Applikationen prozedural konzipieren | SASD

Risiken der Softwareentwicklung

- Termin- und Budgetüberschreitungen
- Nicht-Erfüllen der Anforderungen
- mangelnde Qualität
- Verständigungsprobleme und Konflikte
- Einführungsprobleme (Umstellung, Migration)
- veraltetes System nach zu langer Entwicklungszeit
- Projektabbruch

Risiko-Ursachen

- zunehmende Komplexität der Software
- zunehmende Qualitätsanforderungen
- rasche technologische und geschäftliche Entwicklung
- Softwarequalität schwer messbar
- unklare Auftragssituation
- stets ändernde Benutzeranforderungen
- Fachwissen verstreut oder nicht mehr vorhanden
- fehlende Informatik-Strategie (Insellösungen, heterogene Systeme)
- strukturierte Entwicklung zuwenig verbreitet
- beschränkte Produktivität, viel Handarbeit trotz CASE-Tools (Computer-Aided Software Engineering)

Vorteile methodisches Vorgehen

- Nutzung von gemachten Erfahrungen
- strukturierter, planbarer Entwicklungsablauf: genauere Aufwandschätzung, Termin- und Kostenplanung
- Umsetzung der Anforderungen in vordefinierter Qualität
- klar definierte Ergebnisse
- Minimierung Entwicklungsrisiken
- Entwicklungsprozess wiederholbar, nachvollziehbar und personenunabhängig

Strukturierte Analyse

Eine strukturierte Analyse (structured analysis) ist eine Methodenklasse, die die Basiskonzepte Datenflussdiagramm, Data Dictionary-Einträge, Pseudo-Code und Entscheidungstabellen zu einer Methode zur Definition von Anforderungen kombiniert.

Strukturiertes Design

Ein strukturierter Entwurf (structured design) ist eine Entwurfsmethode, die zu einer Softwarearchitektur führt, die aus funktionalen Modulen besteht. Die Struktur der Architektur ist ein Baum oder ein azyklisches Netz. Die Beschreibung erfolgt durch Strukturdiagramme.

Entwicklungsphasen

Planungsphase (Vorstudie)

Aktivitäten:
- Situationsanalyse und Zielformulierung
- Voruntersuchung des Produkts
- Untersuchung der fachlichen, ökonomischen und personellen Durchführbarkeit

Ergebnisse:
- grobe Anforderungsspezifikation
- Rahmenbedingungen
- Projektkalkulation
- Projektplan
- Entscheid:
  Eigenentwicklung, Standardanwendung, Stopp

Definitionsphase (Analyse)

Erstellung Produktdefinition aus Anwendersicht (Was).

Aktivitäten:
- Anforderungsanalyse (requirements engineering)

Ergebnisse:
- detaillierte Anforderungsspezifikation
- Produktmodell
- Konzept der Benutzerinteraktion

Entwurfsphase (Design)

Umsetzung Produktanforderungen in software-technischen Entwurf (Wie).

Aktivitäten:
- Entwicklung Softwarearchitektur

Ergebnisse:
- Softwarearchitektur
- Spezifikation der Softwarekomponenten
- Dialogentwurf

Realisierungsphase

Aktivitäten:
- Datenstrukturen und Algorithmen konzipieren
- Programme strukturieren
- Implementationsentscheidungen dokumentieren
- entwickelte Programme testen

Ergebnisse:
- Quellprogramme
- Dokumentation
- Objektprogramme
- Testplanung und Testprotokolle

Einführungsphase

Aktivitäten:
- Übergabe an den Auftraggeber
- Rahmenorganisation zum Betrieb
- Integration, Datenübernahme, Schnittstellen
- Installation in die Zielumgebung
- Schulung der Benutzer

Ergebnisse:
- installiertes Produkt
- Gesamtdokumentation
- Abnahmeprotokoll
- Einführungsprotokoll

Definitionsphase

SASD

Systemanalyse

Systematische Vorgehensweise zur Ermittlung der Anforderungen in einem iterativen Prozess.

- Anforderungen ermitteln
- Anforderungen festlegen und beschreiben
- Anforderungen analysieren
- Anforderungen simulieren und ausführen (exploratives Prototyping)
- Anforderungen verabschieden

Resultat ist eine korrekte, vollständige, konsistente und eindeutige Anforderungsspezifikation.

Vorteile Strukturierte Analyse

- vereinfacht die Definition von Anforderungen an komplexe Systeme durch Top-Down-Vorgehen
- leicht erlernbare aufeinander abgestimmte Techniken
- übersichtlich durch hierarchische Gliederung
- für den Auftraggeber nachvollziehbar
- legt zu erbringende Ergebnisse fest
- Qualitätssicherung durch Quervergleiche: syntaktisch: Case-Tools, semantisch: Review

Resultat ist ein essenzielles Modell des Systems.

Essenzielles SA-Modell

Vollständiges, formal konsistentes Modell ohne Implementationsdetails.

- gibt die essenziellen Anforderungen wieder
- beschreibt Funktionalität des Systems
- zeigt notwendige Informationen zur Sicherstellung der Funktionalität auf
- ist auch für den Benutzer verständlich

Komponenten:
- Umgebungsmodell
- Verhaltensmodell

SA-Methode

Hierarchiekonzept (stufenweise Verfeinerung):
- Kontextdiagramm
- Datenflussdiagramm Level 1, Level 2 usw.
- nicht weiter unterteilbare Elementarprozesse

Vorgehensweisen:
- funktionsorientiert: Funktionsbaum FB
- datenorientiert: Datenflussdiagramme
- ereignisorientiert: Ereignistabelle

Qualitätskriterien:
- geringe Komplexität
- Minimalität
- Technische Neutralität

Umgebungsmodell

Legt Randbedingungen und Schnittstellen des Systems fest (Black Box).

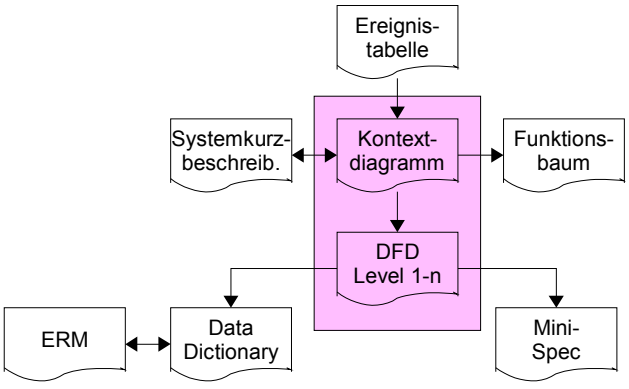
Komponenten:
- Ereignistabelle
- Kontextdiagramm
- Systemkurzbeschreibung

Verhaltensmodell

Beschreibt das Innere des Systems mit allen Details (White Box).

Komponenten:
- Datenflussdiagramme
- Mini-Spezifikationen (Prozessbeschreibungen)
- Data Dictionary (Datenkatalogeinträge)
- Entity Relationship-Modell

Ergebnisse Strukturierte Analyse



Ereignistabelle

Liste der systemaktivierenden externen oder zeitlichen Ereignisse.

Dient zur Ermittlung der externen Anforderungen an das System.

Ereignistypen:
- extern: Anstoss durch externe Agenten
- zeitlich: Erreichen eines Zeitpunkts
- Kontroll-Ereignis: Erfüllung einer Bedingung

Beispiele Ereignistabelle

Beispiel Seminarhotel:

Nr.	Auslöser	Ereignis	Antwort/Reaktion
1	Gast	Reservation	Reserv.bestätigung
2	Gast	Check-Out	Rechnung
3	Auftraggeber	Anfrage	Offerte

Beispiel Auftragsbearbeitung:

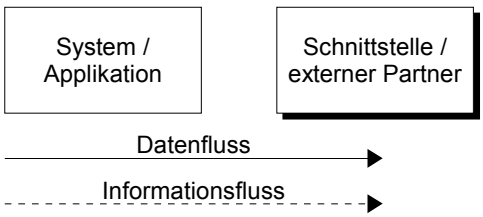
Nr.	Auslöser	Ereignis	Antwort/Reaktion
1	Kunde	Anfrage	Angebot
2	Angebot	Kundenauftrag	Auftragsbestätigung, Fertigstellungsauftrag
3	Fertigungs-auftrag	Fertig-Meldung	Kundenrechnung, Buchungssatz
4	Zeit	Monatsanfang	Artikelpreisliste

Kontextdiagramm

Grafische Modellierung der Ereignisse und Reaktionen.

Datenflussdiagrammm, das die Schnittstellen des zu modellierenden Systems mit seiner Umwelt beschreibt.

Notation:

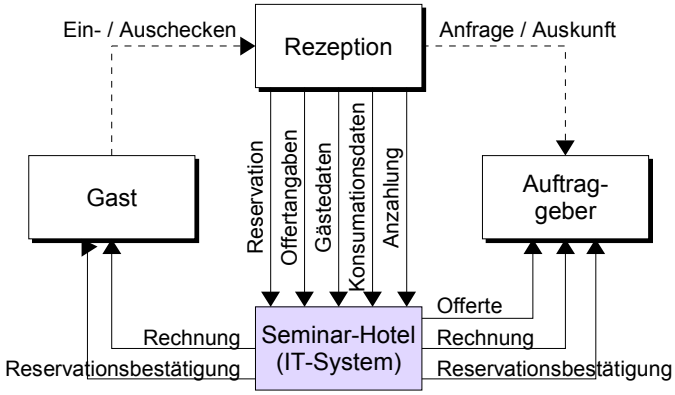


Regeln Kontextdiagramm

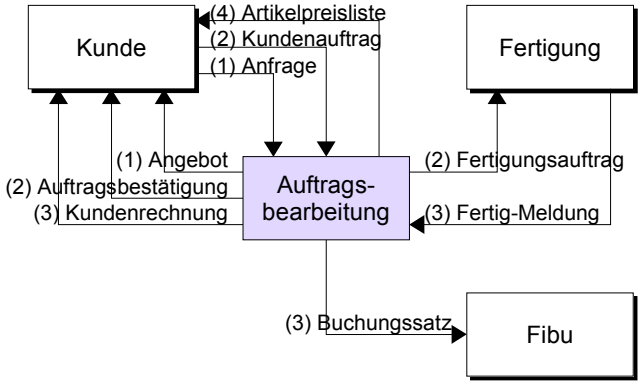
Syntaktische Regeln:
- enthält nur einen Prozess, die Applikation als Ganzes
- enthält mindestens eine Schnittstelle
- keine Datenflüsse zwischen den Schnittstellen (evtl. Informationsflüsse zur Erläuterung)
- keine Speicher und keine Funktionen

Semantische Regeln:
- beschreibt den Anwendungsbereich (problem domain)
- zeigt Datenflüsse, welche die Systemgrenze passieren
- Schnittstellen geben ursprüngliche Quelle oder Senke an

Beispiel 1 Kontextdiagramm



Beispiel 2 Kontextdiagramm



Systemkurzbeschreibung

Kurzbeschreibung in Textform
Ergänzung zum Kontextdiagramm

Beispiel:

Die Applikation "Seminar-Hotel" erlaubt es, Seminar- und Zimmer-Reservationen für Kunden (Gäste und Firmen) zu erfassen, den Check-In / Check-Out der Gäste zu handhaben sowie die Abrechnung der erbrachten Leistungen zu regeln.

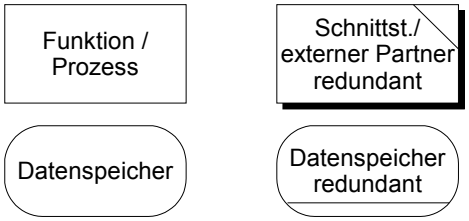
Datenflussdiagramm

SASD

Datenflussdiagramm DFD

Teil des Verhaltensmodells. Beschreibt die Wege von Daten bzw. Informationen zwischen Funktionen, Speichern und Schnittstellen sowie die Transformation der Daten bzw. Informationen durch Funktionen.

Notation (zusätzlich zum Kontextdiagramm):

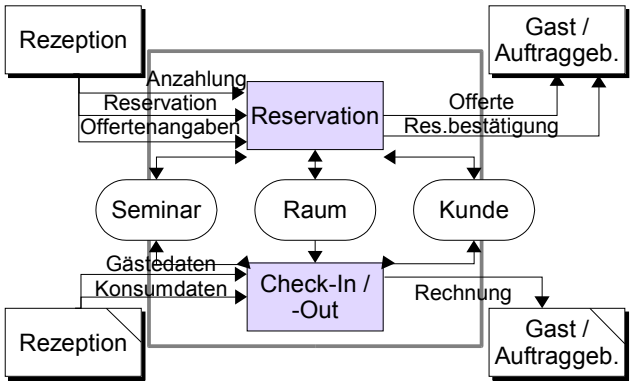


Regeln Datenflussdiagramm

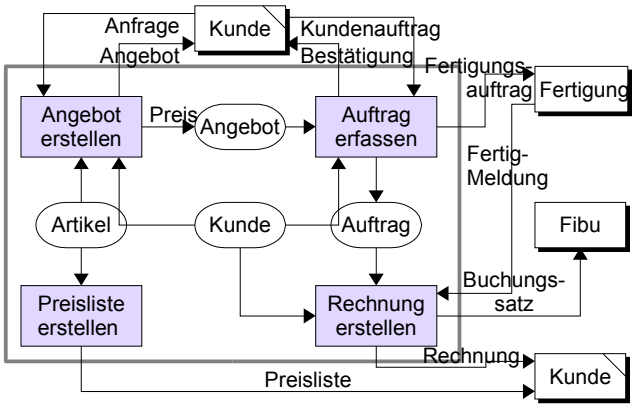
Syntaktische Regeln:
- enthält mindestens eine Schnittstelle
- keine Datenflüsse zwischen Schnittstellen, zwischen Speichern, oder zwischen Schnittstellen und Speichern
- Benennung Datenflüsse: (Adjektiv und) Substantiv (Ausnahme: gesamte Daten von und zu Speichern)
- Benennung Funktionen: Substantiv und Aktions-Verb
- pro Funktion mind. 1 Eingangs- und 1 Ausgangsdatenfluss

Semantische Regeln:
- beschreibt Datenfluss, nicht Kontrollfluss
- Schnittstellen geben ursprüngliche Quelle oder Senke an
- nur wesentliche Funktionen

Beispiel 1 DFD Level 1



Beispiel 2 DFD Level 1



Erstellung Datenflussdiagramm

- Stufenweise Verfeinerung, Level 1, 2 usw.
- Zusammenhänge mittels Data Dictionary
- Schnittstellen und Speicher können nicht verfeinert werden
- Anzahl Prozesse pro Stufe 7 +/-2
- zur Ermittlung der Speicher evtl. zuerst ERM erstellen
- Prüfung mit Handsimulation: Prozesse simulieren

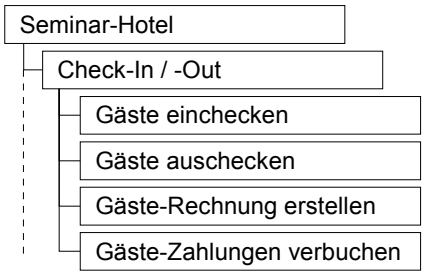
Verhaltensmodell

SASD

Funktionsbaum

Funktionen-Hierarchie-Diagramm FHD, Gliederung einer Funktion in Teilfunktionen (Ergänzung zum DFD).

Beispiel:



Regeln Funktionsbaum

- nur essentielle, d.h. geschäftlich relevante Funktionen
- unabhängige, einzeln ausführbare Funktionen
- Funktion durch Subfunktionen vollständig abgedeckt
- mind. 2 Funktionen pro Ast
- Funktionen können mehrmals vorkommen

Mini-Spezifikation

Inhaltliche Beschreibung eines nicht weiter verfeinerbaren Prozesses (Elementarfunktion).

- Transformation der Eingabedaten in die Ausgabedaten
- keine Implementationsvorschriften
- normaler Text, Pseudocode, Entscheidungstabellen

Beispiel:

Gast einchecken
Für den Gast (= Kunde) wird zum gewünschten Zeitraum geprüft, ob das reservierte Zimmer frei ist. Wenn ja, wird das Zimmer auf den Namen des Gastes gebucht. Wenn nein, wird eine Fehlermeldung aufbereitet. Weiter wird die Anwesenheit des Kunden zum gebuchten Seminar eingetragen.

Data Dictionary DD

Datenkatalog, Verzeichnis der Struktur, Eigenschaften und Verwendung von Daten.

Zweck:
- stufenweise Definition der Datenflüsse und Datenspeicher aus den Datenflussdiagrammen
- Ableiten von Attributen für das Datenmodell (ERM)
- Konsistenzprüfung DFD:
  - durch Case-Tools
  - ermöglicht Handsimulation

Notation Data Dictionary

Symb.	Bedeutung	Beispiel
=	ist äquivalent zu	A=B+C
+	Sequenz (ohne Ordnung)	X=X1+X2+X3
[|]	Auswahl (entweder oder)	A=[B | C]
{ }	Wiederholung	A={B}
M{ }N	Wiederholung von M bis N	A=1{B}10
( )	Option	A=B+(C)
* *	Kommentar	A=X+Y *Komm.*

Beispiel Data Dictionary

Bezeichnung	Inhalt
Kundendatei	= {Kundeneintrag}
Kundeneintrag	= Personal-Nr. + Name + Adresse + (Geburtsdatum) + (Funktion) + Umsatz
Name	= Anrede + (Titel) + Vorname + Nachname
Adresse	= [Strasse + Hause-Nr. | Postfach] + (Länderkennzeichen) + PLZ + Ort + (Telefon) + (Fax)

Entscheidungstabelle

Aufbau Entscheidungstabelle

Stellt komplexe Entscheidungssituationen übersichtlich dar.

Tabellenbezeichnung	Entscheidungsregeln
Feld1: Bedingungsbeschreibung	Feld2: Bedingungsanzeiger
Feld3: Aktionsbeschreibung	Feld4: Aktionsanzeiger

Erstellung Entscheidungstabelle

Begrenzte Entscheidungstabelle:

- Anzahl Regeln: 2 hoch Anzahl Bedingungen
- verdichtet: Regeln mit gleichen Aktionen zusammengefasst (z.B. Regel R5 – R8 in ET1)

Begrenzte Entscheidungstabelle

J = Bedingung muss erfüllt sein / N = darf nicht erfüllt sein
– = Bedingung übt keinen Einfluss aus (Irrelevanzanzeiger)

Eingangsbearbeitung	R1	R2	R3	R4	R5
Lieferfähig	J	J	J	J	N
Angaben vollständig	J	J	N	N	–
Bonität in Ordnung	J	N	J	N	–
Lieferung mit Rechnung	X		X		
Lieferung Nachnahme		X		X	
Angaben vervollständig.			X	X	
Mitteilen: nicht lieferbar					X

Erweiterte Entscheidungstabelle

Bedingungen und Aktionen sind erst mit Bedingungs- bzw. Aktionsanzeiger eindeutig bestimmt (J/N auch erlaubt).

Weihnachtsgeschenk	R1	R2	R3	R4	R5
Alter des Mitarbeiters	<20	20-40	20-40	41-65	41-65
Anzahl Jahre in Firma	–	<10	>=10	<10	>=10
Weihnachtsg.-Typ	I	II	III	IV	V
Nachtessen mit Chef			X		X

Anwendung ET

- Anforderungsanalyse
- Programmentwurf
- Daten-Mapping
- Testfallerstellung

Diagramme

UML:
- Aktivitätsdiagramm
- Zustandsdiagramm

CRUD-Matrix

Stellt den Zusammenhang zwischen der Daten- und Funktionssicht dar.

Anwendung:
- Grundlage für Aufwandschätzung
- Vollständigkeitsprüfung

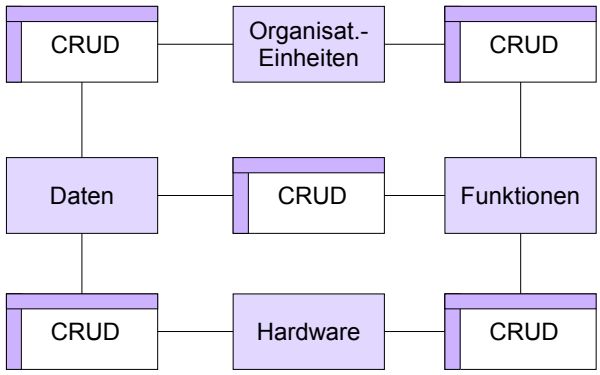
Zugriffsarten:
C = Create
R = Read
U = Update
D = Delete
A = All

Beispiel CRUD-Matrix

Entitätstyp	F1	F2	F3	F4	F5	F6	F7	F8
Raum	A			R	R	R	R	
Infrastruktur	R	A						
Raumbelegung				C	U	U	R	D
Infrastruk-Reserv.				C	U	U	R	D
Seminar			A	R	U	U	R	D
Seminarteilnahme					C	U	R	D

F1: Raumdaten verwalten
F2: Infradaten verwalten
F3:Offerten erstellen
F4 Res. vornehmen
F5: Res. bestätigen
F6: Res. ändern
F7: Res. abfragen
F8: Res. annullieren

Sicht-Kombinationen mit CRUD-Matrix



CASE-Tool

Speichert alle Modell-Informationen in einer Repository-DB (Methodenregeln, Zusammenhänge innerhalb und zwischen Modellen, Data Dictionary).

Vorteile:
- Abstimmung der Modell-Sichten
- automatische Erstellung vieler Dokumente
- Konsistenzprüfungen
- Auswertungen
- Gesamt-Produktdokumentation
- Versionen-Verwaltung, Konfigurationsmanagement

Entwurfsphase

SASD

Systemdesign

Entwurfsprozess:
- Umgebungs- und Randbedingungen ermitteln
- grundlegende Entwurfsentscheidungen treffen
- Benutzergruppen definieren (Nutzungsprofile, Beteiligung)
- Software-Architektur entwerfen
- definiertes System in Software-Komponenten zerlegen
- System durch geeignete Anordnung strukturieren
- Beziehungen zwischen Komponenten beschreiben
- Funktions- und Leistungsumfang sowie Verhalten der Software-Komponenten spezifizieren
- Schnittstellen festlegen

Resultat ist eine Entwurfsspezifikation, bestehend aus Software-Architektur und Spezifikation der SW-Komponenten.

Vorteile Strukturiertes Design

- Erhöhung der Produktivität bei der Entwicklung und Wartung komplexer Software-Systeme
- Qualitätssicherung aus Entwicklersicht, z.B. Wartbarkeit
- Qualitätssicherung aus Benutzersicht, z.B. Zuverlässigkeit

Einflussfaktoren

- Produkteinsatz:
  Mandatenfähigkeit, zentraler / verteilter Betrieb
- Nicht-funktionale Anforderungen:
  Skalierbarkeit, Internationalisierung
- Qualitätsanforderungen:
  Zuverlässigkeit, Effizienz
- Zielplattform:
  Netzdienste, GUI-System, Systemsoftware, Datenhaltung (flat files, HDBMS, RDBMS, ODBMS)

SD-Methode

Vorgehen zur Komplexitätsreduktion:
- Modularer Entwurf:
  - funktionale Abstraktion
  - Datenabstraktion
  - Schichtenarchitektur: verteilte Module
- hierarchische Organisation der Module

Qualitätskriterien:
- minimale Kopplung (coupling)
- maximale Bindung (cohesion)

Kopplung

Grad der Abhängigkeit zwischen Modulen.

Kopplungsarten:
- normal: alle Übergaben über explizite Parameter
  - Datenkopplung: nur benötigte Daten
  - Datenstrukturkopplung: ganze Struktur
  - Kontrollkopplung: Steuerflags
- global: externer globaler Speicherbereich
- inhaltlich: Adressierung auf das Modulinnere

Bindung

Grad der funktionalen Zusammengehörigkeit der inneren Modulelemente.

Bindungsarten:
- normal: eng zusammengehörende Funktionen mit gemeinsamer Datenstruktur
  - funktional
  - sequenziell
  - kommunizierend
- prozedural: diverse aufeinander folgende Funktionen
- zeitlich: gleichzeitig stattfindende Funktionen
- programmstrukturell: über Flag gesteuert
- zufällig: keine Gemeinsamkeiten

Funktionale Abstraktion

Modularisierung

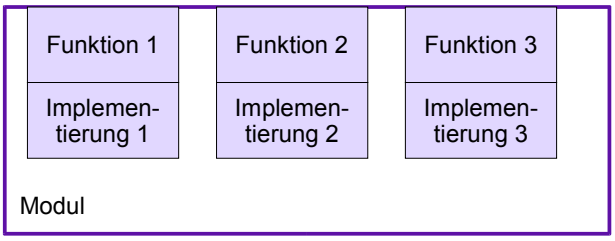
Aufgliederung eines Softwaresystems in mehrere Teile (Module, Software-Bausteine).

Moduleigenschaften:
- funktionale Einheit
- Kontextunabhängigkeit: umgebungsunabhängig entwickelbar, prüfbar, einsetzbar
- definierte Schnittstellen
- handlich, überschaubar, verständlich
- Black Box: Eingabewerte, Ausgabewerte und Funktion sind bekannt, nicht aber die Funktionsweise
- Geheimnisprinzip: Werte nicht von aussen einsehbar
- Kapselung: Werte nur durch eigene Operationen änderbar

Vorteile Modularisierung

- einfache Änderbarkeit: Austausch, Erweiterbarkeit
- bessere Wartbarkeit: Lokalisierung, Kontextunabhängigkeit
- Standardisierung
- Wiederverwendbarkeit
- Überprüfbarkeit, Testbarkeit
- Erleichterung der Arbeitsorganisation

Grafik funktionale Abstraktion



Nachteil: Nebst den Ausgabedaten weiter benötigte Daten müssen ausserhalb des Moduls, also global gespeichert werden und sind damit für andere Module manipulierbar.

Funktionales Modul

Eigenschaften:
- aktiv, aktionsorientiert
- Transformationsverhalten, Eingabedaten werden in Ausgabedaten transformiert
- kein internes Gedächtnis, identische Eingabedaten führen immer zu identischen Ausgabedaten

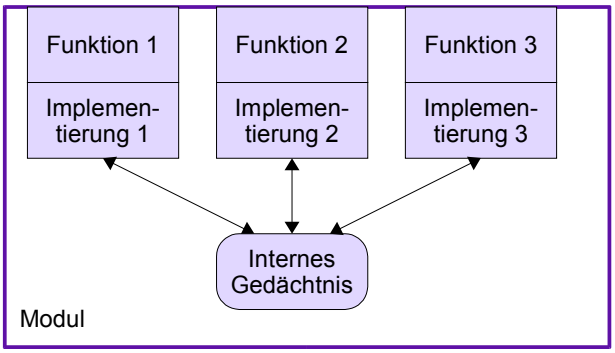
Einsatzgebiete:
- Steuerungs- und Koordinationsaufgaben (z.B. Hauptprogramm)
- Transformationsaufgaben (z.B. Compiler)
- Auswertungsaufgaben (z.B. mathematische Routinen)
- Hilfsaufgaben

Definition Datenabstraktion

Datenhaltung in einem Modul, Modul mit Gedächtnis.

Grade der Datenabstraktion:
- abstraktes Datenobjekt, Datenobjekt-Modul
- abstrakter Datentyp ADT

Grafik abstraktes Datenobjekt

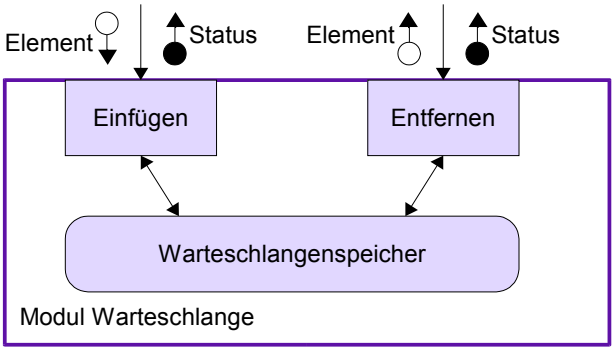


Datenobjekt-Modul

Eigenschaften:
- passiv
- Einheit von Datenstrukturen und Zugriffsoperationen
- Details der Datenstruktur nicht von aussen einsehbar
- Datenzugriff nur über zugehörige Zugriffsoperationen
- identische Eingabedaten führen nur zu identischen Ausgabedaten, wenn das interne Gedächtnis denselben Zustand hat

internes Gedächtnis:
- transient: an die Laufzeit gebunden
- persistent: in einer Datei aufbewahrt

Beispiel Datenobjekt-Modul



Abstrakter Datentyp

Verallgemeinerung eines abstrakten Datenobjekts.

- getrennte Typen- und Variablendeklaration
- beliebig viele Instanzen möglich
- entspricht Objektklasse in OO

Schichtenarchitektur

Definition Schichtenarchitektur

Aufteilung einer Funktion in Softwaremodule gemäss den verschiedenen Schichten einer verteilten Architektur.

3-Schichten-Anwendungsarchitektur (3-tier architecture):
- Präsentationslogik
- Geschäftslogik, Anwendungslogik
- Datenlogik
- Transaktionslogik TL (evtl. vierte Schicht):
  - meist in einem Transaktionsmonitor
  - Funktionsaufruf, z.B. Online-Menus
  - Verarbeitungsreihenfolge, z.B. Batch-Auslösung

Vor- und Nachteile Schichtenarchitektur

Vorteile:
- übersichtliche Srukturierung in Abstraktionsebenen
- liberale Strukturierungsmöglichkeit innerhalb der Schichten
- Unterstützung Wiederverwendbarkeit, Änderbarkeit, Wartbarkeit, Portabilität, Testbarkeit

Nachteile:
- Effizienzverlust durch Datentransport
- Schichten nicht immer eindeutig abgrenzbar

Präsentationslogik PL

Sichert die Verarbeitung von Schnittstellen zu externen Agenten.

- meist in vorgefertigten Modulen
- formale Plausibilisierung der Eingabedaten
- Benutzerschnittstelle:
  Aufbereitung von Masken, Listen, Formularen, GUI
- Systemschnittstelle:
  automatische Datenübertragung zu anderen Systemen

Geschäftslogik GL

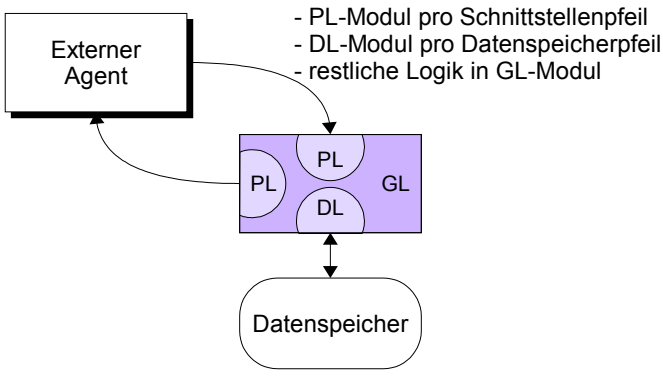
Verantwortlich für applikatorische Logik.

- typischerweise im Applikations-Server
  (übernimmt auch Transaktionssteuerung, Lastverteilung, Sicherheitsfunktion, GUI-Steuerung)
- inhaltliche Plausibilisierung der Eingabedaten
- Entscheidungen, Regeln, Berechnungen usw.
- Transformation der Daten

Datenlogik DL

- meist in vorgefertigten Modulen
- Zugriffe auf persistente Daten
- zugehörige Fehlerroutinen
- Einhaltung der Datenschutz- und Sicherheitskonzepte

Modulzuordnung



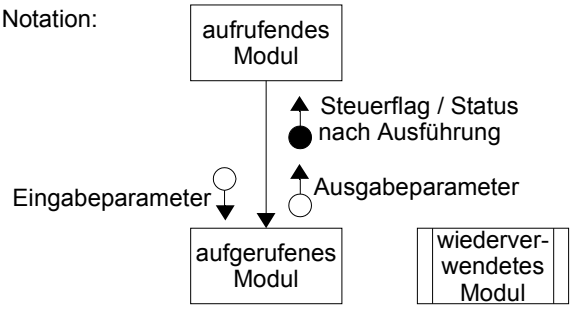
Beispiel Modulkatalog

M-Nr.	Modulverantwortung	Typ	GF
M1	Auftrag eingeben: Masken für Erfassung und Mutation	PL	F100
M2	Auftrag verwalten: Zugriff auf Auftragsdaten (CRUD)	DL	F100
M3	Auftragsbestätigung aufbereiten	PL	F100
M4	Kunde lesen: Zugriff auf Kundendaten (R)	DL	F100
M5	Artikel lesen: Zugriff auf Artikeldaten (R)	DL	F100
M6	Auftrag erfassen: inhaltliche Plausibilisierung, Berechnungen und Zusammenstellung der Auftragsdaten	GL	F100

Strukturdiagramm (Structure Chart)

Grafische Darstellung funktionaler Module, zeigt hierarchische Aufrufstruktur und Datenflüsse.

Notation:



Modulspezifikation

- Funktionsname
- Funktionsbeschreibung
- Eingabeparameter mit Datentyp, Format, Abhängigkeiten
- Ausgabeparameter mit Datentyp, Format, Abhängigkeiten
- Aufrufsyntax mit Beispiel
- Voraussetzungen für die Anwendung
- Bedingungen nach der Anwendung
- Verhalten bei inkorrekten Eingabewerten oder Fehlfunktion des Basissystems

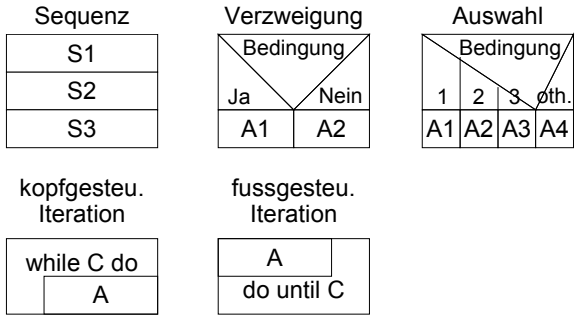
Modulentwurf

Beschreibung des Modul-Inneren (nur für komplexe Module).

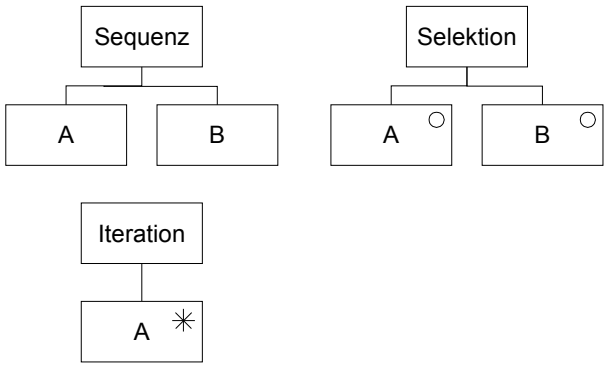
- Pseudocode
- Jackson-Diagramm
- Nassi-Shneiderman-Diagramm
- Programmablaufplan
- Entscheidungstabelle

Nassi-Shneiderman-Diagramm

Struktogramm nach DIN 66261
Nachteile: änderungsunfreundlich, unübersichtlich



Jackson-Diagramm



Entwurf Benutzerschnittstelle

SASD

Benutzerinteraktion

Aspekte:
- Benutzeroberfläche in Form von Masken und Interaktionselementen
- Steuerung der Dialogschritte

Massstab für die Gestaltung ist die Softwareergonomie.

Softwareergonomie

Menschen- und aufgabengerechte Gestaltung der Interaktion zwischen Mensch und Computer.

Kriterien für Benutzerfreundlichkeit:
- Aufgabenorientierung:
  Aufgabenangemessenheit, Ganzheitlichkeit
- Kalkulierbarkeit:
  Selbstbeschreibungsfähigkeit, Erwartungskonformität, Konsistenz, Feedback
- Steuerbarkeit:
  Wahlmöglichkeiten, Individualisierung
- Kontrollierbarkeit:
  Fehlerrobustheit, Kompatibilität

Dialogformen

- Menügesteuerter Dialog (pull-down, pop-up, Kaskade):
  - Aktionsmenü: Anwendungsfunktion
  - Eigenschaftsmenü: Parameter-Einstellung
- Formulardialog:
  zeichenorientierte Systeme, Funktionstastensteuerung
- Grafischer Dialog (graphical user interface GUI):
  Fenster, Interaktionselemente
- natürliche Sprache
- Frage-Antwort Dialog
- Kommando-Dialog
- Primärdialog: dient der direkten Aufgabenerfüllung
- Sekundärdialog: kurzzeitiger optionaler Hilfsdienst

Dialogentwurf

- Menüstrukturen definieren
- Bildschirmmasken entwerfen
- Dialogablauf festlegen

Menüstrukturen:
- nach Aufgabengebieten (verrichtungsorientiert):
  kann aus dem Funktionsbaum abgeleitet werden
- nach Geschäftsbereichen (objektorientiert)
- nach Benutzergruppen

Dialogablauf

- Interaktionsdiagramm
- Zustandsdiagramm
- Masken zu jedem Dialogzustand
- Bildschirmbeschreibung mit Ein-/Ausgabefeldern
- Beschreibung virtuelle Tasten
- Beschreibung Aktionen
- Petri-Netz für grafische Dialoge mit parallelen Prozessen