST oder SFC selbst programmiert werden. **Bei simultanen Abläufen muss darauf geachtet werden, dass sich das Netz nicht verklemmen kann !**

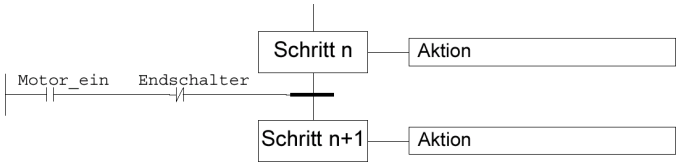


Figure 15: Beispiel SFC

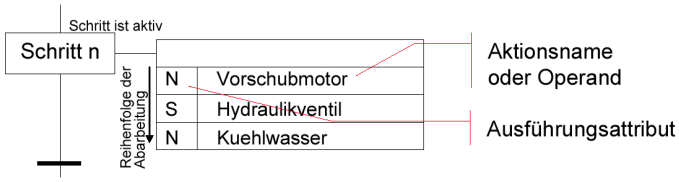


Figure 16: SFC Schritt mit Aktionen

6.2 Programmorganisationseinheiten (POE)

- kleinste unabhängige Software-Einheiten in einem Projekt

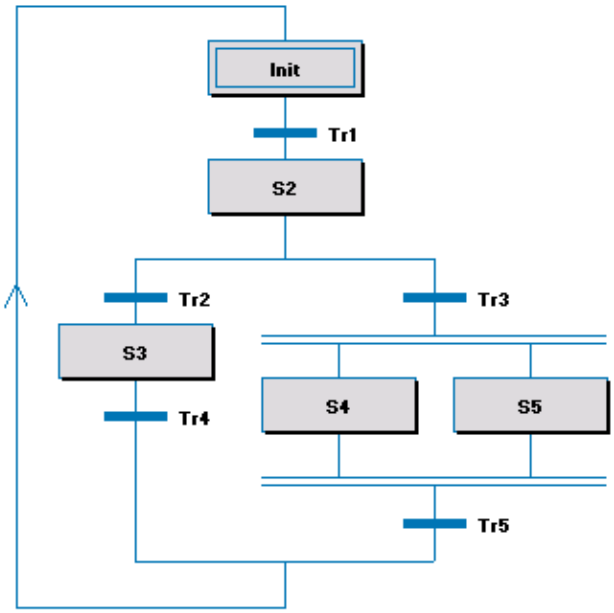


Figure 17: SFC Schrittkette

- dienen der Strukturierung
- können einzlen compiliert und später gelinkt werden
- müssen im Projekt mit einzigartigen Namen bezeichnet werden
- ————-
- Hauptprogramm
- Funktion (ausgelagerter Code, keine statischen Variablen)
- Funktionsbaustein (ausgelagerter Code, statische Variablen - anlegen von Instanzen - ähnlich OOP)

6.3 Softwarekonfiguration

Konfigurationen bilden Konzepte wie Multitasking und verteiltes Rechnen praxisgerecht auf die Steuerungstechnik ab. Dabei stehen die übersichtliche Strukturierung, die Partabilität durch eine einheitliche Softwarearchitektur

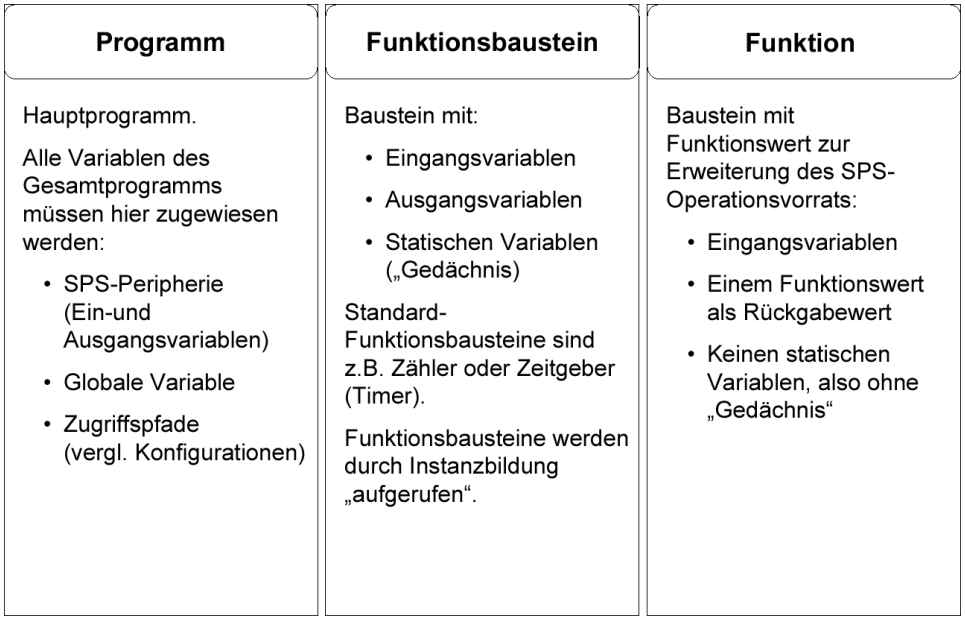


Figure 18: POE Programm - Funktionsbaustein - Funktion

sowei die Verbesserung der Dokumentation und Fehlerdiagnose durch die Modularisierung im Vordergrund.

Es sollen die Laufzeiteigenschaften von Programmen und Funktionsbausteinen (Task CPU resources) festgelegt und die Variablen auf die Adressen der SPS-Hardware gelinkt werden.

7 OPC - Open Process Communication / Object Link and Embedding for Process Control

- OPC = Object Linking and Embedding (OLE) for Process Control
- Basiert auf COM/DCOM von Microsoft
- offene Schnittstellenspezifikation
- COM = Component Object Model (IPC mit RPC)
- DCOM = COM-Erweiterung für Netzwerkkommunikation

- Will Kompatibilität von unterschiedlichen Geräten zueinander schaffen
- Geräte bringen bereits eine OPC-Schnittstelle mit oder können mit Adapter nachgerüstet werden

7.1 Interfacetypen

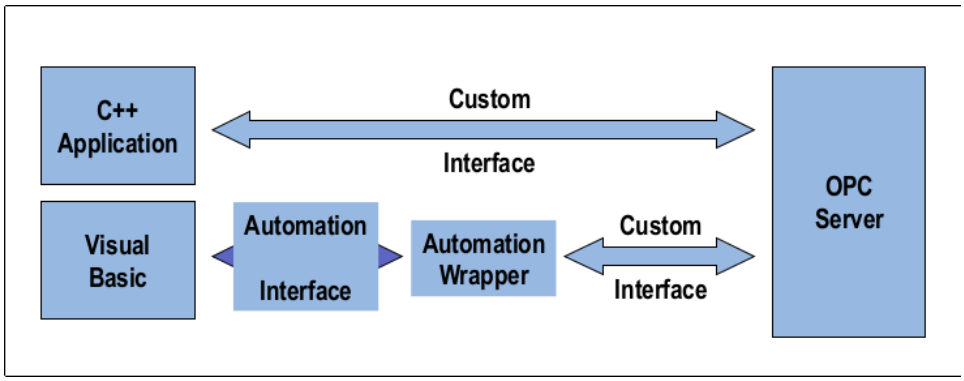


Figure 19: OPC Interface Typen

7.1.1 Custom Interface

Funktionsaufrufe per Funktionszeiger in C/C++.

7.1.2 Automation Interface

Für Sprachen die keine Funktionszeiger unterstützen, Standardschnittstelle, Aufruf über Funktionsnamen.

7.2 OPC Data Access

Schnittstelle zwischen Client- und Serverprogrammen zur Prozessdatenkommunikation. Data Access Server stellt verschiedene Datenquellen bereit. Data Access Client greift über den Server auf diese Daten zu. Es besteht die Möglichkeit normale Lese- und Schreibanfragen zu senden oder über Callback-Mechanismen bei Änderungen informiert zu werden.

Der OPC-Server regelt den direkten Datenverkehr mit dem OPC-Client. Er

erstellt OPC-Group-Objecte über die der Clientzugriff auf die der Group zugeordneten Items erfolgt. Über das Groupobject können Subscriptionparameter festgelegt werden. Die Werte an sich liegen in Form der OPC-Items vor. (OPC-Group-Objects entsprechen DBUS-Interfaces und die OPC-Items den DBUS-Properties.) Groups können entweder 'private' (nur für den Client) oder 'public' für alle Clients angelegt werden.

7.2.1 Properties

Properties können sowohl Knoten als auch Blättern des Namensraums zugeordnet werden. Durch anlegen eines OPC-Items für die Propertie kann diese Beschrieben werden.

7.3 OPC Alarms and Events

Schnittstelle zum Strukturieren, Übertragen und Quittieren von Ereignissen und Alarmen. Im Unterschied zum OPC-DA werden keine Werte sondern Informationen über Ereignisse verschickt. Es können verschiedene Events angelegt werden, bei deren Eintreten Informiert wird. Die Events können an einen OPC-DA Item verknüpft werden. Einordnung der Events in Event-Areas (Raum, Ereignisquellen). Einordnung der Events in einen Filterspace (zB. Hierarchiestufen entsprechen Eventtypen). Zuordnung von Prioritäten.

7.4 OPC Historical Data Access

Zugriff auf historische Daten, entweder Rohdaten oder bereits komprimierte Daten (zB. nur Max-, Min-, Mittelwerte). Der Zugriff geschieht ohne Group oder Item Objecte. Der Client adressiert direkt auf die Daten.

7.5 Nachteile von COM/DCOM

- Konfigurationsprobleme
- Proprietär von Microsoft -¿ auch keine Sicherheit
- Bindung an MS Windows Betriebssystem
- Entwicklern sind Bugs in der DCOM-Schnittstelle ausgeliefert

7.6 OPC UA (inified architecture)

Neueste OPC Spezifikation die einen eigenen Kommunikationsstack mitbringt, der die Bindung an COM/DCOM ablöst. Dieser ist frei verfügbar, portierbar, beherrscht timeouts, multithreading und aktuelle Sicherheitstechniken.

7.7 OPC und XML

XML bietet sich an um OPC in Web-Anwendungen (Internet of Things) einzubinden. Das eingesetzte Protokoll ist zum Beispiel SOAP (Simple Object Access Protocoll).

8 Datenbanken

8.1 Datenmodelle

8.1.1 Hierarchisches Datenmodell

- Datenbeschreibung durch Bäume
- Knoten: Datensätze
- Kanten: Beziehungen zueinander

8.1.2 Netzwerkmodell

- Datenbeschreibung durch Graphen
- Vernetzung der Daten untereinander

8.2 Relationale Datenbanksysteme

- Datendarstellung in Form von Tabellen
- Deutliche Trennung zwischen logischem und physischem Datenmodell -¿ Unabhängigkeit

8.3 Dateisysteme (Windows Explorer)

Vorteile eines Dateisystems

- Zugriffssoftware spezifisch zugeschnitten

- Komportabel für geringe Datenmengen
- Schnelle Durchführung einfacher Abfragen

Nachteile eines Dateisysetms

- Hohes Maß an Redundanz
- Gefahr Inkonsistenz
- Unflexibel
- Probleme im Bereich der Datensicherheit
- Kaum einheitlicher Standard durchsetzbar
- Mehrbenutzerbetrieb je nach System nicht möglich

8.4 Aufbau eines Datenbanksystems (DBS)

- Daten
- Datenbank-Software (MySQL-Server)
- Rechner-Hardware
- Anwendungssoftware

8.4.1 Database Management System (DBMS)

- Mittelsmann zwischen Anwender und Daten
- Nutzt Dienste des Betriebssystems zur Verwaltung
- Überprüfung der Zugriffsberechtigungen
- Recovery Funktionen
- Optimierungsmechanismen
- 3 Datenbestände
  - Datenbank (DB)
  - Logbuch (Protokollierung der Änderungen): notwendig für Recovery
  - Datenlexikon (Struktur der Daten): für Kontroll-Funktionen

8.4.2 Drei-Ebenen-Modell

Externe Ebene

- Definition der Benutzersicht
- Datenmanipulationssprache (SQL)
- Kommandozeileneingabe
- GUI-Schnittstelle

Konzeptionelle Ebene

- konzeptioneller Aufbau und Definition des Informationsgehlats der DB
- Verwendung der Datendefinitionssprache (Data Definition Language DDL)
- Ausführliche Beschreibung der Datenobjekte
- Erläuterung der Beziehungen zwischen den Datenobjekten
- Definition Integritätsbedingungen
- Zugriffsrechte

Interne Ebene

- Art und Aufbau der Datenstruktur
- Regelung des Datenzugriffs auf Speichermedien
- Interaktion mit dem Betriebssystem
- Definition der internen Datenbankstrukturen über (Data Storage Definition Language, DSDL)

Eine Datenbankanfrage kommt durch die Externe Ebene herein und wird nach einer Kontrolle an die Konzeptionelle Ebene weitergeleitet. Dort wird der Befehl verarbeitet und an die interne Ebene durchgereicht. Hier werden die physikalischen Datensätze bestimmt und geladen. Danach gehen diese wieder an die externe Ebene wo sie dem Benutzer angezeigt werden.

8.5 Ziele eines Datenbankeinsatzes

- Datenunabhängigkeit
- Datenintegrität
- Vermeidung von Redundanz
- Sicherheit
- Effizienz
- Mehrbenutzerbetrieb

8.6 Datenintegrität

8.6.1 Arten von Integrität

- Operationale Integrität (synchronisation bei gleichzeitigem Zugriff)
- Datenschutz (permission control)
- Semantische Integrität (falsche Eingaben)

8.6.2 Reaktion bei Integritätsverletzung

- Markieren
- Melden (normal mit Markieren)
- Behandlung (Verweigerung der Änderung, evtl. rückgangig machen)
- Unterbrechung
- Korrektur

8.7 Datenbankfehler

8.7.1 Transaktionsfehler

Transaktion erreicht nicht das Ende

8.7.2 Systemfehler

DBS wird selbst funktionsunfähig, Arbeitsspeicherinhalte gehen verloren. Bei Absturz des Betriebssystems oder Hardwarefehlern.

8.7.3 Speicherfehler

Fehler die zum Verlust von Daten auf dem Massenspeicher führen.

8.8 Entity-Relationship-Modell

Das Entity-Relationship-Modell kann im Entity-Relationship-Diagramm (ER-Diagramm) grafisch dargestellt werden. Hierbei sind Entities Rechtecke, Attribute Ellipsen und Relationships werden als Raute dargestellt.

8.8.1 Entities

Wohlunterscheidbare DInge der realen Welt (Personen, Autos, Bücher, ...)

8.8.2 Attribute

Eigenschaften der Entites. Ein Attribut kann 'einwertig' (Geburtsdatum), 'mehrwertig' (Hobbies) oder 'zusammengesetzt' (Name := Vorname, Nachname) sein.

8.8.3 Werte

Konkrete Ausprägung eines Attributes

8.8.4 Domäne

Wertebereich = Menge aller möglichen oder zugelassenen Werte für ein Attribut Zum Beispiel: Haustier kann nur den Typen Katze, Hund, Maus annehmen.

8.8.5 Entity-Set

Zusammenfassung von zusammengehörenden Entities (alle Mitarbeiter einer Firma)

8.8.6 Schlüssel / Key

Kombination von Attributen, die ein spezielle Entity identifizieren.

8.8.7 Schlüsselkandidat

Entfernung aller überflüssigen Attribute aus einem Schlüssel. Es kann mehrere Schlüsselkandidaten geben.

8.8.8 Primärschlüssel

Ausgewählter Schlüsselkandidat. Andere Schlüssel werden Sekundärschlüssel genannt.

8.8.9 Relationships

Verknüpfung zweier Entites durch eine Beziehung.
zB. —Mitarbeiter— (arbeiten) in —Abteilung—.
Es existieren 1:1, 1:n, n:1 oder n:m Beziehungen. n:m Beziehungen sollten nach möglichkeit vermieden werden, das diese sehr schnell sehr komplex werden.

8.9 SQL - Structured Query Language

8.9.1 SQL - DDL - Data Definition Language

- CREATE TABLE
- DROP TABLE
- ALTER TABLE
- ...

8.9.2 SQL - DML - Data Manipulation Language

- Update
- Delete
- Insert

8.10 Objectorientierte Datenbanken

Es existiert der ODMG Standard (Object Database Management Group). Zur Modellierung und Maniuplation werden analog zu relationalen Datenbanken ODL (Object Definition Language) und OQL (Object Query Language) definiert. Im Gegensatz zu Relationalen Datenbanken werden hier Objekte mit Methoden gespeichert. Die Attribute der Objecte können über entsprechenden Methoden manipuliert werden. Vorteil wäre eine deutlich performantere und einfachere Anbindung an Objectorientierte Programmiersprachen.

9 XML - eXtended Markup Language

9.1 Unterschied HTML - XML

HTML:
HTML hält sich an den XML Syntax und beschreibt die Darstellung Feste Bezeichner und Semantik.

XML:
Wird zum Datenaustausch genutzt und beschreibt den Inhalt. XML kann als Datenmodell für semistrukturierte Daten in From von Bäumen genutzt werden. Freie Bezeichner zur Beschreibung anwendungsspezifischer Syntax.

9.2 XML-Element

Beschreibt ein Object, enthält Text und/oder weitere Elemente. Klammerung durch Start- und Endtag. Es gibt auch leere Elemente.

```
<author>
  <firstname>Peter</firstname>
  <lastname>Mueller</lastname>
</author>
```

9.3 XML-Attribut

Name-Zeichenwert-Paar, assoziiert mit einem Element.

```
<author lastname="mueller" firstname="peter"/>
```

9.4 Wohlgeformtes XML

- Alle Elemente sind korrekt mit Start- und Endtags geklammert
- Genau ein Wurzelelement
- Dürfen immer noch unstrukturierten Freitext enthalten

9.5 Gültiges XML

- Das Dokument ist wohlgeformt
- Ist zu assoziiertem Schema uneingeschränkt konform
- Gültigkeit kann mittels eines Schemas validiert werden

9.6 DTD - Document Type Definitions

- Einfache Grammatik für XML-Dokumente
- Deklaration von Elementen und Attributen
- Beschränkt die beliebige Verschachtelung von Elementen und Attributen
- Teil des XML-Standards

Es wird festgelegt, was das Wurzelelement ist, welche Elemente es gibt, welche und wie viele Attribute/Unterelemente diese besitzen können/müssen.
Schwächen von DTD:

- Ungewolltes festlegen der Reihenfolge
- Kann teilweise zu vage werden
- Alle Elemente sind global in einem Namenraum

9.7 XML-Schema

- komplexe Datendefinitionssprache
- Abwärtskompatibel zu DTD
- Viele Basisdatentypen
- Klassenhierarchien / Vererbung
- Konsitenzbedingungen
- Benutzt selbst XML-Syntax zur Schemadefinition
- — (* nicht so im DTD enthalten)
- Typ* (DTD nur atomar) (XML-Schema: Basisdatentype zB. byte, short, time, date, ...)
- Kardinalitäten (minOccurs, maxOccurs, ...)(*)
- Wertvorgaben (default, fixed)*
- Existenz (optional, required)

9.7.1 Datentypen

Atomare Typen

- byte
- short
- time
- ...

Einfache Typen
Von bestehenden Typen können weitere einfache Typen abgeleitet werden:

```
<simpleType name="humanAge" base="unsignedShort">
<maxInclusive value="200" /> </simpleType>
```

Diese einfachen Typen können keine Verschachtelten Elemente enthalten.
Komplexe Typen Dürfen zusätzlich eingebettete Elemente und Attribute besitzen.

```
<complexType name="authorType">
  <sequence>
    <element name="firstname" type="string" minOccurs="0" maxOccurs="unbou
    <element name="lastname" type="string" />
  </sequence>
  <attribute name="age" type="string" use="optional" />
</complexType>
```

Datentypen können entweder durch Erweiterung oder aber auch durch Restriktion aus anderen hervorgehen (Beispiel eingrenzung der Kardinalitäten). Alle Typen bilde eine Typhierarchie. Elemente eines bestimmten Typs akzeptieren auch Daten einer Erweiterung oder Restriktion des geforderten Typs.

9.7.2 Konsistenzbedingungen

Es ist möglich bestimmte Attribute als einzigartig zu definieren (primary key). Außerdem können bestimmte Pattern hinterlegt werden um zB. zu überprüfen, ob eine eingegebene Email-Adresse syntaktisch korrekt ist.

9.8 XSL - Extensible Stylesheet Language

XSLT = XSL + xPath + formatting objects.
Mit XSLT können Regeln erstellt werden wie gewisse XML elemente dargestellt werden sollen. Damit kann ein XML-Dokument beispielsweise in ein HTML format umgewandelt werden, der dann bequem im Browser angezeigt werden kann.

```
<xsl:template match="address">
  <p>
    <i><xsl:value-of select="name"/></i><br/>
    <xsl:value-of select="street"/><br/>
    <b><xsl:value-of select="town"/></b>
  </p>
</xsl:template>
```

9.9 xPath and XQL

Mit xPath und XQL können Teile aus XML Dokumenten selektiert werden

9.9.1 XQL Expressions

- **"/"** document root
- **"/addresses/address/name"** Absolutes Element name
- **"book//name"** Elemente 'name' irgendwo unterhalb von 'book'
- **"//name"** Elemente 'name' irgendwo im document
- **"address/@type='email'"**
  Attribute mit Name 'type' unterhalb von 'address'
- **"address/@type='email'"**
  Attribute mit Name 'type' unterhalb von 'address' dessen Wert "email" ist
- **"address[name]"** 'address' Elemente mit direktem Unterelement 'name'

9.10 XML-QL — XML-Query-Language

Anfragesprache für XML ähnlich zu SQL. Für den Umgang mit großen Datenbeständen besser geeignet als XQL. Beeinhaltet regüläre Pfadausdrücke

und Muster. XQL eher aus Dokumentverarbeitungssicht sinnvoll. XML-QL erlaubt gleich wie SQL auch komplexere konstrukte wie zB. INNER JOINS.

```
where <book language="french">
            <publisher>
                  <name> Morgan Kaufmann </name>
            </publisher>
            <author> $a </author>
        </book>
in "www.a.b.c/bib.xml"
construct $a
```

Figure 20: XML-QL example

10 Modellierung

Bei der Abbildung von Teilen der realen Welt auf Datenbankmodelle gibt es imemr mehrere Sichtweisen, von denen aus die Zusammenhänge betrachtet werden können.

10.1 UML - Unified Modelling Language

Bei der Modellierung des Projektes ergänzen sich die Diagramme nach und nach. Erst bei der Modellierung aus einer Sichtweise kommen neue Aspekte auf, die im anderen Diagramm zu ergänzen sind.

- Spezifikation, Visualisierung, Konstruktion und Dokumentation von Software
- Standard für Objectorientierte Softwareentwicklung
- Einheitliche Notation
- Offen für neue Konzepte
- Bietet verschiedene Abstraktionsstufen

- Darstellung in verschiedenen Varianten / Sichtweisen

Sichtweise	Diagrammtyp	Aufgaben
Anforderungen	1. Anwendungsfalldiagramm	• Benutzersicht darstellen • Systemsicht zur Umwelt definieren • Überblick über die Funktionalität des Systems geben
Struktur	2. Klassendiagramm 3. Objektdiagramm	• Systemstruktur darstellen
Dynamik	4. Zustandsdiagramm 5. Aktivitätsdiagramm 6. Sequenzdiagramm 7. Kollaborationsdiagramm	• Systemdynamik darstellen • Beziehungen zwischen Objekten und Aktivitäten definieren
Implementierung	8. Komponentendiagramm 9. Verteilungsdiagramm	• physikalische Architektur des Zielsystems definieren

Figure 21: UML-Diagramme

10.1.1 Klassendiagramm

- Klassenname
- Attribute (statisch: Klassenattribute)
- Methoden (statisch: Klassenoperationen)
- Datenkapselung: private, protected, public
- Kompositionen (Widerstand setzt sich zusammen aus Bauform und Material)
- Vererbung

10.1.2 Objektdiagramm

Stellt eine konkrete Instanzierung eines Objektes mit dem aktuellen Zustand dar.

- Assoziationen zwischen Objekten (ist-Teil-von)

Figure 22: UML-Klassendiagramm

Figure 23: UML-Objektdiagramm

- die Assoziationen können Kardinalitäten enthalten
  - 1: genau eine Beziehung
  - *: null eine oder mehrere
  - ...

10.1.3 Zustandsdiagramm

Beschreibt dynamisches Verhalten von Komponenten (Objekte und Operationen) Konzepte entsprechen den hierarchischen Automaten

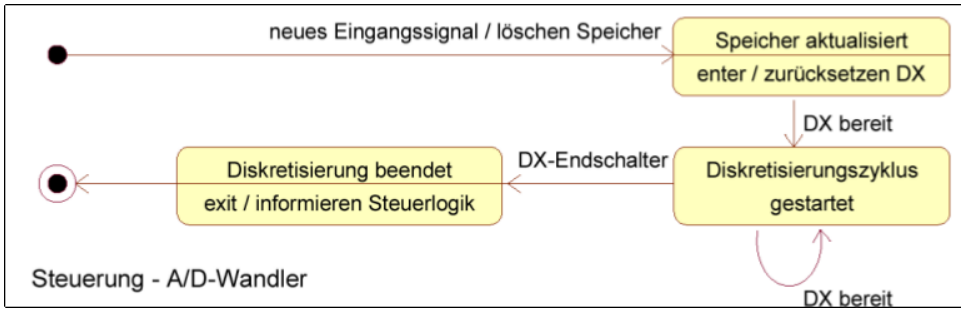


Figure 24: UML-Zustandsdiagramm

10.1.4 Anwendungsfalldiagramm

Zeigt die externen Schnittstellen eines Systems als Teil der externen Kommunkation und seine Hauptaufgaben. Sie dienen der Überprüfung des Modells und sind Grundlage für funktionale Tests.

10.2 SADT - Structured Analysis and Design Technique

- Das System wird durch Daten und Aktivitäten charakterisiert
- Aktivitätensichtweise
- Datensichtweise (beide sind komplementär und ergänzen sich)
- das System kann in verschiedenen Detailstufen dargestellt werden (Blackboxen aufdröseln)
- Wechselseitige Überprüfung auf Vollständigkeit und Konsistenz (Daten und Aktivitätensichtweise)

10.2.1 Erstellen einer Anforderungsdefinition

1. Aktivitätensichtweise mit Top-Down
2. Review
3. Datensichtweise mit Top-Down
4. Vergleich von Aktivitäten- und Datensichtweise
5. Modifikation falls nötig

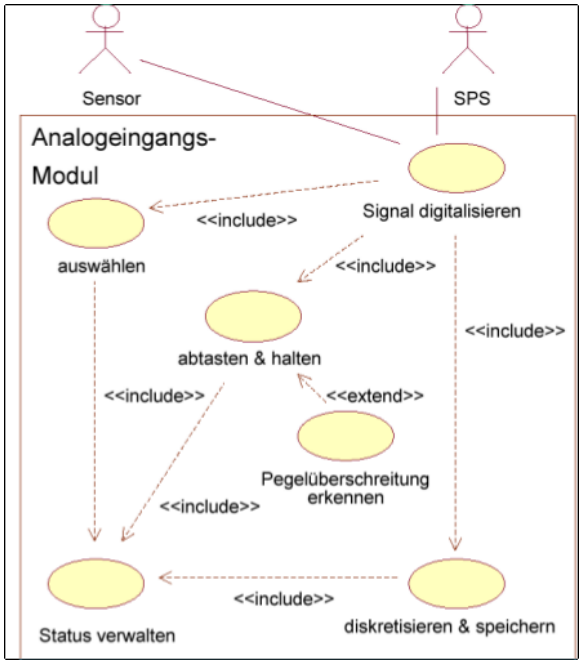


Figure 25: UML-Anwendungsfalldiagramm

10.2.2 Vorteile SADT

- leicht erlernbar, intuitiv
- manuell einsetzbar
- Top-Down zerlegung für klare Problemabgrenzung

10.2.3 Nachteile SADT

- Semantische Ungenauigkeit wegen natürlicher Sprache
- keine formale Prüfung möglich
- keine direkte methodische Anbindung an spätere Phasen
- Erstellen ohne Rechnerunterstützung aufwendig

10.3 Beziehung zu DIN EN 61131 Sprachen

Je nach Darstellungsweise eigenen sich bestimmte Sprachen aus der DIN EN 61131 Norm besser als andere um das Modell direkt zu implemen-

tieren. Beispielsweise können die Automaten des **Steuermodells direkt in Ablaufsprache (AS)** codiert werden.

11 Anlagendmodellierung

11.1 Integration

Die richtige Information ist in Echtzeit am richtigen Platz unabhängig von der Quelle verfügbar. Ziel ist die Zusammenführung aller Informationen in einer Platform trotz deren Unterschiedlichkeit.

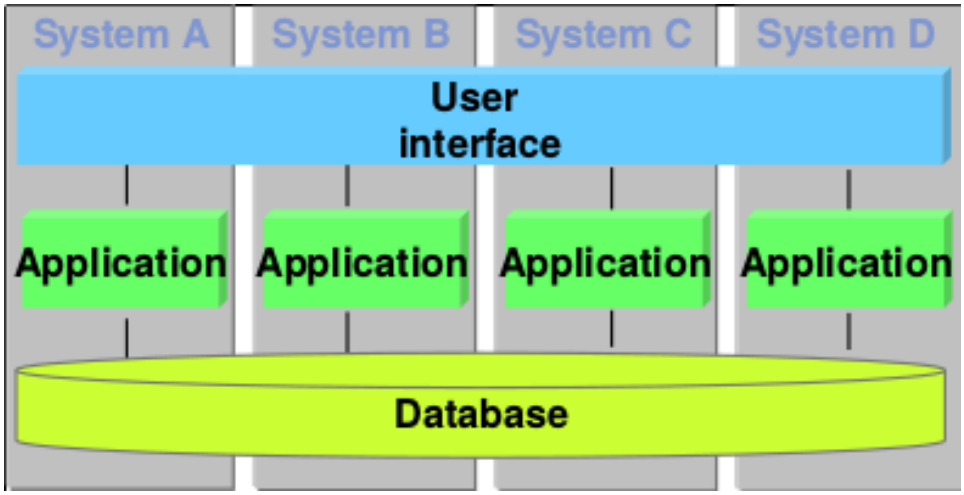


Figure 26: Zentrale Datenbank

- Momentan Trend stark zu Webbasierten Systemen
- Modelliertes Object besitzt viele Verschiedene Aspekte
- Jeweilige Aspekte liegen meist in unterschiedlichen Formaten vor (MS Excel, MS Word, Eagle, ...)
- Ziel: Alle Aspekte über einheitliches Interface verfügbar machen (Aspect Objects^{TM})
- Zusammenhängendes System durch das sich grafisch oder Explorer-View navigiert werden kann
- Datenkonsistenz

- 'Apect Systems' (Control, Process Graphics, Reports, ...) Interagieren durch ein Aspekt-Framework mit der Client-Anwendung
- Objektorientierter Ansatz, Widerverwendbarkeit von Modellkomponenten

11.1.1 Aspect Objects™ - Integration Levels

- **Level 0:**
  Datenblatt öffnen, nur Konfiguration
- **Level 1:**
  Datenblatt an richtiger Stelle öffnen, Parameter möglich
- **Level 2:**
  Systemweit konsistenter Zugriff und Navigation, OPC Zugriff auf andere Aspect Systems
- **Level 3:**
  Engineering Efficiency:
  - Widerverwendbarkeit von Lösungen, Libraries
  - Abstraktion in Objekttypen, einfachere Konfiguration
- **Level 4:**
  Zentrale Systemadministration, Systemupgrades,
  zentrale Prozessbackups, ...
- **Level 5:**
  - Wohldefinierte Fehlerbehandlung
  - Version handling
  - Life Cycle Management

12 Asset Management

12.1 Selbstüberwachende Geräte

- Verwenden Geräteinterne Informationen
- Wartung nach Bedarf möglich
- Nicht alle Fehler werden erkannt

- Testsignal mit bekannter Antwort
- Redundantes Referenzsignal
- Sensor gibt zusätzlich bekannten Referenzwert aus
- Beziehungen zwischen internen Größen werden verglichen
- Erfahrungswissen über Messsignal (ungewöhnliche Änderungen, Außerhalb von möglichem Bereich)

12.2 Übergreifende Überwachung

- Verwendung mehrerer Feldgeräte
- Wird im Leitsystem druchgeführt
- Diagnose komplexer als bei Selbstüberwachung
- Vor-/Erfahrungswissen
- Plausibilitätsbetrachtung
- Math. Modelle: Schätzwert ungleich Messwert
- Empirische Modelle (Vergleich mit gemessener Kurve)
- ergänzend zur Selbstüberwachung

12.3 Asset Management

Asset: Alles was in irgendeiner Form finanziellen Wert besitzt. Umfasst alle Maßnahmen, die dem Ziel dienen mit den Assets unter minimalem Aufwand, einen maximalen Benefit zu erwirtschaften. Dh. maximal auslasten ohne dass etwas dabei kaputt geht. Dabei geht es darum qualitativ gute Informationen zur richtigen Zeit am richtigen Ort zu haben.

- Weniger ungeplante Stillstandszeiten
- Höher Qualität des Endproduktes
- Reduzierung der Betriebs- und Wartungskosten
- Kürzere Umlaufzeiten durch höheren Durchsatz

Eine vom Leitsystem unabhängige Technologie allgemin, herstellerübergreifend verfügbar zu machen ist offenbar zu teuer. Information sollte offen und Herstellerunabhängig verfügbar gemacht werden.

12.4 Overall Equipment Effectiveness (OEE)

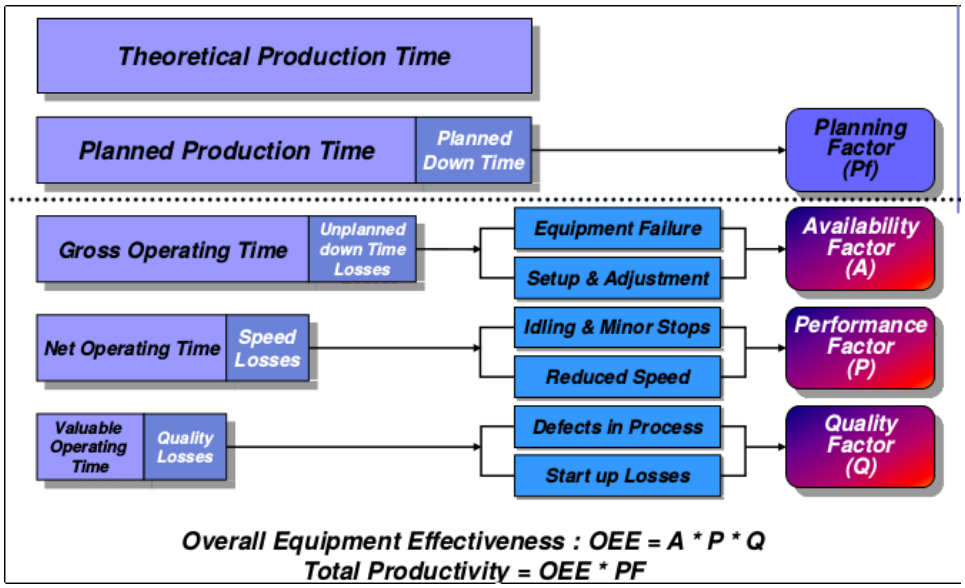


Figure 27: Overall Equipment Effectiveness

12.5 Types Of Device Maintenance (Wartung)

12.5.1 Reactive Maintenance

Nachdem das Geräte kaputt gegangen ist

12.5.2 Preventive Maintenance

Gerät wird nach festem Serviceintervall gewartet. Es besteht die Gefahr unnötiger Wartung.

12.5.3 Predicitve Maintenance

Das Wartungsintervall wird durch Statistiken optimiert. Beispiel: Ölwechsel nach Zeit oder Laufleistung.

12.5.4 Proactive Maintenance

Die eingebauten Diagnosefunktionen zeigen die Notwendigkeit einer Wartung an. Condition Monitoring Vergrößert den Profit und minimiert die Risiken.

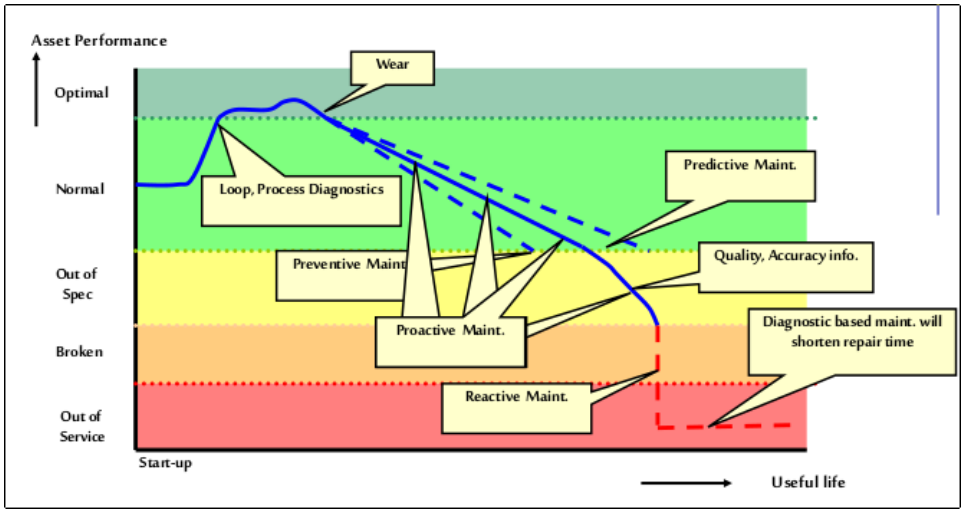


Figure 28: Asset Lifecycle

12.6 Vorgehen Asset Management

12.6.1 Zugang

- Erfassen der zu inventarisiernden Komponenten
- Erfassen vorhandener Tools und Systeme
- Analyse vorhandener Datenstrukturen und Modelle
- Potentialermittlung

12.6.2 Design

- Definition der Schnittstellen zur bestehenden Toollandschaft
- Evaluierung
- Stufenplan für Implementierung
- Kostenbetrachtung
- Toolvorschläge

12.6.3 Integration

- Projektleitung
- Datenmodell
- Schnittstellen
- Toolintegration
- Supportzusicherung für Software und integrierte Lösungen

13 Entwicklungsprocess

Ein System bestehe aus einer Menge von Elementen, welche bestimmte Eigenschaften besitzen und welche durch Relationen miteinander verknüpft sind.

13.1 Entwicklungsstufen

13.1.1 Aufgabendefinition

Der Auftraggeber, die Nutzergruppe beschreiben die Zielvorstellung. Zusammen mit den Entscheidungsträgern den Fachexperten und den Entwicklern wird dann ein gemeinsames Modell des Zielsystems herausgearbeitet. Die Entwickler setzen das Projekt dann um.

13.1.2 Entwicklung

- System wird aus Elementarkomponenten zusammengesetzt
- Abstrakten Wirkprinzipien des Modells werden konkrete Systeme mit diesen Grundfunktionen zugeordnet
- Modellierung kann in verschiedenen Detailstufen vorgenommen werden

13.2 Exploratives Prototyping

Demonstration verschiedener Lösungskonzepte anhand von Prototypen bei unklarer Aufgabenstellung.

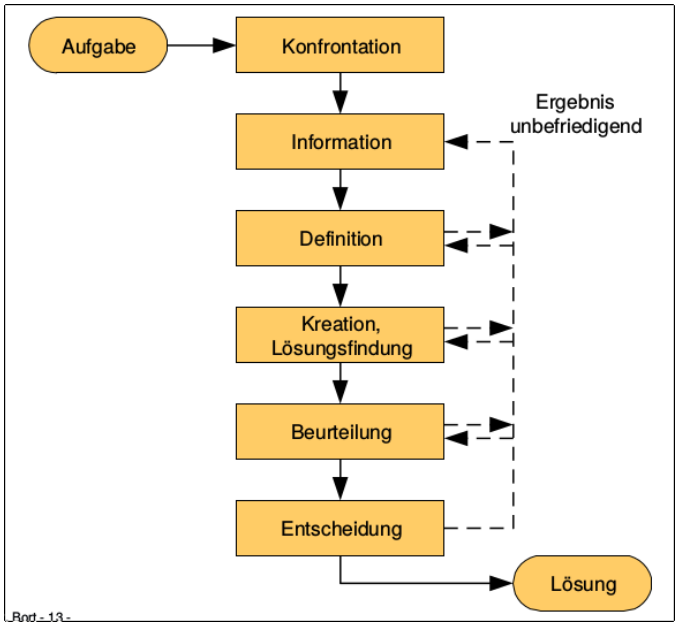


Figure 29: Allgemeiner Lösungsprozess

13.3 Evolutionäres Prototyping

Prototyp wird als "Rohling" des Zielsystems gesehen, der dann sukzessive modifiziert und erweitert wird.

13.4 Phasen eines Automatisierungsprojektes

13.4.1 V-Modell

Auf der linken Seite wird mit einer funktionalen/fachlichen Spezifikation begonnen, die immer tiefer detailliert zu einer technischen Spezifikation und Implementierungsgrundlage ausgebaut wird. In der Spitze erfolgt die Implementierung, die anschließend auf der rechten Seite gegen die entsprechenden Spezifikationen der linken Seite getestet wird. Die Dokumentation sollte während aller Phasen gepflegt werden.

13.4.2 Analyse

- Lastenheft (Ausschreibung, Idee des Auftraggebers)
- Pflichtenheft (Lösung, Grundlage für den Vertrag), Grobstruktur des

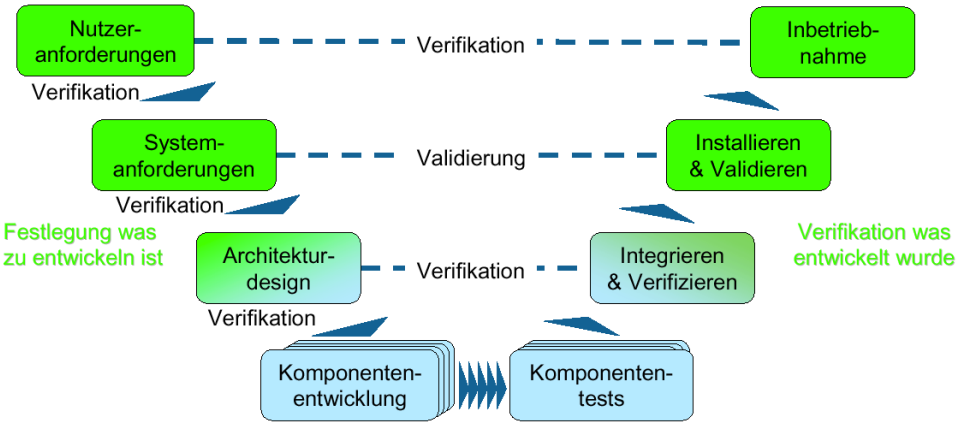


Figure 30: V-Modell

Entwicklers

13.4.3 Entwurf

- Hardware-Entwurf
- Software-Entwurf
- Konzept für Bedienung, Visualisierung, ...

13.4.4 Implementierung, Realisierung

- Geräte beschaffen, Montage, ...
- Programmierung der SPS, ...

13.4.5 Integration und Test

- Testfallentwicklung
- Testdurchführung

13.4.6 Installation, Inbetriebnahme, Abnahme

- Aufbau vor Ort, Test und Abnahme gemäß Plfichtenheft
- Einweisung und Schulung des Bedien- und Wartungspersonals

13.4.7 Wartung und Nutzung

- Fehlerbeseitigung
- Verbesserungen
- Modernisierungen

13.5 Entwurf von Steuerungsprogrammen

- Entwurf der Programmmodule (Ablaufketten der wichtigsten Verknüpfungen)
- Strukturierung des Gesamtprogramms in POEs
- Wiederverwendbarkeit der POEs in weiteren Projekten

13.6 Fehlervermeidung in SPS Software

- Simulation
- Unittests
- Systemtests
- unterschiedliche Entwicklerteams
- redundante Hardware
- redundante Software

Bei der Fehlersuche nach korrektem Betrieb, erst Signalgeber, Verkabelung, etc. untersuchen. Dann Ein-/Ausgabebaugruppen und erst wenn dort nirgends ein Fehler zu finden ist wirklich in der SPS selbst suchen.

13.7 Dokumentation Automatisierungsprojekt

- Geordnete Dokumentation ist Vorraussetzung für ISO Zertifierung
- Änderungsfreundlich halten
- Schnelle Störungsbeseitigung und Reparatur möglich
- –
- Programmstrukturen

- Verdrahtungspläne (Registerzuordnung)
- Belegungspläne der Ein-/Ausgabebaugruppen
- Beschreibung der Signale
- Verkabelungspläne Schaltschrank
- Lastenheft, Pflichtenheft, Änderungsvereinbarungen
- Bedienungsanleitung und Sicherheitsvorschriften

Großer Teil der Dokumentation kann automatisch erzeugt werden (eg. Doxygen). Der meiste Projektaufwand geht in die Planung und die Tests.

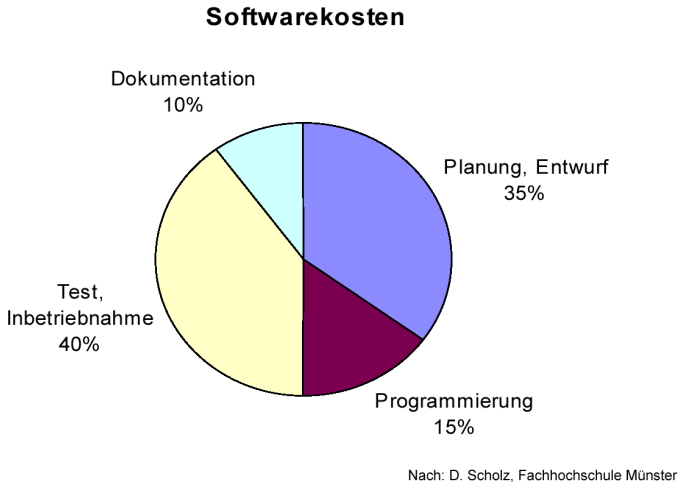


Figure 31: Softwarekosten

14 Enterprise Application Integration (EAI)

Aus eigenständigen isolierten Anwendungen werden integrierte Geschäftsprozesse. Informationen werden immer mehr zu einem entscheidenden Wettbewerbsfaktor. Ziel von EAI ist es barrierefreien Informationsfluss sowohl innerhalb eines Unternehmens, als auch zwischen verschiedenen Unternehmenspartnern zu schaffen. Dazu besteht der Bedarf nach einheitlich strukturierten Produktdaten.
Interne Integration: Application to Application, Daten werden zwischen Anwendungen ausgetauscht

Externe Integration: Business to Business

- Entstanden aus ERP (Enterprise Resource Planning (Software))
- EAI geht über Unternehmensgrenzen hinweg
- Unterstüzt den optimalen Ablauf der Geschäftsprozesse
- EAI ist Kommunikationsdrehscheibe für den Transport von Datenströmen zwischen verschiedenen Anwendungen.

14.1 Middleware

- Middleware transportiert und transformiert Daten (Protokoll, Darstellung)
- API versteckt komplexe Kommunikation vor dem Anwender

14.2 Integration von Middleware

- Software Bus
- Integration der bestehenden Middleware
- Integration der Physikalischen Netze
- Integration der Anwendungen (mittels Connectoren)
- Integration der Administratortools

14.3 Anwendungsintegration

Prinzip: Integration über Referenzmodell (gemeinsames Protokoll) → XML.

14.3.1 Entwurfsmuster

Adapter:
Technologie-Adapter: Anbindung der Anwendung an die Infrastruktur.
Anwendungs-Adapter (Wrapper): Kapselung der Anwendungsschnittstellen.
Bridge:
Trennung von Abstraktion und Implementierung für bessere Portierbarkeit
Facade:
Generalisierte Schnittstelle

14.4 Connectoren

14.4.1 Standardconnectoren

Für Standardanwendungen (SAP, DCOM, ...)

14.4.2 Systemconnectoren

Legacy-Systeme auf Mainframe-Systeme (Emulationsschnittstelle für Terminalkasten)

14.4.3 Eigenentwickelte Connectoren

Oft SDK vom Hersteller angeboten

14.5 Integrationsarchtitekturen

14.5.1 Individuelle Integration

individuell Programmiert, klassische Middleware

14.5.2 Datenbank Integration

Datenbank-Connectoren, gesamter austausch über Datenbanken

14.5.3 EAI-Framework

Entwicklungs- und Runtime Framework zur API Integration des Workflows und der Daten

14.6 Integrationsserver

- Modellierungstool für Geschäftsprozesse
- Softwaregenerator zur Umsetzung von Ablauflogik
- Integrationsfunktionen
  - Datenbank
  - Middleware
- Message Broker (Server-Centric / Network-Centric)
- Anwendungsübergreifende Transaktionen

14.7 Produkinformationen/-merkmale

- Bezeichnung ist eindeutig und wird überall verstanden → ausreichend
- Es gibt unterschiedliche Definitionen für die verschiedenen Produkte → Merkmal + Definition
- Datenaustausch zwischen Betrieben: Hier muss eine genau Definition und Übersetzung vorhanden sein.

14.8 Normierung für einheitliche Produktdaten

- Beteiligung der Normungsgremien + beteiligter Industrie
- Muss sich dem Markt anpassen können, erweiterbar sein (aufwärtskompatibel)
- Alle Industriegebiete abgedeckt
- Beginn mit einzelnen Teilgebieten
- Verwendung eines standardisierten Merkmallexikons (Datenbank mit genormten Merkmalen die eindeutig beschrieben werden)
- Solche Merkmallexikons sind bereits vorhanden (verschiedene Sparten)

15 Projektabwicklung

15.1 Phasen der Projektierung

15.1.1 1. Grundlagenermittlung

- **Projektziele festlegen**
- Grobe Kalkulation

15.1.2 2. Vorplanung

- Anlagenkonzept festlegen
- Einflussgrößen, Aussagen zum Umfeld, Ausbau, ...

15.1.3 3. Basisplanung

- PLT Funktionen festlegen
- Verfahrenstechnische Realisierungen festlegen

15.1.4 4. Ausführungsplanung

- Geräte festlegen
- Leitsystem spezifizieren
- Funktionspläne, etc. erzeugen

15.1.5 5. Errichtung

- Lieferung
- Software konfiguration
- Montage
- Funktionen prüfen

15.1.6 6. Inbetriebsetzung

- Personal ausbilden
- Dokumentation übergeben

15.1.7 7. Projektabschluss

- Abschlussbericht
- Projektabrechnung

15.2 Planungsunterlagen der Prozessleittechnik

Zur Planung der Prozessleittechnik kommen eine vielzahl an verschiedenen Modellen zum Einsatz. Jedes Modell bietet verschiedene Sichtweisen und Datailierungsstufen. (Vorteile siehe Kapitel UML).

15.2.1 Anlagenmodell

15.2.2 Prozessmodell

15.2.3 Phasenmodell

15.2.4 Fließbilder zur Darstellung des Prozesses

Dienen zur Darstellung der Prozessflüße und der PLT-Aufgaben. Rohrleitungs- und Instrumentalisierungsfließbild, Grundfließbild, ...

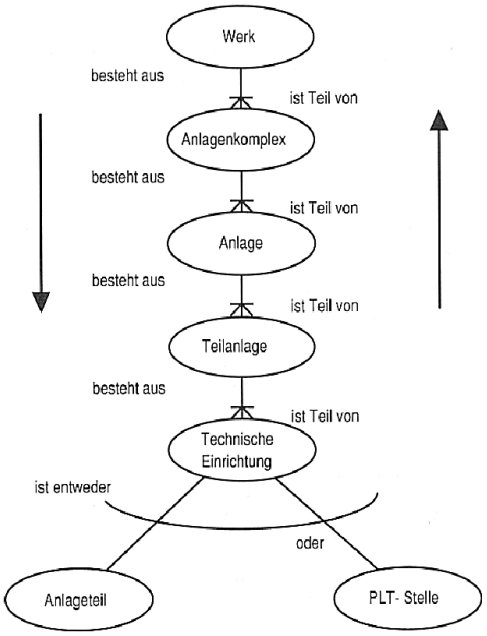


Figure 32: Anlagenmodell

15.2.5 PLT Dokumentation

Enthält die wesentlichen verfahrenstechnischen Betriebsdaten, projektspezifische Festlegungen, Beschreibung der Apparate und Rohrtechnik

15.2.6 Markt für Automatisierungsprodukte

Die meisten Marktanteile bestehen in der Wartung und Reparatur von bestehenden Anlagen. Der Anteil von neuen Anlagen beträgt gerademal 15Das liegt daran, dass der Markt schnell ist und man sich kaum lange Anlaufphasen für Neuanlagen leisten kann.

15.3 Strategien zur Anlagenmigration

- Verwendung von Standards
- Modularer Anlagenaufbau
- Wiederverwendbarkeit von Komponenten
- Konfiguration statt Programmierung

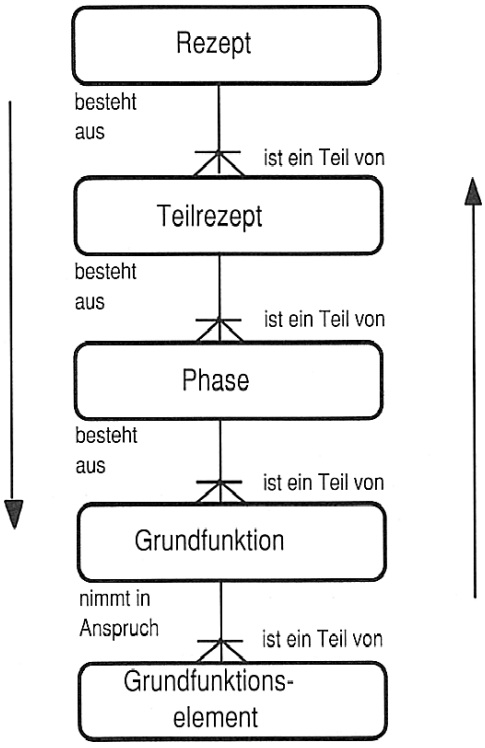


Figure 33: Prozessmodell

- Automatisierung der Anpassung
- 'Online-Update'

15.4 Konzepte für automatisierte Migration

- Erstellen von Regeln um bestimmte Muster zu erkennen und zu transformieren
- Baugruppen werden dann erst Abtrahiert und anschließend in das Zielsystem Deabstrahiert
- Beispiel Neuzuordnung von I/O-Pins für neuen Controller (IO-Transformation)
- Einzelne Teile müssen von Hand mit Hilfe eines Regeleditors bearbeitet werden (sind danach auch in Library)
- Unnötige Logik wird dabei erkannt und wegoptimiert

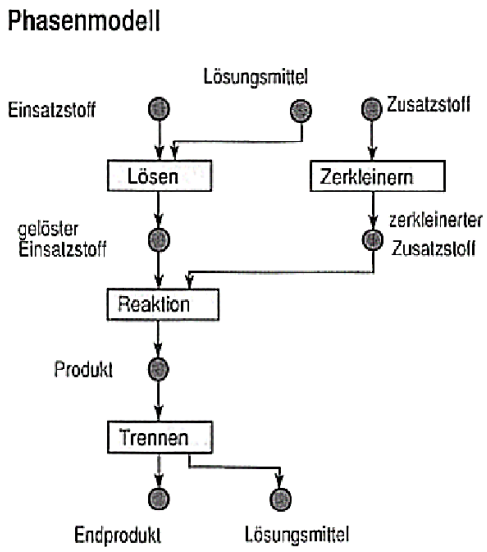


Figure 34: Phasenmodell

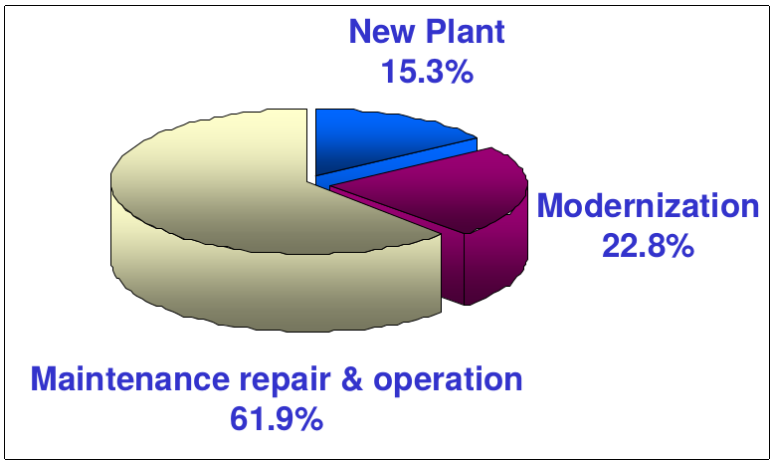


Figure 35: Marktanteile Automatisierungsprodukte

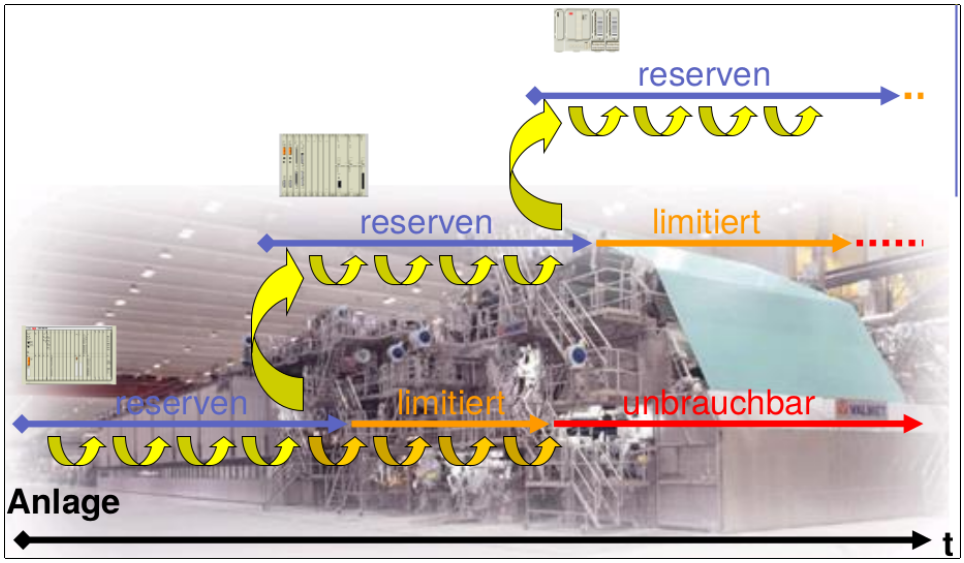


Figure 36: Lebenszyklusbetrachtung

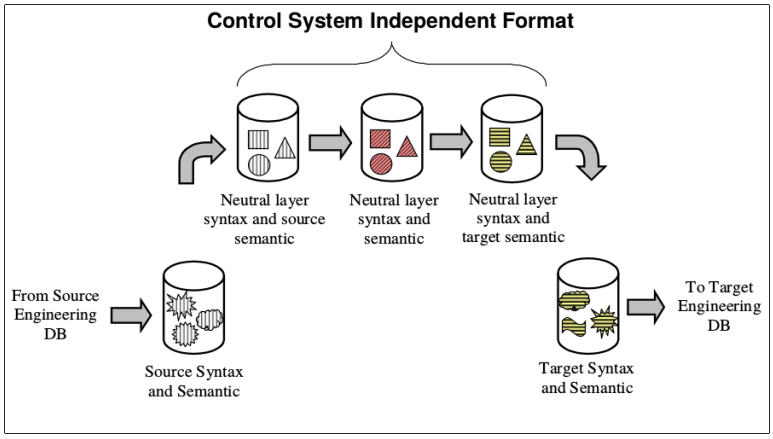


Figure 37: Konzept für automatisierte Übersetzung