Teilsysteme

Inhalt

1. Das Teilsystem als Bauelementecontainer
2. CEDL für die Beschreibung von Teilsystemen und Teilsystemstrukturen
3. CEDL für die Beschreibung von elementaren Softwarebauelementen
4. Teilsysteme und elementare Softwarebauelemente in C/C++
5. Teilsysteme in Java

Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Teilsystem (Begriff)

- Ein Teilsystem ist ein technisches Hilfsmittel zur Gruppierung, Verwaltung usw. von Softwarebauelementen (betrifft auch Teile) auf Modell- und Quellkodeniveau. Da ein *Softwarebauelement* ein Interface besitzt, welches die bereitgestellten Leistung beschreibt und dem Nutzer sichtbar ist und einen Implementationsteil, welcher dem Nutzer verborgen bleibt, muss das Teilsystem auch ein Interface und eine Implementation besitzen.
- Das Interface des Teilsystems kann Deklarationen für Softwarebauelemente verschiedenen Typs enthalten.
- Die Implementation des Teilsystems enthält die Implementation der Bauelemente.
- Eine Komposition von Teilsystemen ergibt wieder ein Teilsystem.
- Zwischen Teilsystemen treten folgende Typen von Beziehungen auf:
  - Teil_von-Beziehung (part_of),
  - Import-Beziehung (import).

Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Softwarebauelement

- Ein Softwarebauelement ist eine relativ selbstständige Programmeinheit für die Programmierung im Großen, dessen Verhalten über Schnittstellen beschrieben, separat gefertigt und wiederverwendet wird.
- Softwarebauelemente für die Programmierung im Kleinen sind z.B. Typen, Variablen, Prozeduren, Funktionen, Makros.
- Softwarebauelemente für die Programmierung im Großen (komposite Softwarebauelemente) sind Moduln, Klassen, verteilbare Objekte, usw.

Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Softwarebauelement (Fortführung)

- Durch Komposition werden sie unter Verwendung von Kompositionsregeln zu Softwaresystemen zusammengebaut.
- Die Kompositionsregeln werden durch Konnektoren (zeitinvariante Regeln) oder Werkzeuge/Tools (zeitvariante Regeln) verwirklicht.
- Softwarebauelemente können durch Beschreibungen modelliert werden.
- Ein Softwarebauelement hat einen Namen und existiert als Bestandteil eines Softwaresystems in einer Umgebung (Environment), die durch andere Softwarebauelemente und die virtuelle Maschine gebildet wird.

Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Schnittstelle
Eine Schnittstelle (synonym: Interface) ist der sichtbare Bereich eines Softwarebauelements. Ihre Beschreibung stellt die Nutzungs- und Anforderungsbedingungen für das Softwarebauelement dar.

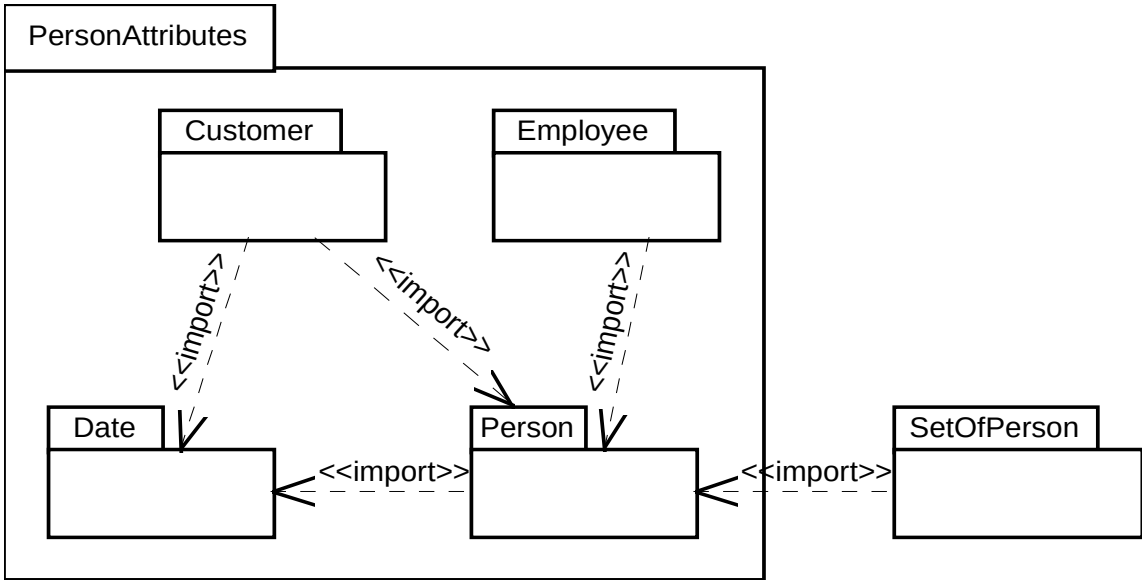
Implementation
Die Implementation ist die Abbildung von Softwarebauelementen auf Speicherobjekte.

Speicherobjekt
Ein Speicherobjekt ist ein Objekt, das für die Ablage auf einem Speicher entwickelt wurde.
Es liegt nach seiner Ablage auf einer Struktur von Speicherbereichen eines Datenträgers und hat einen Identifikator, unter dem es eindeutig angesprochen werden kann. Dieser Identifikator wird als Adresse bezeichnet.
Die Adressen werden in Adressräumen organisiert.
Ein Adressraum ist eine metrische Struktur.
Das Erzeugen eines Speicherobjekts kann statisch oder dynamisch durch eine (virtuelle) Maschine erfolgen.
Ein Speicherobjekt kann durch ein Softwarebauelement modelliert werden.

Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Teilsysteme in UML-Darstellung



Unified Modeling Language

Für die Modellierung u.a. von objektorientierten Systemen hat sich die Unified Modeling Language (UML) etabliert.

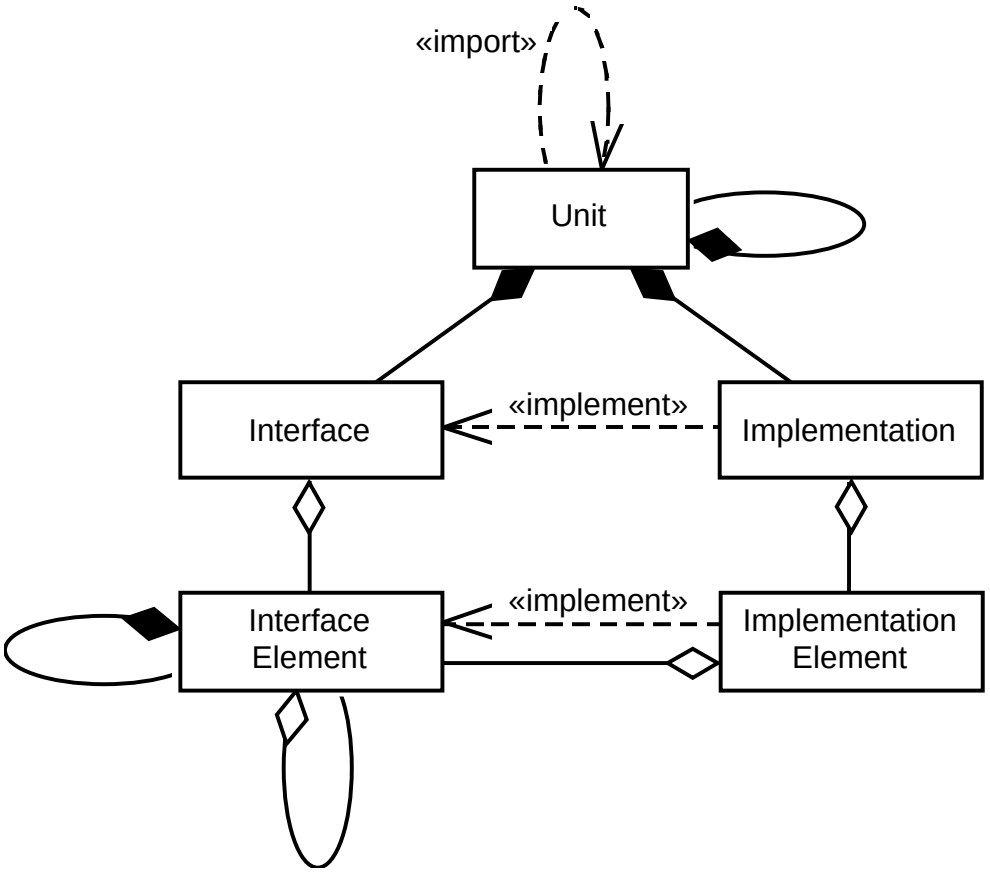
Die UML ist eine standardisierte vereinheitlichte Modellierungssprache für Software-Systeme.

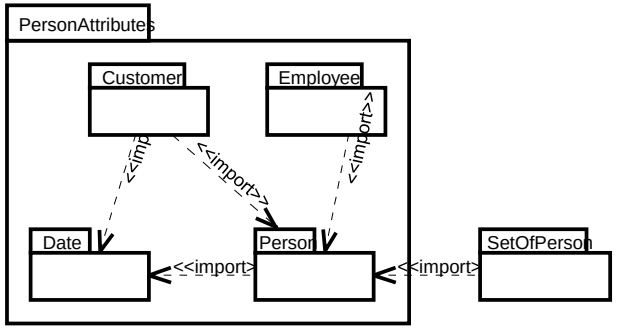
Die UML umfasst 13 Diagrammarten zur statischen und dynamischen Beschreibung von Software-Systemen.

Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Modell eines Teilsystems





Teilsysteme

1. Das Teilsystem als Bauelementecontainer

Grundformen der Benutztbeziehung zwischen den elementaren Softwarebauelementen:

- Elementare Bauelemente sind Typen, Variablen, Konstanten und Operationen.
- Es wird eine Variable oder Konstante eines *importierten Typs* deklariert.
  Die Benutzung erfolgt auf Ebene der Typen.
- Es wird eine *importierte Variable* oder Konstante (Datenobjekt)
  aus dem importierten Teilsystem benutzt.
- Es wird eine *importierte Operation*, d.h. ein Codeobjekt aus dem importierten Teilsystem aufgerufen

Teilsysteme

2. CEDL für die Beschreibung von Teilsystemen und Teilsystemstrukturen

CEDL

- CEDL ist eine Sprache zur Beschreibung von Softwarebauelementen (Construction Element Definition Language).
- Zentrale Aufgabe ist die *Spezifikation der Leistungen* eines Softwarebauelements und nicht von Implementationsdetails.

Teilsysteme

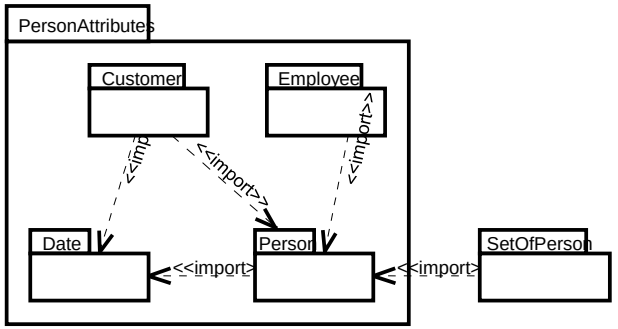
2. CEDL für die Beschreibung von Teilsystemen und Teilsystemstrukturen

CEDL Beispiel

```
unit PersonAttributes
  unit Date
    interface
     …
    interfaceend
    implementation
     …
    implementationend
  unitend
```

```
unit Person
    import Date;
     …
  unitend

  unit Customer
   import Date, Person;
   …
  unitend

  unit Employee
    import Person;
     …
  unitend
unitend
```

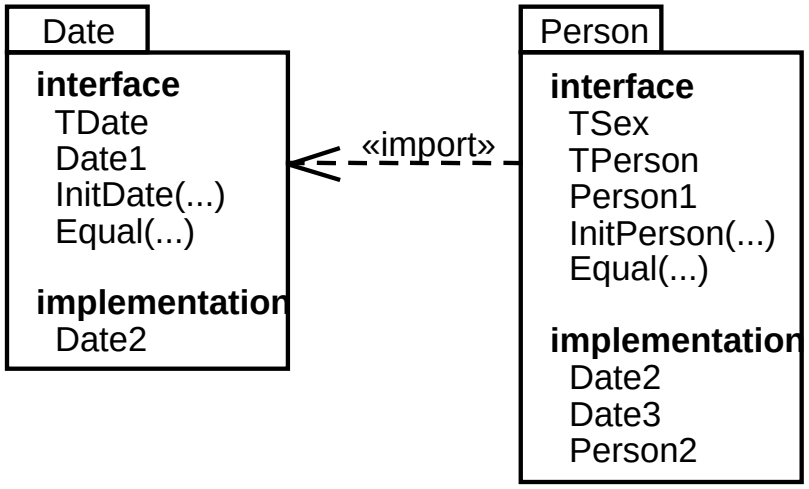
```
unit SetOfPerson
  import PersonAttributes.Person
…
unitend
```



Teilsysteme

3. CEDL für die Beschreibung von elementaren Softwarebauelementen

Beispiel

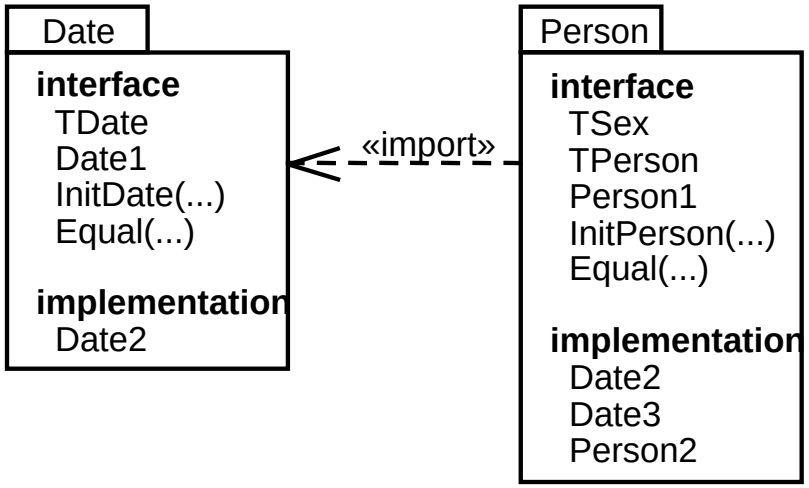



```
unit Date
 interface
   types
     TDate = record
       Day: Ordinal,
       Month: Ordinal,
       Year: Ordinal
     recordend;
   variables
      Date1: TDate;
   operations
      InitDate(inout dat: Tdate, in da: Ordinal, in mo: Ordinal, in ye: Ordinal),
      // Initialize a date
      Equal(in dat1: TDate, in dat2: TDate, out res: Boolean);
      // Compares two dates
 interfaceend
```

Teilsysteme

3. CEDL für die Beschreibung von elementaren Softwarebauelementen

Beispiel




```
implementation
  variables
    Date2: TDate;
  operation
    InitDate(inout dat: Tdate, in da: Ordinal, in mo: Ordinal, in ye: Ordinal)
    ...
  operationend;
  operation
    Equal(in dat1: TDate, in dat2: TDate, out res: Boolean)
    ...
  operationend;
implementationend
```

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Besonderheiten von C/C++ (part. Wiederholung):

- Es gibt keine Schlüsselworte für die Einleitung einer Variablendeklaration.
  Die Deklaration erfolgt durch:
  ```
  tname vname;
  ```
- Operationen werden in C/C++ als Funktionen oder Operatoren deklariert.
  Der Typ des Rückgabewertes der Funktion wird am Anfang der Deklaration notiert:
  ```
  tname oname(...);
  tname& operator =(const tname &vname);
  ```
- Hat die Operation keinen Rückgabewert, wird `void` notiert.
- Outputparameter haben den Typ Zeiger `tname*` oder Referenz `tname&`.
- Inputparameter können explizit mit **`const`** als nicht veränderbar qualifiziert werden.

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Interpretation der Beschreibungsmittel von CEDL für C/C++

CEDL		C/C++
Elementare Softwarebauelemente		
constants	⇒	**const**
types	⇒	**typedef**
variables vname : tname;	⇒	tname vname;
oname(...)	⇒	... oname(...)
in parname: tname	⇒	tname parname, const parname ...
out parname: tname	⇒	tname* parname tname& parname
inout parname: tname	⇒	tname* parname tname& parname

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Interpretation der Beschreibungsmittel von CEDL für C/C++

CEDL		C/C++
Elementare Softwarebauelemente		
Interface		
`unit uname`	⇒	Headerfile `uname.h`
`interface`		
`import iname, ...;`	⇒	`#include <iname.h>, <iname> oder #include "iname.h"` für jeden Import existiert eine include-Anweisung
`...`		`...`
`gname: tname ...`	⇒	`extern tname gname ...`
`...`		`...`
`interfaceend`		

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Interpretation der Beschreibungsmittel von CEDL für C/C++

Besonderheiten von C/C++

Die Präcompileranweisungen

```
#ifndef __DATE_H__
#define __DATE_H__
...

#endif
```

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

C++-Beschreibungen elementarer Bauelemente

Interface der Unit Date

```
#ifndef __DATE_H__
#define __DATE_H__

typedef struct{

  unsigned int Day;
  unsigned int Month;
  unsigned int Year;
} TDate;

extern TDate Date1;

void InitDate(TDate*,unsigned int,unsigned int,unsigned int);
// Initializes the attributes of a date.


bool Equal(TDate, TDate);
// Compares two dates. Returns true, if the attributes
// of the dates are equal, else return false.

#endif
```

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

C++-Beschreibungen elementarer Bauelemente

Implementation der Unit Date

```
...
#include "Date.h"

TDate Date1 = {26, 11, 2000};
static TDate Date2;

void InitDate(TDate *pdat, unsigned int da,
        unsigned int mo, unsigned int ye) {
  pdat -> Day = da;
  pdat -> Month = mo;
  pdat -> Year = ye;
}

bool Equal(TDate dat1, TDate dat2) {
  return( (dat1.Day  == dat2.Day) &&
          (dat1.Month == dat2.Month) &&
          (dat1.Year == dat2.Year) );
}
```

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

C++-Beschreibungen elementarer Bauelemente

Interface der Unit Person

```
...
#include "Date.h"        //import interface of unit Date

enum TSex {m, f};

typedef struct  {
  char*   FirstName;
  char*   Name;
  char*   Address;
  TDate   Birthday;
  TSex    Sex;
} TPerson;

extern TPerson Person1;
void InitPerson(TPerson*, const char*, const char*,
                const char*, const TDate, enum TSex);
bool Equal(TPerson, TPerson);

#endif
```

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

C++-Beschreibungen elementarer Bauelemente

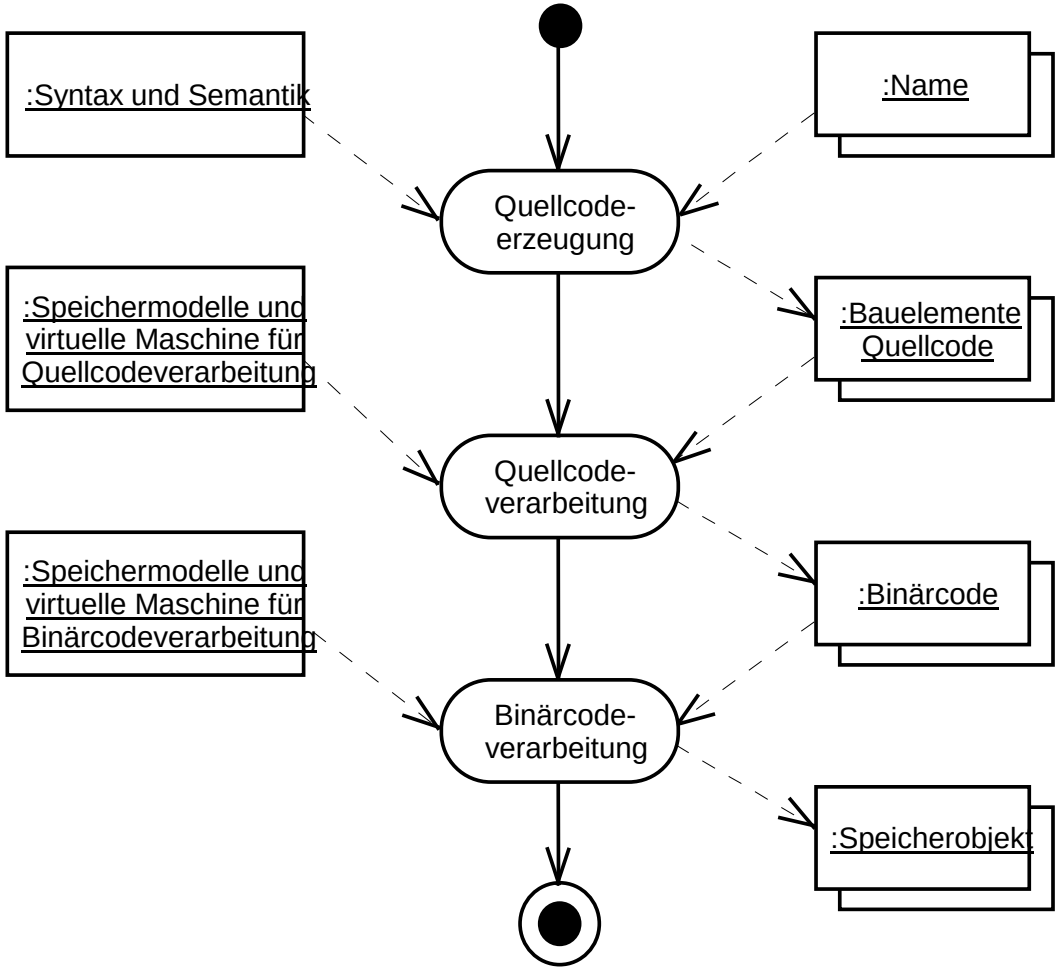
Implementation der Unit Person

```
…
#include "Person.h"

TPerson Person1;

static TDate   Date2, Date3;
static TPerson Person2;

void InitPerson(TPerson* pers, const char* fn, const char*
na,
        const char* ad, const TDate dbd, enum TSex se) {
       …;
}

bool Equal(TPerson pers1, TPerson pers2)  {
       …;
}
```

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Aktivitätsdiagramm der Codeverarbeitung (Verarbeitungsmodell)



Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Glossar

Quellcodeerzeugung

Umfasst die Aufschreibung des Quellcodes für Softwarebauelemente unter Berücksichtigung der Syntax und Semantik von Programmiersprachen und der Verwendung von Namen.

Quellcodeverarbeitung

Beschreibt die Verarbeitung des Quellcodes von Softwarebauelementen und die Erzeugung eines auf einer virtuellen Maschine interpretierbaren Binärcodes.

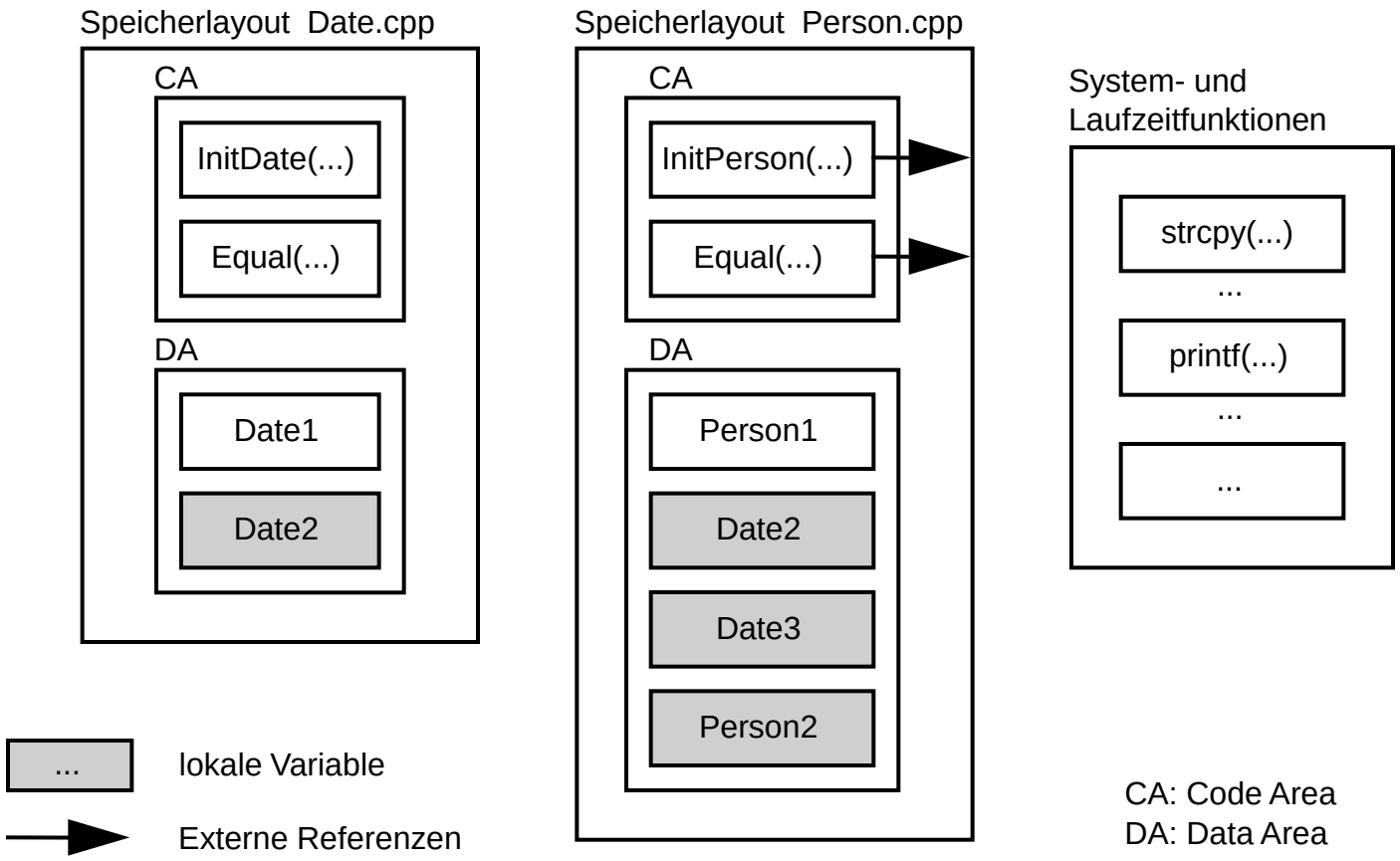
Binärcodeverarbeitung

Beschreibt die Interpretation des Binärcodes, das Anlegen, Benutzen und Vernichten von Speicherobjekten. Dabei werden Speichermodelle und Vorstellungen über eine virtuelle Maschine der Binärcodeverarbeitung verwendet.

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Speichermodelle nach der Compilation



Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

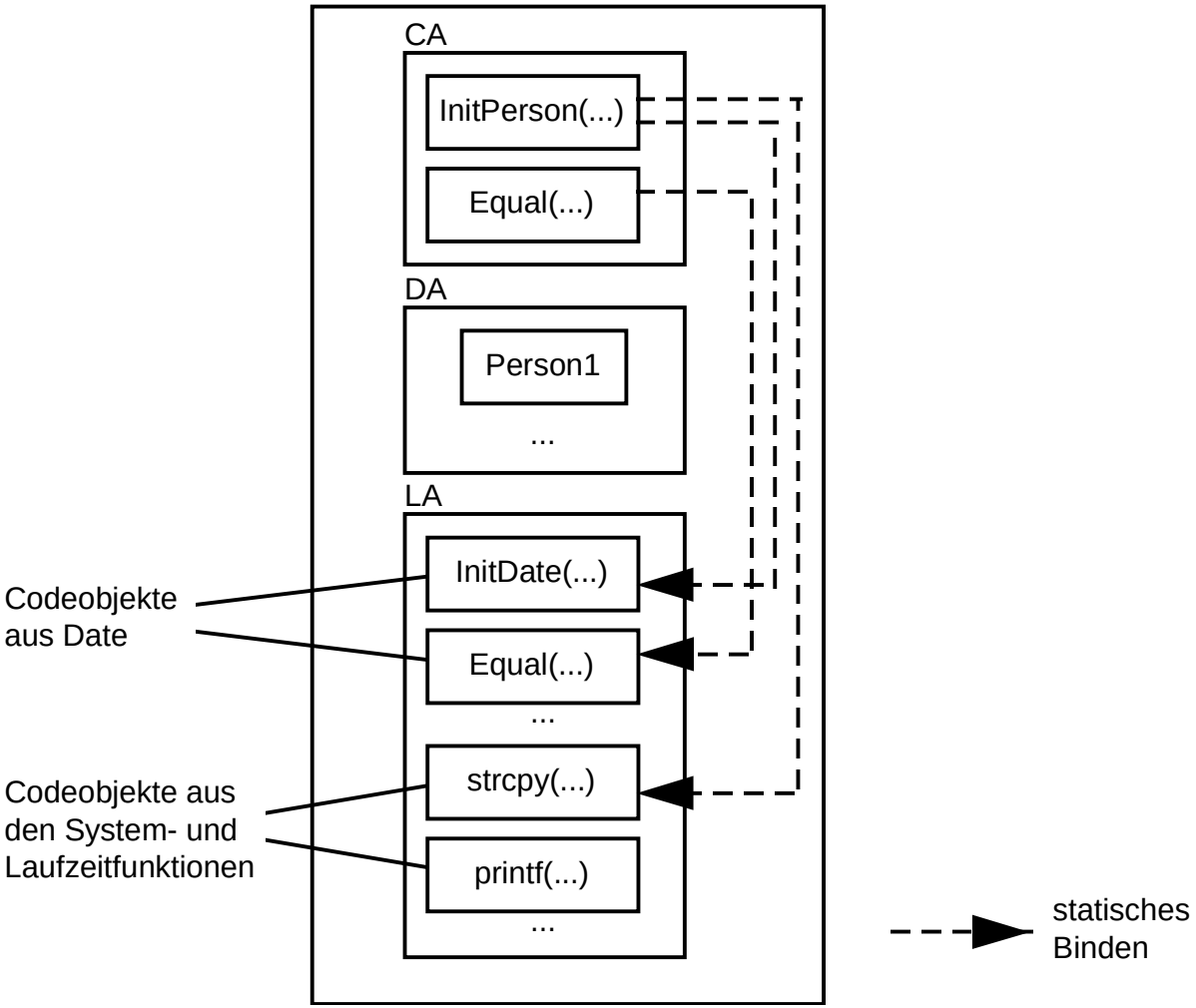
Speichermodelle nach der Compilation

Speichermodell
Ein Speichermodell beschreibt, aus welchen Teilen ein Speicher besteht, welche Beziehungen zwischen diesen Teilen existieren und wie und wo das Anlegen und Vernichten von und der Zugriff auf Speicherobjekte erfolgt.

Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

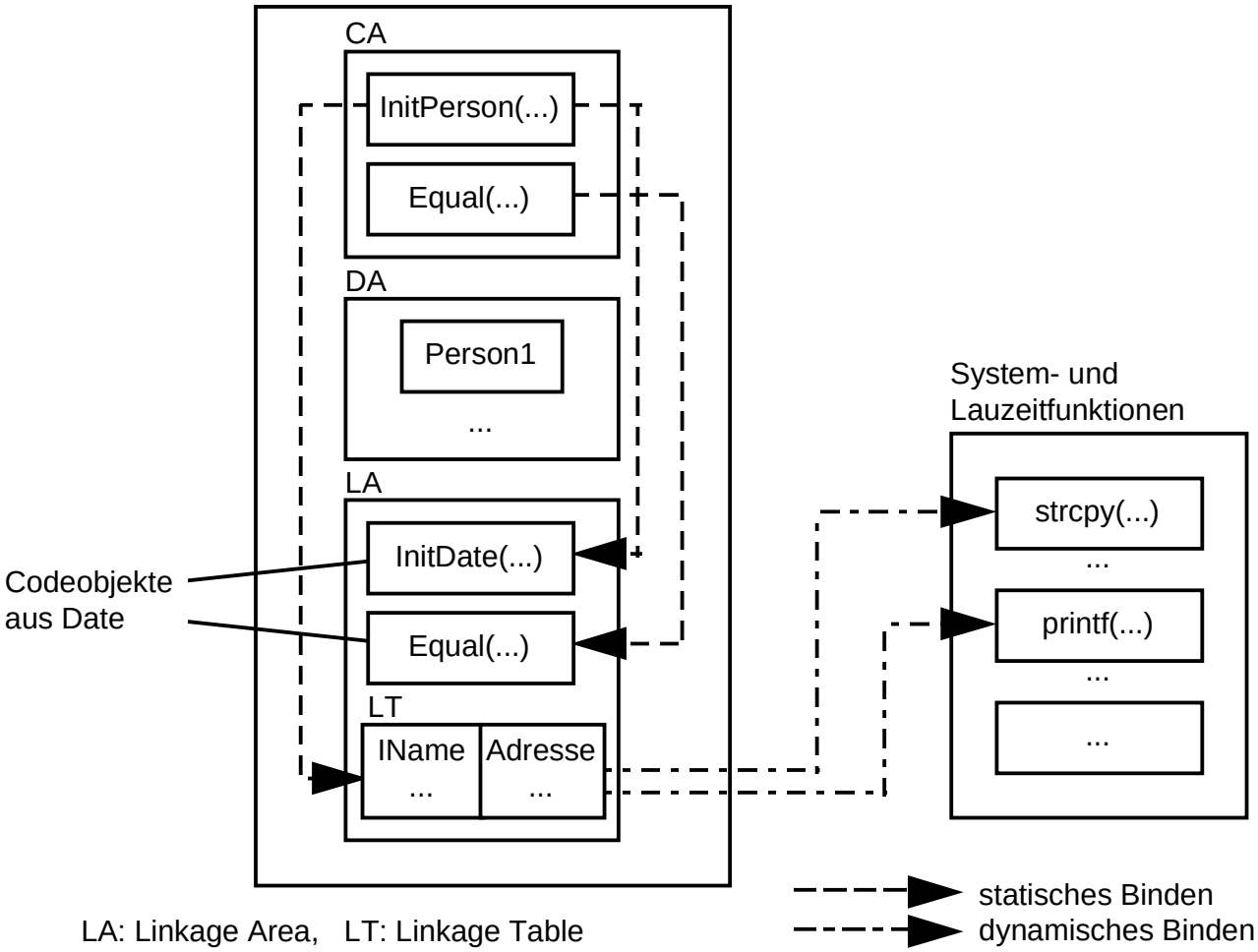
Verarbeitungsmodell für statisches Binden



Teilsysteme

4. Teilsysteme und elementare Softwarebauelemente in C/C++

Verarbeitungsmodell für dynamisches Binden



Teilsysteme

4. Teilsysteme in Java

CEDL		Java

unit uname
  import iname, ...;
  interface
  ...
  interfaceend
 	
  implementation
  ...
  implementationend
unitend
	⇒	**package** uname;
		  import iname;
		
		für jeden Import existiert eine Anweisung
		
		...

Teilsysteme

4. Teilsysteme in Java

Notationsformen für den Import

```
import Date.*;
import Date.Systemdate.CSystemdate;
```

Packagestruktur