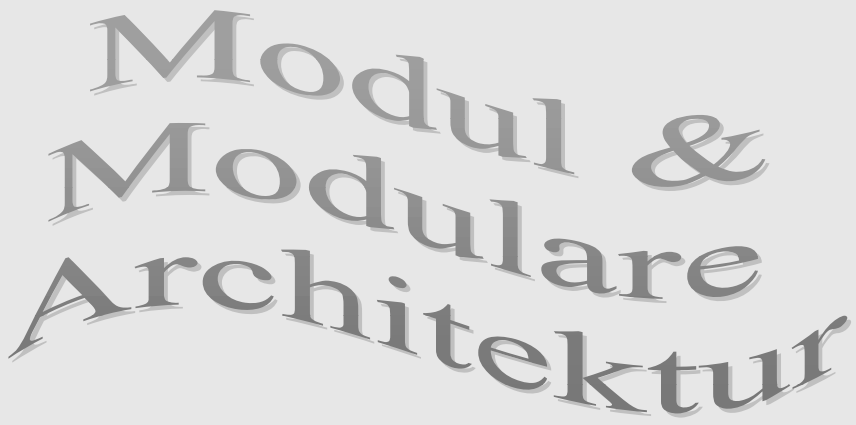
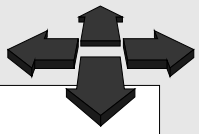
Einführung in die Softwaretechnik



Auf dem Weg zur objektorientierten Architektur

Einführung in die Softwaretechnik



- **Überblick**
  - Architekturbegriff: Zerlegung in Komponenten,
  - Qualitätsanforderungen, Entwurfskriterien,
  - Struktur und Interaktion von Komponenten,
  - Entwurfsmuster und Musterarchitektur,
  - Grundzüge modularer Architektur
- **Einordnung**
  - Teil 1: Große Software
  - baut auf Modellierung auf
  - bildet eine Einheit mit Softwareentwurf
- **Lernziele**
  - Grundbegriffe modularer Architektur verstehen,
  - Qualitätsanforderungen und Entwurfskriterien einschätzen lernen
  - Grundlagen arbeitsteiliger Softwareentwicklung kennen.

Softwareentwurf
- im Zentrum der Softwareentwicklung -

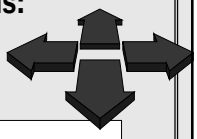
Ziele
- Komplexität verringern
- Arbeitsteiliges Entwickeln vorbereiten
- Existierende Lösungen einbinden
- Weiterentwicklung ermöglichen

Qualitätsanforderungen
- Verständlichkeit
- Änderbarkeit
- Wiederverwendbarkeit

Mittel
- Erarbeitung einer geeigneten Architektur

Einführung in die Softwaretechnik AB Softwaretechnik

Das Beispiel für Modularisierung aus der Praxis: Die SAP-Filenet-Brücke von d.d. synergy - Das Document Warehouse for SAP

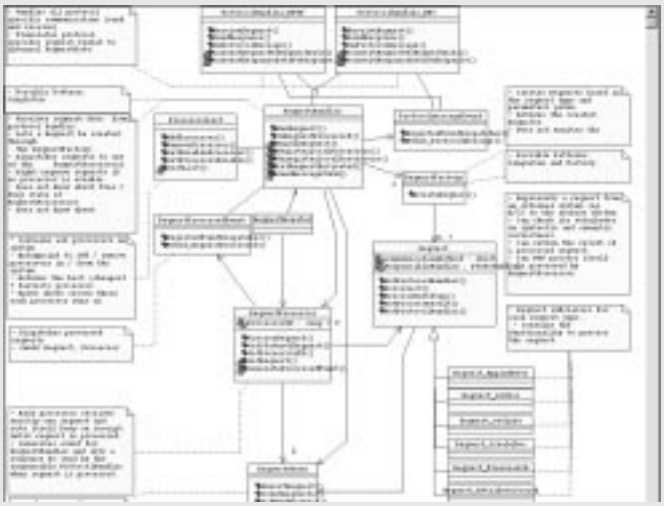


- **Die Aufgabenstellung**
  - **Aus SAP-Modulen soll das Document-Management-System (DMS) Filenet aufgerufen werden können. Der Zugang zum DMS erfolgt wahlweise über einen Remote Function Call (RFC) oder über ein HTTP-Aufruf**
- **Das Projekt**
  - Das Projekt wird von der Hamburger Firma **d.d. synergy** (www.dd_synergy.de) mit einem Team bestehend aus 4 - 5 Entwicklern durchgeführt.
- **Eingesetzte Techniken**
  - Standard Modellierungsmittel der objektorientierten Modellierung: CRC-Karten, Entwurfsmuster; Programmiersprache ist C++.
- **Vorgehensweise**
  - **Schnelle Entwicklung eines funktionalen Prototypen um daraus Erfahrungen und Rückschlüsse für den weiteren Ausbau der Brückenmodule zu ziehen.**

Einführung in die Softwaretechnik AB Softwaretechnik

Das Ziel: Wir verstehen eine OO-Architektur Hier: Die Architektur des Document Warehouse

Dazu notwendig:
- **Was ist Modularisierung?**
- **Kriterien für Modularisierung**
- **Was sind CRC-Karten?**
- **Was ist UML?**
- **Was sind Entwurfsmuster?**
- **Wie paßt das alles zusammen?**



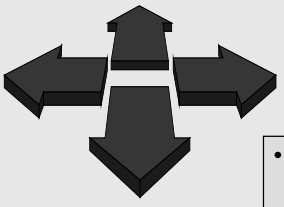
Einführung in die Softwaretechnik AB Softwaretechnik

Software-Architektur

- **Die *Architektur* eines Softwaresystems beschreibt seine äußere Ausgestaltung und seinen inneren Aufbau.**
- **Die *äußere Architektur***
  - bestimmt die geographische Plazierung, Verfügbarkeit und die Art der Benutzung des Softwaresystems;
  - legt seine Verbindung mit anderen Komponenten des Basissystems (aus Hardware und Software), wie

    Betriebssystem, Datenbank, Fensterverwaltungssystem, Verteilung und Vernetzung fest.
  - wird durch vorhandene Bibliotheken und Frameworks geprägt.
- **Die *innere Architektur***
  - bestimmt die Zerlegung des Systems in Komponenten und ihr Zusammenwirken;
  - legt sowohl die statische Struktur als auch die Möglichkeit zur dynamischen Interaktion der Komponenten fest.

Einführung in die Softwaretechnik AB Softwaretechnik



Das Modulkonzept

- Die *klassische imperative Programmierung* ist eng mit dem **Modulkonzept** verknüpft.
- Entstanden aus der Notwendigkeit, große Programmtexte in für den Übersetzer faßliche Einheiten zu zerlegen, ist das Modulkonzept zum zentralen *Organisationskonzept* für Entwürfe und Programmtexte geworden.
- Die neueren Entwicklungen bei (imperativen) Programmiersprachen zeichnen sich u.a. durch eine neue Interpretation des Modulkonzepts aus (z.B. Typ entspricht Modul, Klasse entspricht Modul).
- Module werden hier als wichtiges Konstruktionsmerkmal einer imperativen Sprache kurz eingeführt. Die dahinterstehenden Prinzipien der Verwendung (Pragmatik) diskutieren wir beim *Klassenentwurf*.

Einführung in die Softwaretechnik AB Softwaretechnik

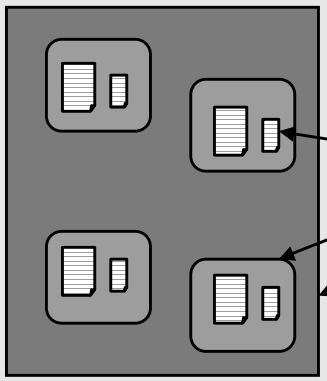
Modul

Modul : Sammlung von Programmobjekten und Operationen, die eine Einheit des Entwurfs und der Compilation ist. Ein Modul bildet einen Namensraum. Nur die an der Exportschnittstelle sichtbar gemachten Bezeichner können von anderen Modulen über Importschnittstellen benutzt werden.

Funktionale Sprachen (z.B. Miranda) kennen ebenfalls Module; in Prolog kann der Workspace nur durch Laden und Löschen von Termen verändert werden.

Einführung in die Softwaretechnik AB Softwaretechnik

Das Modulkonzept



- Das **Modulkonzept** leistet vorrangig zweierlei:
  - Es ermöglicht **Datenabstraktion** und
  - verfeinert das Konzept der **Namensräume**:
    - Prozedur,
    - Modul,
    - Programm.
- **Datenabstraktion** bedeutet hier:
  - Die Repräsentation von Daten ist vor den benutzenden Programmeinheiten verborgen, so daß Daten nur über die exportierten Operationen bearbeitet werden können.

Einführung in die Softwaretechnik AB Softwaretechnik

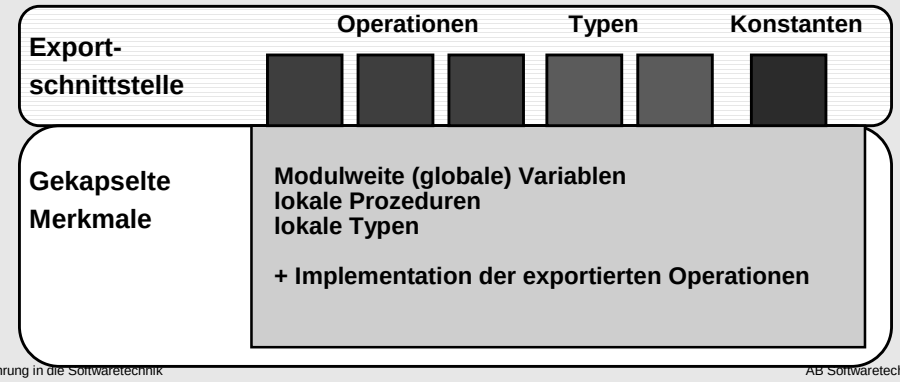
Struktur und Interaktion modularer Software

- **Die Struktur modularer Software ist definiert durch *Benutzt-Beziehung zwischen Modulen*.**
- **Modul A benutzt Modul B wenn A ein Element der Exportschnittstelle von B importiert. A heißt benutzendes (applizierendes), B benutztes (definierendes) Modul.**
- ***Statische Benutzung* "hat Kenntnis von"**
  - bezieht sich auf Typen und Konstanten,
  - wird zur Übersetzungszeit wirksam;
- ***Dynamische Benutzung* "ruft auf"**
  - bezieht sich auf Operationen (Prozeduren und Funktionen),
  - wird zur Laufzeit wirksam,
  - legt die möglichen Interaktionen fest.
- **Module sind statische Komponenten, Prozeduren bzw. Funktionen dynamische Komponenten einer modularen Architektur!**

Einführung in die Softwaretechnik AB Softwaretechnik

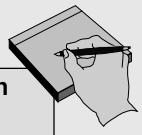
Modul - Exportschnittstelle

- Um **Programmobjekte und ihre Bezeichner über Modulgrenzen** hinaus verwenden zu können, müssen sie nach **außen bekanntgemach**t werden.
- Die **Exportschnittstelle** definiert, welche deklarierten Objekte ein Modul **seiner Umgebung zur Verfügung** stellt.

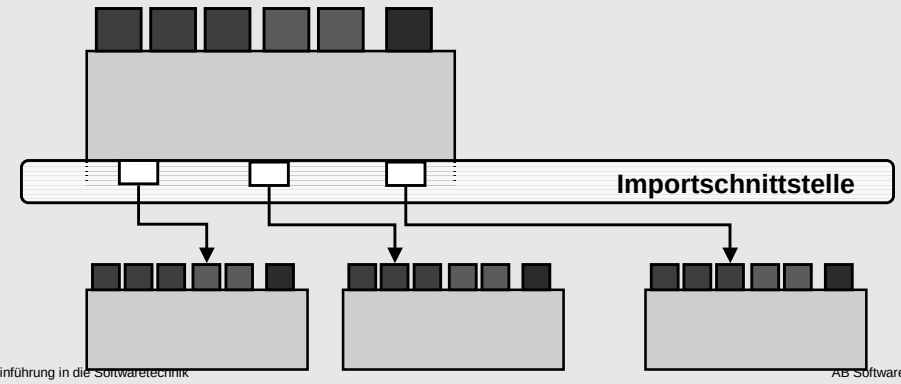


Einführung in die Softwaretechnik AB Softwaretechnik

Import



- Um **Programmobjekte und ihre Bezeichner über Modulgrenzen** hinaus zu verwenden, müssen sie gezielt **angefordert** werden.
- Die **Importschnittstelle** beschreibt, welche deklarierten und exportierten Objekte ein Modul **von seiner Umgebung benötigt**.



Einführung in die Softwaretechnik AB Softwaretechnik

Datenabstraktion (1): Objektmodul

Bsp.: Das Objektmodul Keller

```
procedure Create();
procedure Push(e : KellerElement);
procedure Pop();
procedure Top(Var k : KellerElement);
```

Schnittstelle = Spezifikation

```
type Keller;
var MeinKeller: Keller;
```

- ein Exemplar (Objekt): im Datenraum des Moduls;
- Operationen: zustandsverändernde Prozeduren, operieren auf modulglobalen Variablen;
- Implementation flexibel: z.B. als Reihung oder verkettete Liste;
- Elementtyp fest (wird in der Regel importiert).

Modulrumpf = Implementation

Einführung in die Softwaretechnik AB Softwaretechnik

Beispiel Objektmodul LeserListe

```
DEFINITION MODULE LListe;
...
PROCEDURE FuegeEin(rLeser:RefLeser);
PROCEDURE EntferneLeser(rLeser:RefLeser);
...
PROCEDURE IstEnthalten(rLeser:RefLeser):
  BOOLEAN;
PROCEDURE IstLeer(): BOOLEAN;
PROCEDURE SucheLeser(VergleichsString:
  String; kriterium:CARDINAL);
...
END LListe.
```

- **Kapselung *genau eines* Exemplares von LeserListe**
- **für Komponenten des Systems sinnvoll, die nur als ein Exemplar benötigt werden**
- **Objektmodul für 'Leser' sinnlos, da davon mehrere Exemplare benötigt werden**
- **Beispiel: 'LeserListe'**

Einführung in die Softwaretechnik AB Softwaretechnik

Datenabstraktion (2): Typmodul

Bsp.: Das Typmodul Keller

```
type Keller;
procedure Create(): Keller;
procedure Push(e : KellerElement; k : Keller) : Keller;
procedure Pop(k : Keller) : Keller;
procedure Top(k : Keller) : KellerElement;
```

Schnittstelle = Spezifikation

- Kein Exemplar im Modul, keine modulglobalen Variablen;
- Operationen als wertliefernde Funktionen;
- Typexport erlaubt das Anlegen von Exemplaren in benutzenden (applizierenden) Modulen;
- Implementation flexibel
  z.B. als Feld oder verkettete Liste;
- Elementtyp fest (wird in der Regel importiert).

Modulrumpf = Implementation

Einführung in die Softwaretechnik AB Softwaretechnik

Beispiel für Typmodule

```
DEFINITION MODULE Leser;

TYPE LeserNr = CARDINAL;
...
TYPE RefLeser;
...
PROCEDURE Erzeugen(): RefLeser;
PROCEDURE Loeschen(rLeser:RefLeser);
PROCEDURE GibNachname (rLeser: RefLeser):RefString;
PROCEDURE GibLeserNr (rLeser: RefLeser): LeserNr;
...
END Leser.
```

- **mehrere Exemplare können existieren**
- **für Komponenten des Systems sinnvoll, die mehrfach benötigt werden**
- **Typmodul für LeserListe sinnvoll, wenn verschiedene Gruppen von Lesern verwaltet werden sollen**
- **Beispiel: 'Leser'**

Einführung in die Softwaretechnik AB Softwaretechnik

Module als Entwurfseinheit

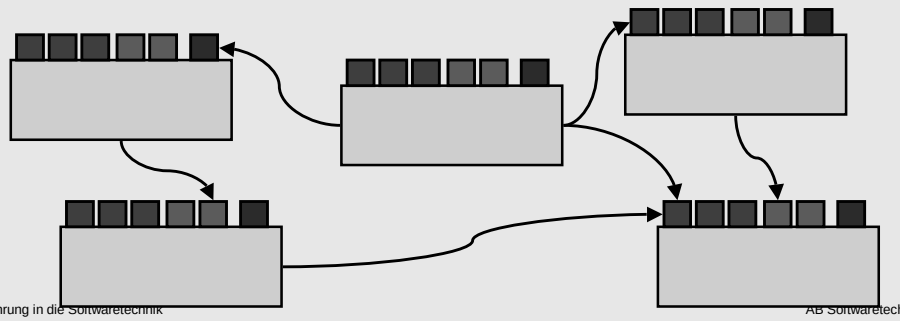
- Module bilden klassischerweise die Bausteine des Programmentwurfs im Großen. Um zu einer qualitativ hochwertigen Softwarearchitektur zu gelangen, müssen bestimmte Entwurfsprinzipien beachtet werden, u.a.:
  - das Geheimnisprinzip,
  - die Modulkopplung,
  - die Modulkohäsion.

Einführung in die Softwaretechnik AB Softwaretechnik

Modul-Architektur

Module **benutzen** andere Module:

- Das Zusammenspiel verschiedener Module bezeichnet man auch als **Architektur**.
- Die Benutzt-Beziehung **verkoppelt** Module miteinander.
- Die Interaktion zwischen Modulen soll ausschließlich über Operationen, d.h. über Prozeduren und Funktionsprozeduren erfolgen. (Prozedurale Schnittstellen)



Einführung in die Softwaretechnik AB Softwaretechnik

Das Geheimnisprinzip (Information Hiding)

Geheimnisprinzip (*information hiding*) bezeichnet nach Parnas ein Entwurfsprinzip für Module.

- Module sollen *Entwurfsentscheidungen* verbergen, die als tendenziell änderbar erkannt wurden, insbesondere Details, die Struktur oder Umsetzung der Funktionalität eines Moduls betreffen.
- Komponenten eines Software-Systems werden als *Black Box* betrachtet, die nur relevante Informationen nach außen zeigen. Dadurch werden die Interaktionen zwischen den Komponenten möglichst einfach gehalten, was die Fehleranfälligkeit reduzieren und Änderungen der Realisierung lokal halten soll.

David L. Parnas: *On the Criteria to be Used in Decomposing Systems into Modules*. Communications of the ACM, 15[12], S. 1053-1058, Dezember 1972.

Einführung in die Softwaretechnik © Floyd, Züllighoven in Rechenberg, Pomberger AB Softwaretechnik

Modulkopplung

Modulkopplung (*coupling*) ist ein Konzept zur Gestaltung von Modulbeziehungen (vgl. Yourdon & Constantine), die nach Graden abgestuft sind.

- Modulkopplung bezeichnet den Grad der Abhängigkeit zwischen Modulen, z.B. ausgetauschte Daten, Ablaufsteuerung oder gemeinsam benutzte Daten.
- Eine Minimierung der Modulkopplung ist anzustreben, insbesondere die Vermeidung zyklischer Benutzung.

E. Yourdon, L.L. Constantine: *Structured Design: Fundamentals of a discipline of computer program and systems design*. Englewood Cliffs, NJ: Prentice Hall International 1979

Einführung in die Softwaretechnik © Floyd, Züllighoven in Rechenberg, Pomberger AB Softwaretechnik

Modulkohäsion

Modulkohäsion (*cohesion*) bezeichnet den inneren Zusammenhalt eines Moduls und wird allgemein auf die ”Stimmigkeit” oder den funktionalen Zusammenhang der in einem Modul gekapselten Komponenten bezogen.
Ein Modul soll eine möglichst hohe fachliche und logische Kohäsion aufweisen.

- Die Begriffe Kohäsion und Kopplung beziehen sich auf beliebige benannte und getrennt übersetzbare Einheiten eines Programmsystems.
- Bei Verwendung des Parnas’schen Modulkonzepts sind nur die höchsten Kohäsions- und niedrigsten Kopplungsstufen relevant.
- **Kohäsion bezeichnet den inneren Zusammenhalt eines Moduls. Er ist am höchsten, wenn**
  - **das Modul eine Datenstruktur kapselt, oder**
  - **genau eine Funktion realisiert.**
- **Funktionsmodule werden nur in Sonderfällen verwendet (z. B. Zufallszahlengenerator).**
- **Kopplung bezeichnet die Wechselwirkung zwischen Modulen. Anzustreben sind die kleinsten Stufen**
  - Datenkopplung über parametrisierte Prozeduren, oder
  - keine Kopplung.

Modul als programmiersprachliches Konstrukt: Zusammenfassung und Diskussion

- **Module** sind *Sammlungen von Programmobjekten und Algorithmen*:
  - Sie sind eine *Einheit für die Übersetzung*.
  - Sie sind *keine direkt aufrufbaren Programmeinheiten* (wie Prozeduren).
  - Sie können *nicht zur Deklaration von Bezeichnern* verwendet werden (keine Typen und daher keine Exemplare).
- Module *verbergen Implementationen* d.h. ihren inneren Aufbau und zeigen nur ihre *Schnittstelle*:
  - Es gibt unterschiedliche starke Möglichkeiten des Verbergens.
  - Der Aufbau einer Schnittstelle unterliegt bestimmten Qualitätskriterien.
- Module sind die *Einheiten des klassischen Entwurfs im Großen*.
  - Es gelten das *Geheimnisprinzip* und die Prinzipien der *Kopplung* und *Kohäsion*.

Qualitätsanforderungen: Änderbarkeit

Änderungen sollen nur lokale Auswirkungen haben!

Arten von in Betracht zu ziehenden **Änderungen:**

Fachliche:
beziehen sich auf Funktionalität des Softwaresystems, d.h. auf die modellierten Objekte und Operationen

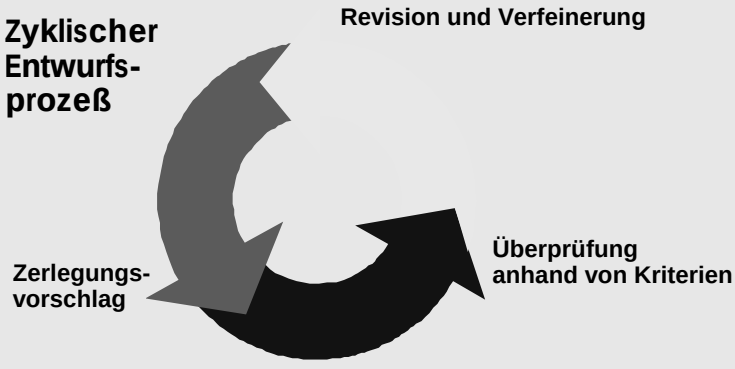
Technische:
beziehen das Basissystem: auf Programmiersprache und -umgebung, Datenbank, Fensterverwaltung, Vernetzung etc.

Strukturelle:
beziehen sich auf die entworfene Struktur selbst, z.B. aufgrund von Leistungsmessungen beim Einsatz

Einführung in die Softwaretechnik AB Softwaretechnik

Entwurfskriterien (Architekturprinzipien)

- **geben Anhaltspunkte dafür, wie eine Zerlegung in Komponenten erarbeitet werden soll;**
- **orientieren sich an den Qualitätsanforderungen;**
- **sind auf die Art der gewählten Zerlegung abgestimmt.**

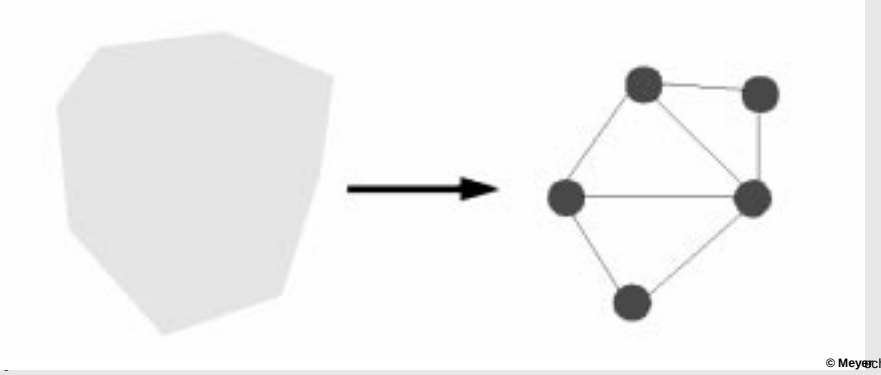


Einführung in die Softwaretechnik AB Softwaretechnik

Zerlegbarkeit

Kriterien der Modularisierung:

Zerlegbarkeit: Ein Entwurfsproblem sollte sich in kleine weniger komplexe Teilprobleme zerlegen lassen, die als Einheiten abgebildet werden. Diese sollten eine einfache Struktur bilden und weitgehend unabhängig konstruiert werden können.



Zerlegbarkeit in der SAP-Filenet Brücke

- **Aufträge an das Document-Management-System werden mit unterschiedlichen Protokollen geliefert:**
  - **Ein „Auftrag“ ist unabhängig vom gewählten Protokoll (http, RFC). Dieser Umstand motiviert einen „Konverter“, dessen Aufgabe es ist, aus den syntaktisch verschiedenen Aufträgen ein einheitliches, für die weitere Verarbeitung unabhängiges Modell des Auftrags zu generieren.**
- **Für die Verarbeitung von Aufträgen stehen eine Anzahl von Prozessoren zur Verfügung. Die Verwaltung dieser Prozessoren ist eine eigenständige Funktionalität.**

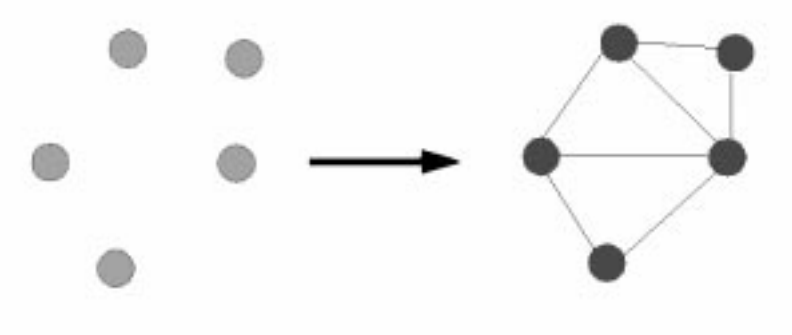
Kriterien der Modularisierung:

Zerlegbarkeit: Ein Entwurfsproblem sollte sich in kleine weniger komplexe Teilprobleme zerlegen lassen, die als Einheiten abgebildet werden. Diese sollten eine einfache Struktur bilden und weitgehend unabhängig konstruiert werden können.

Kombinierbarkeit

Kriterien der Modularisierung:

Kombinierbarkeit: Die Einheiten sollten sich durch einfache Rekombination zu neuen Softwaresystemen in verschiedenen Anwendungsbereichen zusammenfügen lassen.



Qualitätsanforderungen: Wiederverwendbarkeit

Softwaresysteme sollen mehrfach nutzbar werden!

Anliegen der **Wiederverwendung**

Fachliche:
Eine Lösung soll auf verwandte Problemstellungen übertragbar und auf eine umgreifende Domäne verallgemeinerbar sein.

Technische:
Eine Lösung soll auf ein anderes Basissystem übertragbar oder auf einer Klasse von Basissystemen nutzbar sein.

Aufwandbezogene:
Vorgefertigte Teillösungen sollen benutzt werden (z.B. Bibliotheken und Rahmenwerke)

Kombinierbarkeit und Wiederverwendbarkeit in der SAP-Filenet-Bridge

- **Für das eigentliche DMS wird eine objektorientierte Hülle entworfen. Zweck dieser Hülle ist es, das DMS auch von anderen Dienstleistungsabnehmern als SAP ansprechbar zu machen.**

Kriterien der Modularisierung:

Kombinierbarkeit: Die Einheiten sollten sich durch einfache Rekombination zu neuen Softwaresystemen in verschiedenen Anwendungsbereichen zusammenfügen lassen.

Verständlichkeit

Kriterien der Modularisierung:

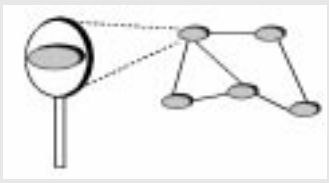
Verständlichkeit: Jede Einheit eines Softwaresystems sollte weitgehend unabhängig von den anderen verständlich sein.

Aus Entwurfsdokumenten muß die Funktionsweise des Softwaresystems klar werden!
Ebenen der **Verständlichkeit:**

Gesamtsystem:
Statische **Struktur** von Komponenten soll die mögliche dynamische **Interaktion** von Komponenten aufzeigen.

Komponenten:
Eine Komponente soll alles enthalten, was zusammengehört und keine Seiteneffekte auf andere haben.

Beziehungen zwischen Komponenten:
Direkte und indirekte Abhängigkeiten zwischen Komponenten sollen nachvollziehbar sein.



Verständlichkeit in der SAP-Filenet-Bridge

- **Entwurfseinheiten beziehen sich auf Gegenstände oder Abläufe, die aus der DMS- und Softwaretechnik-Welt bekannt sind:**
  - **Konverter für Aufträge: Eine „einfache“ Syntaxanalyse**
  - **Aufträge: Vergegenständlichen die unterschiedlichen Auftragsarten (Erzeugen eines Dokuments; Holen eines Dokuments aus dem DMS; Anfrage, ob ein Dokument vorhanden ist.) Achtung: Die Auftragsart ist von der Durchführung des Auftrags zu unterscheiden.**
  - **Die Prozessorverwaltung zur Abarbeitung von Aufträgen kapselt die snchrone oder asynchrone Abarbeitung von Aufträgen-**

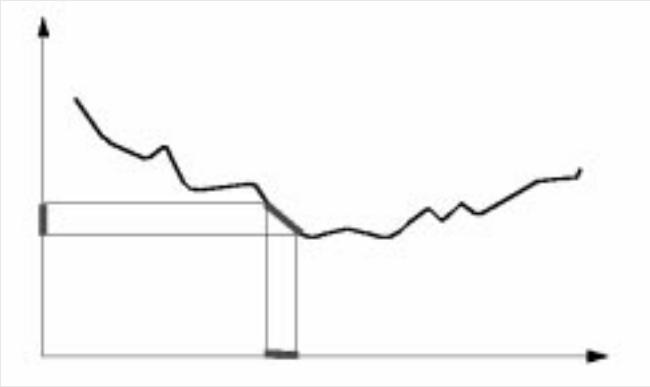
Kriterien der Modularisierung:

Verständlichkeit: Jede Einheit eines Softwaresystems sollte weitgehend unabhängig von den anderen verständlich sein.

Kontinuität

Kriterien der Modularisierung:

Kontinuität: Eine Änderung des Entwurfsproblems, die aus dem Anwendungsbereich oder dem technischen Kontext eines Softwaresystems resultiert, sollte Änderungen nur in einer oder wenigen Entwurfs- und Konstruktionseinheiten erfordern.



Kontinuität in der SAP-Filenet-Bridge

- **Folgende - in der Zukunft nicht unwahrscheinliche Änderungen im Anforderungsprofil an die Brücke - sind in das Design eingeflossen:**
  - **Das Protokoll, mit dem das DMS angesprochen wird, ändert sich. Diese Änderungen soll der Protokoll-Konverter abfangen.**
  - **Die Art der Aufträge ändert sich. Diese Änderungen sollen von der Entwurfseinheit zur Erzeugung von Aufträgen verarbeitet werden.**

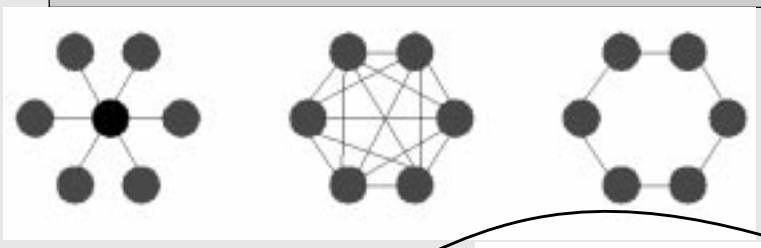
Kriterien der Modularisierung:

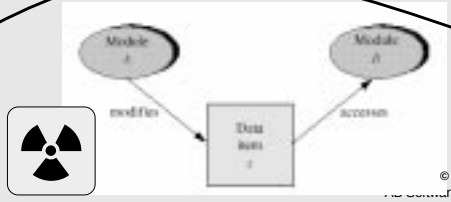
Kontinuität: Eine Änderung des Entwurfsproblems, die aus dem Anwendungsbereich oder dem technischen Kontext eines Softwaresystems resultiert, sollte Änderungen nur in einer oder wenigen Entwurfs- und Konstruktionseinheiten erfordern.

Schnittstellen

Regeln für die Modularisierung:

Wenige, kleine, explizite Schnittstellen: Jede Entwurfs- und Konstruktionseinheit sollte mit möglichst *wenig anderen* interagieren, dabei *explizit* so *wenig Information* wie möglich und nötig *austauschen*.

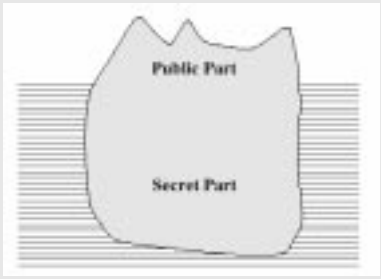




Geheimnisprinzip

Regeln für die Modularisierung:

Geheimnisprinzip: Nur relevante Merkmale einer Entwurfs- und Konstruktionseinheit sollten für Klienten sichtbar und zugänglich sein.



Datenabstraktion

- **Wichtigstes Entwurfskriterium: *Datenstruktur* samt *Operationen* wird in einer benannten Komponente zusammengefaßt:**
  - als *abstrakter Datentyp spezifiziert* (meist teil-formalisiert),
  - durch Komponenten des Softwaresystems realisiert.
- **Zu unterscheiden sind *Abstraktionsstufen*:**

1 *Kapselung*: genau ein Exemplar einer Datenstruktur samt Operationen in einer Komponente;

2 *Abstrakter Datentyp*: nur das Baumuster (Typ) wird gekapselt, beliebig viele Exemplare können angelegt werden.

3 *Vorgefertigte Datenstrukturen* werden eingebunden.

4 *Verallgemeinerung bzw. Spezialisierung*: gemeinsame Eigenschaften werden allgemein verfügbar gemacht.

5 *Parametrisierung abstrakter Datentypen*: Datenstrukturen können unabhängig vom Typ ihrer Elemente realisiert werden.

Struktur und Interaktion von Komponenten

- **Die Struktur**
  - definiert den statischen Aufbau eines Softwaresystems aus Entwurfskomponenten und ihren Beziehungen,
  - bestimmt die Realisierung der Entwurfskomponenten als Programmkomponenten (eins zu eins oder nach definierten Regeln) ,
  - läßt sich aus der Gesamtheit der Programmtexte entnehmen.
- **Die Interaktion**
  - findet dynamisch zwischen den Komponenten zur Laufzeit statt,
  - wird durch die Struktur zwar ermöglicht, aber durch Benutzereingaben ausgelöst,
  - kann nur während der Laufzeit protokolliert werden.

Einführung in die Softwaretechnik AB Softwaretechnik

Modulare Entwurfskriterien: Benutzt-Hierarchie

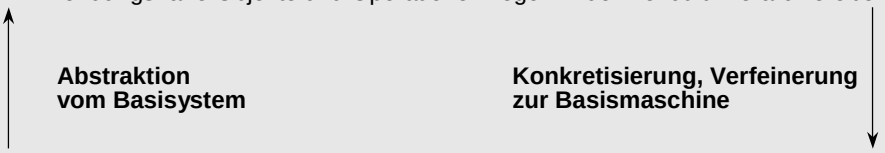
Ziel: Intellektuelle Beherrschbarkeit von Entwürfen

- **Aus A benutzt B folgt, daß weder B noch ein in der Hierarchie tiefer liegendes Modul A benutzt:**
  - dynamisch: durch Aufrufen von exportierten Prozeduren
  - statisch: durch Kenntnis haben von Typen oder Konstanten
- **Starke Benutzt-Hierarchie:**
  - **das System ist in Ebenen gegliedert,**
  - **nur Benutzung von einer Ebene zur nächsten ist erlaubt**
- **Schwache Benutzt-Hierarchie:**
  - **Ebenen dürfen übersprungen werden.**
- **Achtung: die Benutzt-Hierarchie ist nicht immer möglich! (z. B. bei Verwendung eines Fensterverwaltungssystems)**

Einführung in die Softwaretechnik AB Softwaretechnik

Modulare Entwurfskriterien: Stufenweise Abstraktion

Anwendungsnahe Objekte und Operationen liegen in der Benutzt-Hierarchie **oben.**



Objekte und Operationen des Basissystems liegen in der Benutzt-Hierarchie **unten.**

Verwendungsarten:

- Trennung der fachlichen von der technischen Ebene.
- Herausziehen der zur Koordinierung, Steuerung des Zusammenwirkens mehrerer Komponenten auf höherer Ebene benötigten abstrakten Objekte aus der auf der tieferen Ebene verwalteten Feinstruktur.
- Gemeinsame Verwaltung gleichgearteter Objekte auf der tieferen Ebene,von denen der höheren Ebene auf Anfrage jeweils eines zur Verfügung gestellt wird.
- Trennung der allgemein verwendbaren tieferen Ebenen von einer problem-spezifischen höheren Ebene

Einführung in die Softwaretechnik AB Softwaretechnik

Komplexität und Zerlegungsebenen

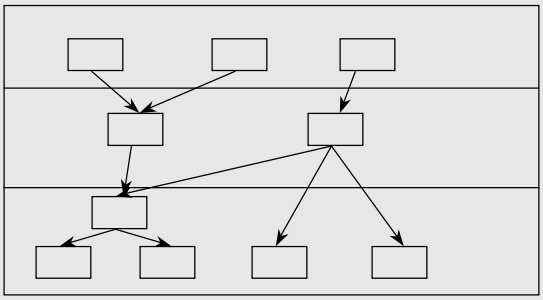
Zielkonflikte:

- **Module sollen klein sein (Verständlichkeit, Änderbarkeit)**
- **Anzahl der Module soll klein sein (Überblick über das Gesamtsystem)**
- **Schnittstellen zwischen den Modulen sollen klein sein**
- **Gesamtzahl der Schnittstellen soll klein sein**

Lösungen: z. B. abstrakte Maschinen, mehrstufige Zerlegung

Einführung in die Softwaretechnik AB Softwaretechnik

Zerlegung in abstrakte Maschinen



Grobzerlegung: vertikal

Feinzerlegung: horizontal

Die Zerlegung in abstrakte Maschinen ist eine wichtige modulare Musterarchitektur.

Softwaresysteme werden dabei in Schichten (Ebenen) zerlegt.

Die Benutzt-Hierarchie wird in der Regel innerhalb der Ebenen nicht gefordert.

Einführung in die Softwaretechnik AB Softwaretechnik

Mehrstufige Zerlegung

- **Bei der mehrstufigen Zerlegung wird zunächst eine modulare Zerlegung erarbeitet.**
- **Zu große Module werden als Subsysteme aufgefaßt und nach denselben Kriterien in weitere Module zerlegt.**
- **Die Schnittstellen der Subsysteme werden auf die Schnittstellen der enthaltenen Module zurückgeführt.**
- **Es entsteht eine zweite Beziehung: die Enthaltensein-Beziehung zwischen Modulen.**
- **Auf Programmebene entfallen die klammernden Subsysteme, es sei denn, die Programmiersprache bietet ein geschachteltes Modulkonzept.**

Einführung in die Softwaretechnik AB Softwaretechnik