1 Software life Cycle (Wasserfallmodell)

Phase	**Produkt der Phase**
Analyse (Probleme finden, **Was** soll das System tun?)	Spezifikation
Design (**Wie** soll das System das tun?)	Programmiervorlage
Programmierung, Realisierung	Programm
Test	Testberichte, Mängel ⇒ besseres Programm
Wartung	

Parallel zu diesen Phasen:
- Standardisierung
- Review
- Dokumentation
- Testfälle (schon in Analysephase)
- Controlling
- Integrationsstrategie

Problem: Mehrere Datenbanken in einem Unternehmen können Mehrfachspeicherung von gleichen Daten und somit Redundanzen zur Folge haben, auch wenn jede Datenbank für sich normalisiert ist ⇒
unternehmensweites Datenmodell (UDM) = **Information Engineering** nach James Martin

Unterschied zwischen Software und Information Engineering:
vor der Analysephase kommt die

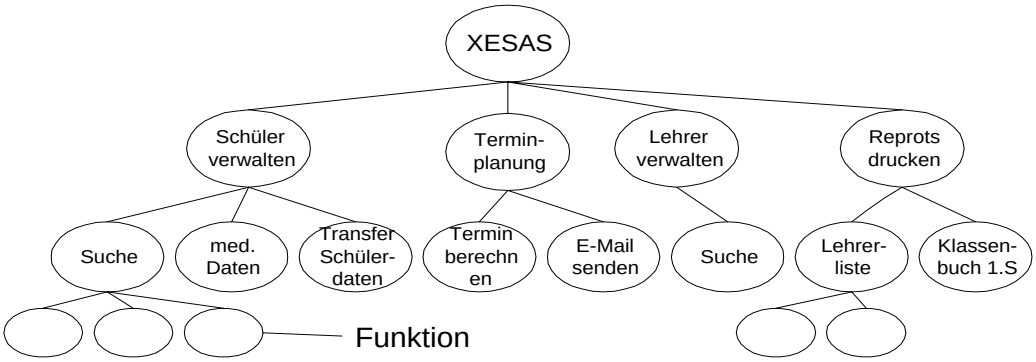
Planningphase
unternehmensweites betrachten des Problems

Vergleich:

Information Engineering	Städteplaner
Software Engineering	Architekt
Programmierer	Maurer

2 Funktionsdekomposition (Aufgabenzerlegung)

Die Funktionsdekomposition dient zur besseren Verdeutlichung der Spezifikation (welche Funktionen sollen enthalten sein?). Es werden die Funktionen Schritt für Schritt verfeinert (Top down) bis das unterste Level der Detailierung erreicht wird. Ist ein Hilfsmittel für die Sezifikation



Zweck:

- Mit dem Kunden ein Bild davon machen, was das Produkt leisten soll.
- Vertragsgrundlage

Nachteile:

- Manche Funktionen lassen sich nicht eindeutig zuordnen (z.B. Stundenplan für Lehrer ausdrucken, sowohl zur Lehrerverwaltung als auch Reports drucken)
- Kann sehr groß und unübersichtlich werden
- Verknüpfung zwischen Daten und Funktionen fehlen ($\Rightarrow$ DFD)

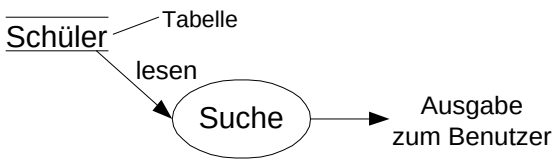
Wenn man die Funktionsdekomposition mit den Datenflüssen erweitert (Funktionsdekomposition + ERD) erhält man ein

3 Datenflußdiagramm

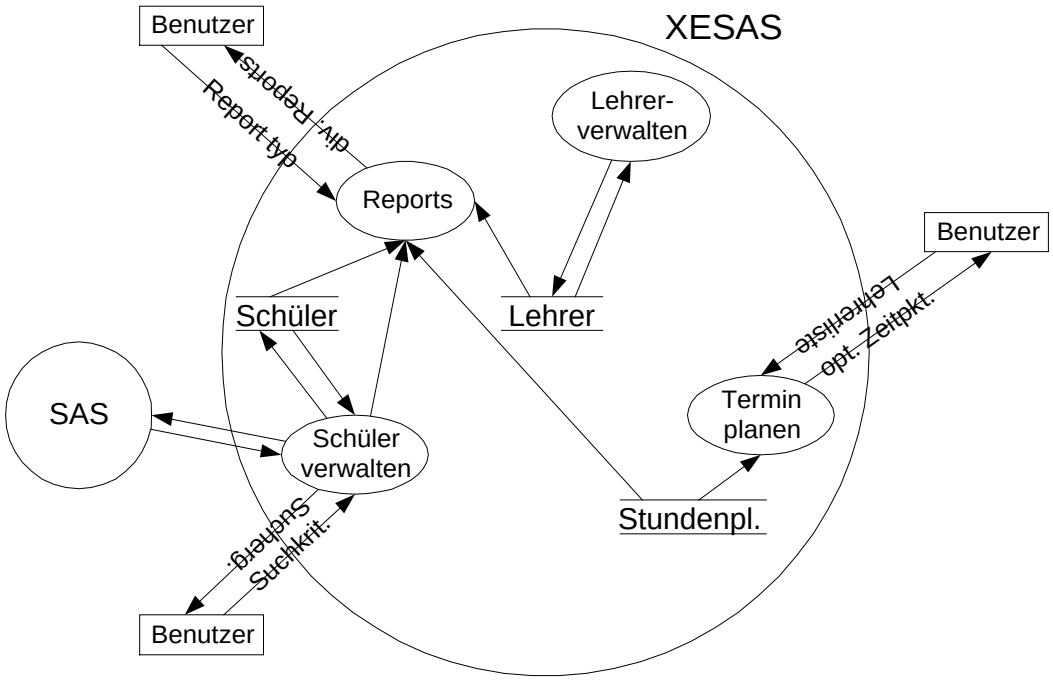
Das Datenflußdiagramm zeigt die Prozesse und den Fluß der Daten durch diese Prozesse.

4 Grundkomponenten:

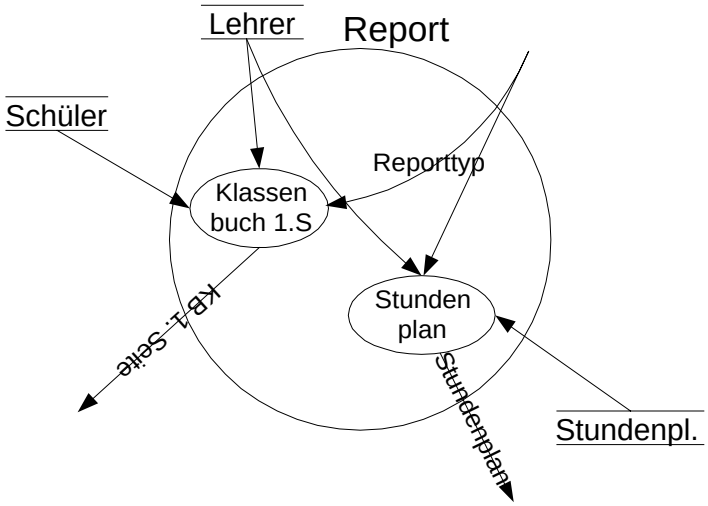
1. **Datenfluß**: Pfeil mit Name darüber, zeigt den Datenfluß durch das System (z.B. Suchergebnis)
2. **Prozeß**: Abgerundetes Rechteck (Ellipse) mit Namen des Prozesses (sprechender Name!), Keine andere Information über den Prozeß in Datenflußdiagramm (z.B. Termin planen)
3. **Data Store**: Name zwischen zwei waagrechten Strichen, repräsentiert ein lokales File in das entweder Daten eingelesen werden, oder von dem Daten gelesen werden (z.B Schüler)
4. **Terminator**: Rechteck mit Namen, Gibt die Herkunft (Source) bzw. den Empfänger (Sink) der Daten an, doch sie werden meistens nicht in die Datenflußdiagramm eingezeichnet (z.B. Benutzer)



Jede Funktion wird auf eine Seite aufgeteilt, es werden alle Datenströme von und zur Funktion eingezeichnet.



DFD für den Prozeß Report:



Ein Datenflußdiagramm zeigt den Datenfluß in einem konkreten Prozeß, der wieder aus mehreren Unterprozessen besteht, die man wieder in eigene Datenflußdiagramm zerlegen kann.
So kann man ein System bis in die tiefsten Ebenen zerlegen. Ein Datenflußdiagramm gibt uns keine Details über die Datenflüsse innerhalb der Unterprozesse, um mehr über diese Datenflüsse zu erfahren, muß für den Prozeß ein eigens DFD erstellt werden.
Die Aufteilung der Diagramme muß konsistent sein, d.h. die Funktion im Unterdiagramm muß genau so viele Zu- und Abflüsse haben, wie das übergeordnete Diagramm.

3.1 Hyperdiagramm (in IEF: Repository)

großes Diagramm mit allen Informationen (Funktionen, Tabellen, Beziehungen)

Nachteil: Durch die vielen Informationen zu kompliziert und unübersichtlich (wird nicht gezeichntet) $\Rightarrow$ Teilung in Teildiagramme (DFD), entweder Funktionen oder Informationen (Tabellen, Beziehungen) weglassen.

IEF (Information Engineering Facility)

CASE - Tool (Computer Aided Software Engineering): EDV unterstütztes Hilfsmittel für die Erstellung von DFD, ERD, FD.