

# Universität Stuttgart

## Fakultät Informatik

Prüfer: Prof. Dr.-Ing. U. G. Baitinger  
Betreuer: Dr. rer. nat. Michael Ryba

Beginn am: 1. Juli 1996  
Beendet am: 2. Januar 1997

CR-Klassifikation D.1.5, D.2.2, D.2.7, J.6

Studienarbeit Nr. 1579

### Optimierung und Erweiterung einer Klassenbibliothek zur Verarbeitung von Zeit- und Datumswerten

Johannes Fasolt

**Institut für Parallele und Verteilte  
Höchstleistungsrechner**

Breitwiesenstraße 20-22

D-70565 Stuttgart



# Kurzfassung

Die Studienarbeit befaßt sich mit der Optimierung und Erweiterung einer in EIFFEL realisierten Klassenbibliothek zur Verarbeitung von Zeit- und Datumswerten.

Folgende Teilaufgaben mußten bearbeitet werden:

- Optimierung der Speicherdarstellung von Zeit- und Datumswerten
- automatische Berechnung von Feiertagen
- Erweiterung der Arbeitszeitarithmetik
- Entwicklung eines flexiblen Konzepts zur Ein- und Ausgabe formatierter Zeit- und Datumswerte

Neben der Verbesserung der Effizienz und der Erweiterung der Funktionalität war die Verallgemeinerung der Anwendungsmöglichkeiten Ziel der Arbeit.

Ausgehend von einer grundlegenden Analyse des Gebiets der Zeitarithmetik wurde ein überarbeiteter Entwurf der Basismechanismen angefertigt. Die bestehende Dreiteilung in Zeitpunkte, Zeitdauern und Zeitintervalle wurde im wesentlichen übernommen.

Die automatische Berechnung von Feiertagen macht vor dem Hintergrund zunehmender Internationalisierung eine Vergleichbarkeit von Datumsangaben verschiedener Kalendersysteme notwendig. Deswegen wurde das allgemeine Konzept des Kalenders, der den fortlaufenden Strom von Julianischen Tagen (nach Scaliger) nach astronomischen Beobachtungen einteilt, abgebildet.

Um der örtlich begrenzten Gültigkeit von Feiertagen gerecht zu werden, geschieht die Verwaltung von Feiertagen in einzelnen Mengen, die mit der Vereinigungsoperation zusammengefaßt werden können, um ein Kalendarium zu erzeugen.

Für die flexible Berechnung von Arbeitszeit wurde das Konzept des abstrakten Arbeitszeitmodells entwickelt. Die Konkretisierung in einem speziellen Modell beschreibt den verwendeten Algorithmus für bestimmte Abfolgen von Arbeitszeiten und berechnet diese für einen gewünschten Zeitraum.

Die Formatierung von Zeit- und Datumswerten beruht auf der Konstruktion von sogenannten Scannern, die als Listen von primitiven syntaktischen Objekten (Terminalen) realisiert sind. Bei einem Umwandlungsvorgang einer Zeichenketten in die interne Repräsentation prüfen alle Scanner eines Objekts nacheinander die Zeichenkette, bis diese als gültige Eingabe erkannt wurde. Das Objekt setzt im Erfolgsfall seine Daten entsprechend, andernfalls wird die Eingabe zurückgewiesen.



# Inhalt

<b>1</b>	<b>Einleitung</b>	<b>11</b>
1.1	Aufgabenstellung .....	11
1.2	PLASMA .....	11
1.2.1	Schichtenmodell .....	12
1.2.2	Einbettung der Komponente Zeitarithmetik .....	12
1.3	Ziele der Arbeit .....	13
1.4	Kapitelübersicht .....	13
<b>2</b>	<b>Methodische Grundlagen</b>	<b>15</b>
2.1	Verwendete Techniken .....	15
2.1.1	Booch-Methodik, Makro-Entwurf .....	15
2.1.2	Booch-Methodik, Mikro-Entwurf .....	16
2.1.3	Grafische Notation: Klassendiagramme nach Booch .....	16
2.1.4	Interaktionsdiagramm für Anwendungsfälle, Szenarien .....	17
2.1.5	Implementierung in EIFFEL .....	18
<b>3</b>	<b>Zeitbegriff</b>	<b>21</b>
3.1	Kalender .....	21
3.1.1	Grundlagen .....	22
3.1.2	Julianischer Kalender .....	22
3.1.3	Gregorianische Kalender .....	23
3.1.4	Julianischer Tag/Periode .....	23
3.1.5	Jüdischer Kalender .....	24
3.1.6	Mohammedanischer Kalender .....	24
3.2	Anforderungen an Zeitmessung .....	24
3.3	Zeitzone(n) .....	25
3.4	Sommerzeit .....	25
3.5	Terminologie .....	25
3.6	Zusammenfassung .....	27
<b>4</b>	<b>Ausgangssituation</b>	<b>29</b>
4.1	Bestehendes Klassensystem .....	29

4.2	Kritik .....	30
4.3	Stand der Technik .....	31
4.4	Zusammenfassung .....	32
<b>5</b>	<b>Optimierung der Speicherdarstellung</b>	<b>33</b>
5.1	Effizienz .....	33
5.1.1	Speichereffizienz .....	33
5.1.2	Geschwindigkeit des Zugriffs .....	33
5.1.3	Vermeidung von Redundanz .....	34
5.2	Ausgangssituation .....	34
5.3	Alternative Implementierungen .....	34
5.3.1	Speicherung in einem Basistyp pro Komponente .....	34
5.3.2	Codierung als Bitfolge .....	34
5.3.3	Zusammenfassung von Komponenten in Basistypen .....	35
5.3.4	Speicherung als Julianischer Tag .....	35
5.4	Ergebnis .....	36
<b>6</b>	<b>Ein- und Ausgabe von Datumswerten</b>	<b>37</b>
6.1	Grundlagen .....	37
6.1.1	Nationale Darstellungen von Zeit und Datum .....	38
6.1.2	ISO 8601, ein internationaler Standard .....	38
6.2	Analyse .....	40
6.2.1	Motivation .....	40
6.2.2	Anwendungsfälle .....	41
6.2.3	Anforderungen .....	41
6.2.4	Bekanntetechniken .....	42
6.3	Entwurf .....	43
6.3.1	Flexibilisierungskonzept .....	43
6.4	Ergebnis .....	43
<b>7</b>	<b>Automatische Berechnung von Feiertagen</b>	<b>45</b>
7.1	Grundlagen .....	45
7.1.1	Feiertage .....	45
7.1.2	Arbeitsfreie Tage .....	48
7.1.3	Betriebliche Ereignisse .....	49
7.1.4	Sonstige .....	50

7.2	Analyse .....	50
7.2.1	Charakterisierung .....	50
7.2.2	Anforderungen .....	52
7.2.3	Zusammenfassung .....	52
7.3	Entwurf .....	52
7.3.1	Mengenmodell .....	52
7.3.2	Attributierung .....	52
7.3.3	Spezialisierung bei Ereignissen .....	53
7.3.4	Abhängigkeitsmechanismus .....	53
7.3.5	Dynamische Bereichserweiterung .....	54
7.3.6	Datenstruktur für effiziente Abfragen .....	54
7.4	Ergebnis .....	55
<b>8</b>	<b>Erweiterung der Arbeitszeitarithmetik</b>	<b>57</b>
8.1	Grundlagen .....	57
8.1.1	Definitionen .....	57
8.1.2	Statisches Arbeitszeitmodell .....	58
8.1.3	Dynamische Arbeitszeitmodelle .....	58
8.1.4	Recht .....	59
8.2	Analyse der Anforderungen .....	60
8.3	Ergebnis .....	61
<b>9</b>	<b>Resultierendes Klassensystem</b>	<b>63</b>
9.1	Übersicht .....	63
9.2	Klassenkategorien .....	64
9.2.1	Klassenkategorie für Zeit und Datum .....	64
9.2.2	Klassenkategorie für Datumseignisse .....	67
9.2.3	Klassenkategorie für Arbeitszeitarithmetik .....	69
9.2.4	Klassenkategorie für Formatierung .....	70
<b>10</b>	<b>Zusammenfassung</b>	<b>73</b>
<b>A</b>	<b>Feiertage</b>	<b>75</b>
	<b>Literatur</b>	<b>79</b>

<b>Internet-Referenzen</b>	<b>83</b>
<b>Glossar</b>	<b>85</b>

# Abbildungsverzeichnis

Abb. 1	Schema eines offenen integrierten Entwurfssystems	12
Abb. 2	Booch-Notation für Klassenhierarchien	17
Abb. 3	Interaktionsdiagramm	18
Abb. 4	Größenordnungen der Zeit (schematisch)	25
Abb. 5	Primäre Architektur, Kernbeziehungen	29
Abb. 6	Binär-Codierung mehrerer INTEGER-Werte in eine einzige Variable	35
Abb. 7	Der Algorithmus von Gauss in EIFFEL [CFD88]	47
Abb. 8	Durchschnittlicher Anteil der Arbeitstage im Jahr 1996	49
Abb. 9	Benachrichtigungsschema bei Änderungen an Datumseignissen	53
Abb. 10	Datenstruktur für eine Liste von Datumseignissen	54
Abb. 11	Aufbau eines Arbeitszeitplans	61
Abb. 12	Hierarchie der Klassenkategorien	63
Abb. 13	Klassen in TIME_GENERAL	65
Abb. 14	Klassen in TIME_DATE_EVENTS	67
Abb. 15	Klassen in TIME_WORKTIME	69
Abb. 16	Klassen in TIME_SCANNING	71



## 1.1 Aufgabenstellung

Im Rahmen des Projekts „PLASMA - Planung und Steuerung methodischer Entwurfsaktivitäten“ [Ryba96] sollte eine existierende Klassenbibliothek zur Verarbeitung von Zeit- und Datumswerten überarbeitet, optimiert und erweitert werden.

Die Arbeit umfaßt die folgenden Teilaufgaben:

- Optimierung der Speicherdarstellung von Zeit- und Datumswerten
- automatische Berechnung von Feiertagen
- Erweiterung der Arbeitszeitarithmetik
- Entwicklung eines flexiblen Konzepts zur Ein- und Ausgabe formatierter Zeit- und Datumswerte

Die Implementierung erfolgte in der Programmiersprache EIFFEL 3 auf SUN Workstations.

Die zu erstellende Bibliothek sollte universell genug gehalten werden, um auch außerhalb von PLASMA verwendbar zu sein.

## 1.2 PLASMA

Der Entwurf von Systemen aller Art, insbesondere der Entwurf von VLSI-Schaltungen, findet oftmals in heterogenen Softwareumgebungen statt. Integrierte Entwicklungsumgebungen werden durch Spezialwerkzeuge ergänzt. Die dabei auftretenden Probleme an den Schnittstellen senken die Qualität des Entwurfsprozesses und damit auch die des Ergebnisses.

Ein Lösungsansatz ist die Schaffung offener, integrierter Systeme. Offenheit entsteht aus der Möglichkeit, das Entwurfssystem um zusätzliche Komponenten (Werkzeuge) zu erweitern. Integration ist ein Ergebnis der Auslagerung werkzeugunabhängiger Systemteile in ein übergreifendes Schichtenmodell. Auf diese Weise werden Redundanzen vermieden und zwingende Schnittstellen zwischen den Werkzeugen geschaffen.

### 1.2.1 Schichtenmodell

Basis einer effizienten Zusammenarbeit verschiedener Systemkomponenten ist der geregelte Zugriff auf sämtliche Stamm- und Projektdaten. Die Speicherung soll redundanzfrei, konsistent und dauerhaft sein. Diese Aufgabe übernimmt eine Komponente zur Datenverwaltung.

Alle Werkzeuge greifen über gemeinsame Datenschemata auf ein Anwendungsdatenmodell zu.

Neben den Werkzeugen ist eine Komponente zur Steuerung des Entwurfsablauf angeordnet.

Eine Komponente für Basisdienste wie Interprozesskommunikation und Netzwerkdienste wird von allen darüber liegenden Schichten genutzt.

Oberste Schicht ist die Benutzungsoberfläche. Hier wird eine einheitliche Bedienung garantiert.

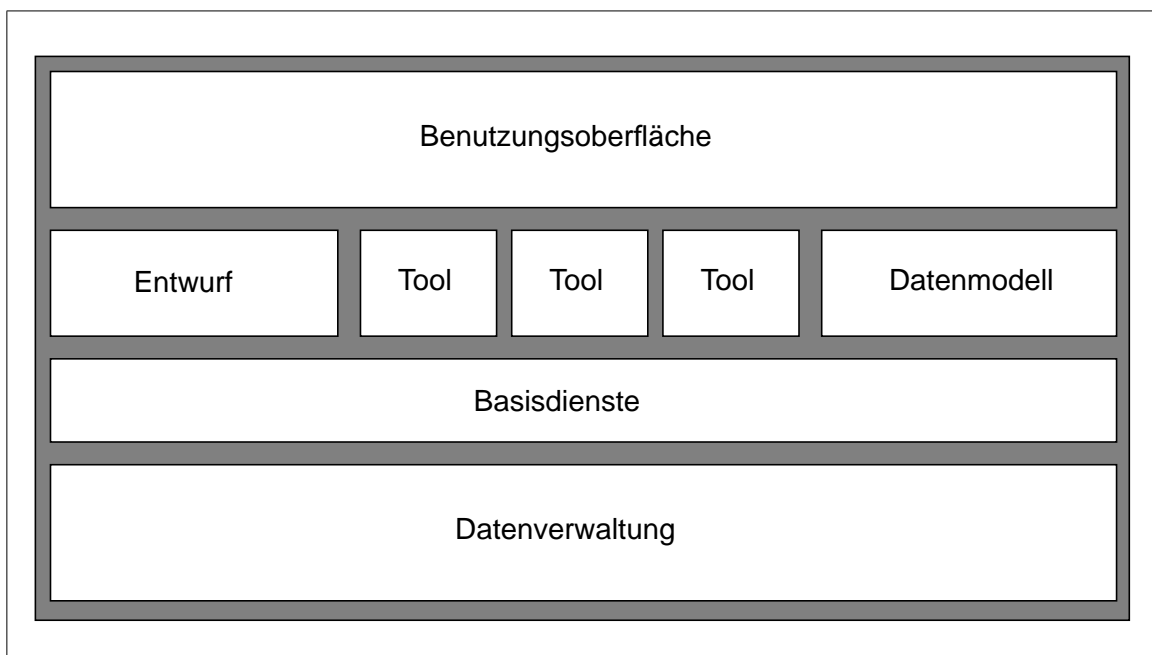


Abb. 1: Schema eines offenen integrierten Entwurfssystems

### 1.2.2 Einbettung der Komponente Zeitarithmetik

Die Einbeziehung vorhandener Ressourcen in die Planung eines Entwurfsprozesses ist eine fundamentale Aufgabe. Nach einer generellen Aufteilung in Hardware, Software, Humanressourcen und Daten erfolgt eine weitere Klassifikation gemäß ihrer Rolle im Entwurfsprozeß.

In einem konkreten Entwurfsprojekt werden nach der Festlegung des Arbeitsflusses, d.h. der Festlegung der Teilaufgaben und deren zeitlicher Abfolge, und der Zeitplanung die Ressourcen für jeden Vorgang, unter Berücksichtigung von Abhängigkeiten und Konflikten, eingeplant.

Ressourcenbezogene Daten sind werkzeugübergreifend, wenn nicht sogar projektübergreifend einzuordnen. Sie müssen deshalb dauerhaft gespeichert und im Anwendungsdatenmodell abgebildet werden.

## 1.3 Ziele der Arbeit

### **Erweiterung der Funktionalität für PLASMA**

Zusätzliche Anforderungen im Bereich Zeitarithmetik in PLASMA machen eine Überarbeitung der bisher verwendeten Bibliothek notwendig.

### **Erfassung und Darstellung des Anwendungsgebiets**

Um die erwähnten Erweiterungen durchzuführen, muß das Anwendungsgebiet vollständig analysiert und dargestellt sein. Notwendige Änderungen an der Architektur können durchgeführt werden, da PLASMA angepaßt werden kann und eine Verwendung der Bibliothek in anderen Projekten noch nicht existiert.

### **Verbessern der Effizienz**

Da ein Teil der bestehenden Bibliothek übernommen werden kann, müssen deren Speicherungsstrukturen hinsichtlich der Effizienz überprüft werden.

### **Verallgemeinerung der Anwendungsmöglichkeiten**

Die Bibliothek soll für andere Anwendungsgebiete als die Projektplanung ebenfalls verwendbar sein.

## 1.4 Kapitelübersicht

Als nächstes Kapitel folgt eine Darstellung der methodischen Grundlagen der durchgeführten Arbeit.

Darauf folgt in Kapitel 3 eine allgemeine Einführung zum Thema Zeit mit einer Begriffsklärung und einer Hinführung auf die Basisarchitektur. In Kapitel 4 folgt eine kritische Analyse der Ausgangssituation.

Die vier Teilaufgaben aus der Aufgabenstellung werden getrennt nacheinander erläutert, wenn auch die Bearbeitung parallel durchgeführt wurde. Zunächst befaßt sich Kapitel 5 mit der Optimierung der Speicherdarstellung von Zeit- und Datumswerten.

Darauf folgt Kapitel 6 mit einem flexiblen Konzept zur Ein- und Ausgabe. In Kapitel 7 wird das Konzept zur Berechnung von Feiertagen vorgestellt. Kapitel 8 beinhaltet den Bereich Arbeitszeitarithmetik.

Darauf folgt eine Darstellung des resultierenden Klassensystems. Kapitel 9 stellt die Gesamtarchitektur im Überblick vor.

Den Schluß der Arbeit bilden die Zusammenfassung der vorliegenden Arbeit und ein Ausblick auf Erweiterungen und Anwendungen.

# 2

## Methodische Grundlagen

Die Grundlagen objektorientierter Programmierung werden als bekannt vorausgesetzt. Von den verwendeten objektorientierten Mechanismen sind abstrakte Basisklassen, Polymorphismus, Mehrfachvererbung und generische Klassen besonders zu erwähnen.

Großes Augenmerk wurde auf die Modularisierung gelegt. Hier ist im wesentlichen die Zerlegung in Teilsysteme zur Reduzierung der Komplexität und zur Erhöhung der Verständlichkeit gemeint. Klassen besitzen möglichst schmale Schnittstellen um eine lose Kopplung zu fördern.

### 2.1 Verwendete Techniken

Die vorliegende Arbeit wurde nach Prinzipien der objektorientierten Softwareentwicklung von Grady Booch [Boo94] durchgeführt.

#### 2.1.1 Booch-Methodik, Makro-Entwurf

Der erste Schritt besteht in dem Herausfiltern von Kernanforderungen. Die Analyse von Anwendungsfällen (use cases) ergibt ein Anforderungsprofil an das zu erstellende Klassensystem. Daraus folgt das Entwickeln eines Modells über den Anwendungsbereich. Beim Erzeugen einer Architektur werden bekannte Techniken untersucht und Entwurfalternativen abgewogen. Auf die Entwurfentscheidung folgt das Entwickeln der Implementierung. Hier wird die Methode der schrittweisen Verfeinerung angewendet.

Ziel ist ein hoher Grad an Wiederverwendung auch durch den Einsatz von Entwurfsmustern [Gam95].

## 2.1.2 Booch-Methodik, Mikro-Entwurf

### Identifizieren von Klassen und Objekten

Ausgangspunkt ist eine Analyse des Anwendungsgebiets mit dem Ziel, die Terminologie zu verstehen und wesentliche Begriffe zu isolieren. Diese Begriffshierarchie ist weitgehend äquivalent zur entstehenden Klassenhierarchie. Zuerst werden Begriffe gesammelt und katalogisiert. Es werden Synonyme und Homonyme erfaßt und eindeutig definiert.

Aus der entstandenen Menge werden die relevanten Begriffe zu Klassen gewandelt. Bei diesem Schritt werden die Grenzen des Problems abgesteckt.

Bei der Zerlegung der Gesamtaufgabe in Teilprobleme entstehen Klassenkategorien, d.h. Klassen werden zu sinnverwandten Gruppen zusammengefaßt.

### Identifizieren der Semantik von Klassen und Objekten

Eine genauere Untersuchung der Begriffe liefert das Verhalten und die Attribute der Klassen. Es findet eine Verfeinerung der Analyse statt. Hierbei werden auch einzelne Verantwortlichkeiten auf die Klassen verteilt.

### Identifizieren von Beziehungen zwischen Klassen und Objekten

Hier wird die Architektur des Systems gebildet. Es entstehen Zugriffspfade zu den Entitäten.

Zuerst werden Assoziationen zwischen den Klassen definiert. Ziel ist, jedes Teilsystem durch einen zusammenhängenden Graphen zu beschreiben. Die Kanten können zu Beginn ungerichtet sein und werden mit der Verfeinerung der Assoziationen, wenn Vererbungs- und Benutzungsbeziehungen eingeführt werden, gerichtet.

### Implementierung von Klassen und Objekten

Bei der Implementierung wird die innere Sicht einer Klasse festgelegt und die definierte Schnittstelle nur noch in Ausnahmefällen verändert. Die Wahl geeigneter Algorithmen und Datenstrukturen hat keinen Einfluß auf das äußere Verhalten einer Klasse.

## 2.1.3 Grafische Notation: Klassendiagramme nach Booch

Für die grafische Beschreibung der statischen Systemarchitektur werden Klassendiagramme nach Booch [Boo94] verwendet. Abb. 2 zeigt ein Beispiel mit Mehrfachvererbung und einer parametrisierten Klasse.

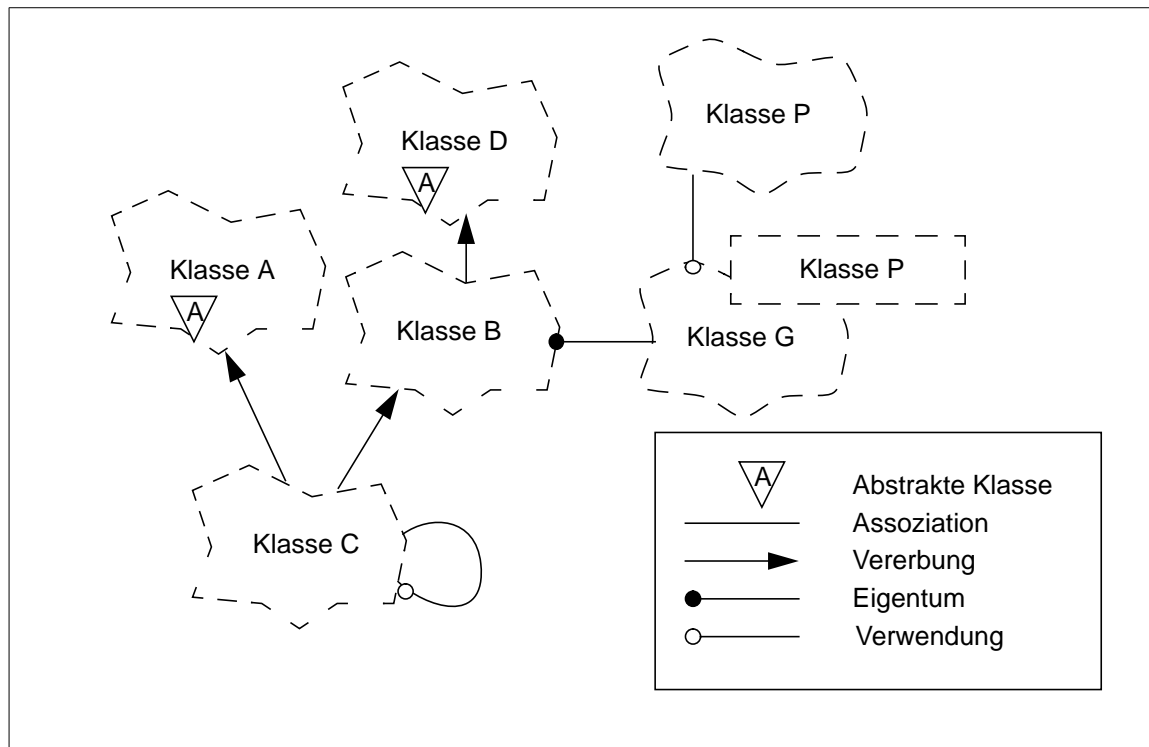


Abb. 2: Booch-Notation für Klassenhierarchien

Die wesentlichen Elemente dieser Notation sind Klassen und ihre Beziehungen zueinander. Klassen werden durch eine Wolke symbolisiert. In dieser Wolke steht der Name der Klasse. Abstrakte Klassen werden durch ein auf der Spitze stehendes Dreieck mit einem „A“ gekennzeichnet. Parametrisierte Klassen werden mit einem Klassennamen in einem rechteckigen Kasten erweitert.

Die wichtigsten Beziehungen zwischen Klassen sind Vererbung, Eigentum (Aggregation) und Verwendung. Sie werden während der Entwurfsphase teilweise nur als einfache Assoziationen dargestellt. Der Pfeil einer Vererbungsbeziehung verläuft von der erbenden Unterklasse zur Oberklasse. Die Eigentumsbeziehung wird durch eine Linie mit ausgefülltem Kreis am Ende der aggregierenden Klasse dargestellt, bei Verwendung ist dieser Kreis nicht ausgefüllt.

### 2.1.4 Interaktionsdiagramm für Anwendungsfälle, Szenarien

Bei größeren Szenarien wie eine Beschreibung von Transaktionen oder eines zeitlichen Ablaufs, werden Interaktionsdiagramme zur Darstellung des Verhaltens verwendet.

Senkrechte Linien symbolisieren den zeitlichen Ablauf einzelner Akteure (Objekte), die in einem bestimmten Zeitraum an einem solchen Szenario teilnehmen. Die Akteure sind waagrecht angeordnet.

Pfeile stellen Nachrichten zwischen einzelnen Akteuren dar.

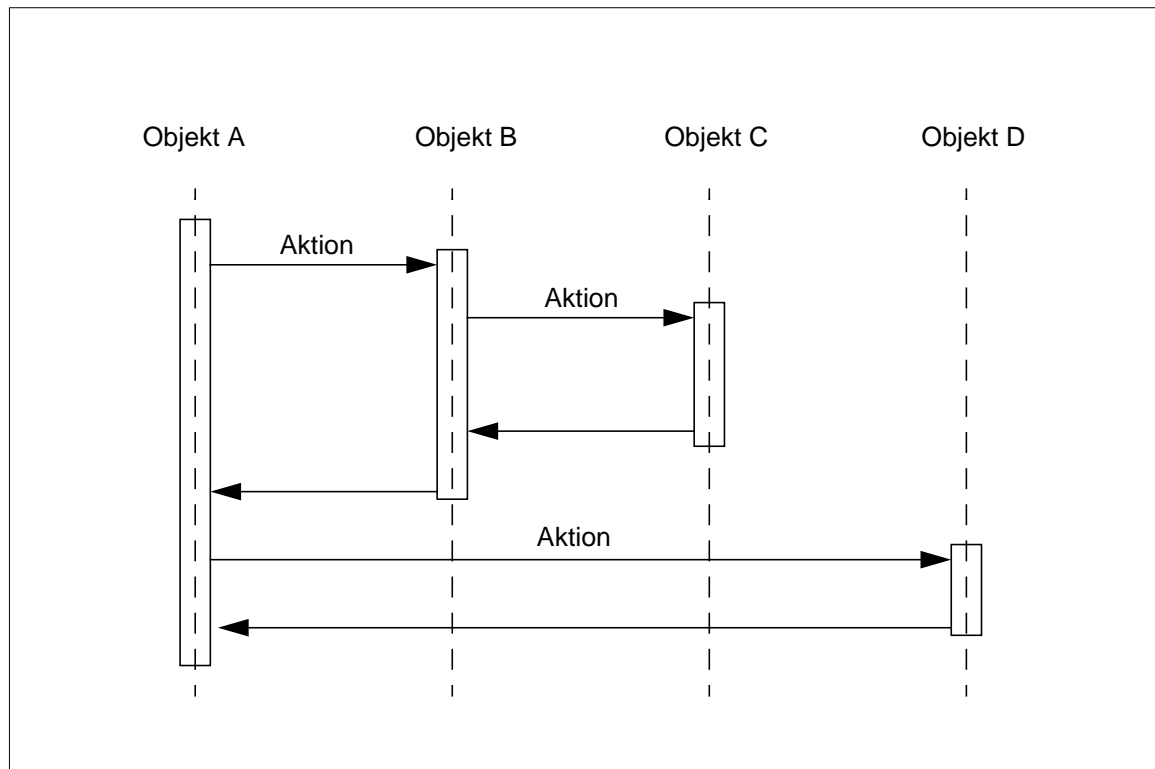


Abb. 3: Interaktionsdiagramm

Abb. 3 zeigt ein schematisches Interaktionsdiagramm.

### 2.1.5 Implementierung in Eiffel

Die verwendete Programmiersprache Eiffel ist objektorientiert und fördert durch die Einteilung einer Anwendung in Klassen und Klassenverbände (Cluster) Entwurfsziele wie Modularisierung und Wiederverwendung.

Das Prinzip des „*Programming by Contract*“ liefert schnell zuverlässige Programmteile, die zu kompletten Systemen kombiniert werden. Hierbei wird die Schnittstelle einer Klasse mit Vor- und Nachbedingungen für Methodenaufrufe und Klasseninvarianten versehen. Somit sind die möglichen Änderungen auf die gekapselten Daten und notwendige Bedingungen hinreichend beschreibbar, ohne die Implementierung preisgeben zu müssen.

Die Sprachdefinition beinhaltet folgende Grundbausteine:

- Mehrfachvererbung
- abstrakte Basisklassen
- Generizität (eingeschränkt und nicht eingeschränkt)
- statische Typisierung und dynamische Bindung

- Zusicherungen
- disziplinierte Ausnahmebehandlung
- automatische Speicherfreigabe (Garbage Collection)

Die verwendete Entwicklungsumgebung der Firma ISE (Interactive Software Engineering) in der Version 3 realisiert alle genannten Sprachelemente und stellt eine umfangreiche Bibliothek mit wichtigen Basis-Datenstrukturen zur Verfügung. Das Werkzeug integriert alle notwendigen Arbeitsschritte zur Programmerzeugung

Für zusätzliche Informationen sei im Besonderen auf Literatur von Bertrand Meyer, der Eiffel entworfen hat und heute mit ISE die Weiterentwicklung betreibt, verwiesen [Mey90, Mey92].



# 3

## Zeitbegriff

*„Zeit ist der Begriff des Aufeinanderfolgens.“*

Die Schwierigkeit bei der Beschreibung des Wesens der Zeit liegt darin, daß der Begriff Zeit ein fundamentaler Bestandteil unseres Kulturgutes ist, den wir ständig in unserem Kontext benutzen. Abstraktion bedeutet hier deshalb den Kulturkreis zu verlassen und global zu denken.

### 3.1 Kalender

Ein Kalender ist eine nach astronomischen Beobachtungen angelegte Einteilung der Zeit. Astronomie war historisch gesehen immer Bestandteil religiöser Forschung und wurde erst in jüngerer Zeit weltlich. Erste Kalender datieren 2770 v.Chr. in Ägypten. Der getrennten Entwicklung der Religionen waren auch die Kalender unterworfen. Hinzu kamen im Wandel der Geschichte Besonderheiten durch die Entwicklung der Nationalstaaten.

Eine technische Betrachtung der Zeit kann also nur innerhalb eines Kalendersystems gelten. Ein Abgleich der Kalender der Welt muß über physikalische oder kalendarische Bezugspunkte stattfinden. Kalendarische Einheiten sind Tage oder deren Vielfache. Die Definition eines Tages ist auf der ganzen Erde identisch: ein Tag entspricht der Dauer einer Drehung der Erde um ihre Längsachse, erfahrbar durch den Stand der Sonne. Zeitmessung geschieht durch das Zählen von Sekunden oder deren Bruchteile. Stellt man sich die Tage fortlaufend numeriert vor, ist klar, daß jeder Kalender deshalb ein Datum seiner Notation in einen Tag dieser monoton steigenden Ordnung konvertieren können muß, da ein Kalender für jeden Tag per Definition ein gültiges Datum berechnet.

Aufgrund der Dominanz der westlichen Wirtschaftsordnung hat sich international der gregorianische Kalender durchgesetzt und kann als Standard für Datumsangaben angesehen werden. Die Berechnung von Feiertagen und die Darstellung von Datumsangaben in landestypischer Form muß sich allerdings nach Konventionen richten, die durch Kalender des jeweiligen Kulturkreises bestimmt sind.

### 3.1.1 Grundlagen

Kalender teilen in der Regel das Jahr in Monate ein. Ein Monat entspricht der Dauer des Umlaufs des Mondes um die Erde. Als synodischer Monat ist die Dauer bis zur Wiederkehr der gleichen Mondphase definiert (29,53 Tage). Frühe Kalender teilten deshalb das Jahr in 6 Monate mit 29 Tagen und 6 Monate mit 30 Tagen ein. Das astronomische Jahr, die Dauer des Umlaufs der Erde um die Sonne, dauert 365,25636 Tage. Für die Jahreszeiten auf der Erde ist allerdings der Stand der Sonne zum Äquator maßgebend. Das tropische Jahr beschreibt mit 365,2422 Tagen die Dauer zwischen zwei Durchläufen durch den Frühlingspunkt (Äquinoktium: Tag und Nacht sind gleich lang, 21.3. und 23.9.). Es wird deutlich, daß die astronomischen Perioden keine ganzzahligen Vielfachen der für den Menschen wichtigsten Einheit Tag darstellen. Für die Praxis schien allerdings eine Parallelität zwischen Jahren und den Jahreszeiten von großer Wichtigkeit zu sein. Die Einführung von Schalttagen bzw. Schaltmonaten gleicht diese Unstimmigkeiten aus.

### 3.1.2 Julianischer Kalender

Julius Cäsar (100 v.Chr. bis 44 v.Chr.) bewirkte eine Kalenderreform im Jahr 46 v.Chr. Der nach ihm benannte julianische Kalender teilt das Jahr in 12 Monate ein, verzichtet aber auf die Abhängigkeit von Mondphasen. Da das Mondjahr 11 Tage kürzer als das Sonnenjahr ist, mußten die Monate eine Verlängerung erfahren. Es entstand folgende Einteilung:

- Januar (nach dem Gott Janus), 31 Tage
- Februar (nach dem Fest Februa), 28 bzw. 29 Tage, Schaltmonat
- März (nach dem Gott Mars), 31 Tage
- April (nach Aphrodite oder nach lat. aperire = öffnen), 30 Tage
- Mai (wahrscheinlich nach der Göttin Maia), 31 Tage
- Juni (wahrscheinlich nach der Göttin Juno), 30 Tage
- Juli (44 v.Chr. nach Julius Cäsar benannt), 31 Tage
- August (8 v.Chr. nach dem Eroberer Augustus benannt), 31 Tage
- September (lat. septem, der siebte Monat im alten röm. Kalender), 30 Tage
- Oktober (lat. octo, der achte Monat im alten röm. Kalender), 31 Tage
- November (lat. novem, der neunte Monat im alten röm. Kalender), 30 Tage
- Dezember (lat. decem, der zehnte Monat im alten röm. Kalender), 31 Tage

Die Einführung dieses Kalenders ist nicht direkt im Jahr 46 v.Chr. festzumachen. Eine endgültige Fixierung dauerte letztlich bis ins 4. Jahrhundert n.Chr.

Als Jahreslänge wurden 365,25 Tage gewählt, was die Einführung eines Schaltjahres zu Folge hatte. An allen durch 4 ohne Rest teilbaren Jahren wird im Februar ein zusätzlicher Tag eingefügt.

### 3.1.3 Gregorianische Kalender

Da die Jahreslänge mit 365,25 Tagen im Julianischen Kalender immer noch nicht der Länge des tropischen Jahres entsprach, führte Papst Gregor XIII im Jahr 1582 den im folgenden Gregorianisch genannten Kalender ein. Auf den 4. Oktober folgte in diesem Korrekturjahr direkt der 15. Oktober, um den bis dahin aufsummierten Fehler zu korrigieren. Von diesem Jahr an wurden alle durch 100 ohne Rest teilbaren Schaltjahre zu normalen Jahren (Säkularjahre), außer wenn das Jahr durch 400 teilbar ist. Der verbleibende Fehler ergibt erst nach ca. 3300 Jahren einen ganzen Tag, der dann ausgelassen werden müßte.

### 3.1.4 Julianischer Tag/Periode

Nicht zu verwechseln mit dem Julianischen Kalender ist die nach Julius Cäsar Scaliger (1484-1558) benannte, durch seinen Sohn Joseph Justus Scaliger (1540-1609) eingeführte Julianische Periode. Dessen Idee bestand darin, alle Tage ihrer Reihenfolge entsprechend aufsteigend durchzunummerieren. Die Julianische Periode beginnt am 1. Januar 4713 v. Chr. (nach Julianischem Kalender) um 12 Uhr und dauert, bedingt durch zugrundeliegende astronomische Konstanten, 7980 Jahre, danach beginnt die Zählung erneut von 1.

Astronomen benutzen den sogenannten Julianischen Tag um die Tage entsprechend zu benennen. Beispielsweise fällt der 1. Januar 2000 auf den Julianischen Tag 2.451.545, folgendermaßen berechnet:

$$6712 \text{ Jahre} * 365,25 \text{ Tage/Jahr} - 13 \text{ Tage Differenz zw. Julian. und Greg. Datum}$$

Oft werden auch gebrochene Julianische Tage verwendet, z.B. 2.451.545,125 für 15 Uhr an obigem Tag.

Eine etwas praktikablere Form ist die des Modifizierten Julianischen Tages (MJD). Diese Nummerierung beginnt 2.400.000,5 Tage später und bringt die Werte für die Gegenwart in einen numerisch besser handhabbaren Bereich. Außerdem wechselt die Nummerierung dann um 0 Uhr.

### 3.1.5 Jüdischer Kalender

Der jüdische Kalender kennt mangelhafte, regelmäßige und unregelmäßige Gemeinjahre mit 353,354,355 Tagen und entsprechend Schaltjahre mit 383,384,385 Tagen. Der Beginn jüdischer Zeitrechnung ist der 7.10.3761 v. Chr., laut dem Alten Testament der Zeitpunkt der Erschaffung der Welt.

Tagesbeginn ist um 18 Uhr, was bei der Umrechnung zwischen Kalendersystemen zu Problemen führen kann. Bei der Zuordnung zu einem Julianischen Tag wird der Name des jüdischen Tages verwendet, der vor 18 Uhr liegt.

Das jüdische Neujahrfest Roscha-Schana wird am am 1. und 2. Tischri (etwa September/Okttober) gefeiert.

### 3.1.6 Mohammedanischer Kalender

Der mohammedanische Kalender beginnt am 16. Juli 622 n.Chr. und definiert die Zeitrechnung seit der Flucht Mohammeds aus Mekka.

Er orientiert sich nur am Lauf des Mondes. Deswegen korrespondieren Monate direkt mit den Mondphasen, was zur Folge hat, daß die Periode von 12 Monaten sich im Lauf von 33 Jahren einmal durch ein gregorianisches Jahr schiebt.

## 3.2 Anforderungen an Zeitmessung

Zeit wird in den verschiedenen wissenschaftlichen und technischen Fachbereichen sehr unterschiedlich gehandhabt. Je weiter der betrachtete Zeitpunkt von der Gegenwart entfernt ist, desto gröber wird die erwartete Genauigkeit von Aussagen über ihn. Die Spanne reicht von astrophysikalischen über erdgeschichtliche, historische, betriebswirtschaftliche, technische, mikroelektronische hin zu atomphysikalischen Zeitangaben, bei zunehmender Präzision. Eine Verrechnung von Zeit zwischen den angeführten Kategorien findet nur in engem Rahmen statt, es besteht also kein Bedarf, die seit dem Urknall des Universums vergangene Zeit in Nanosekunden auszudrücken. Ebenso ist es besser, die Leasingdauer einer Industrieanlage in Jahren anzugeben, anstatt in Sekunden. Das Rechnen in adäquaten Dimensionen ist für jede Anwendung größtmöglicher Genauigkeit vorzuziehen.

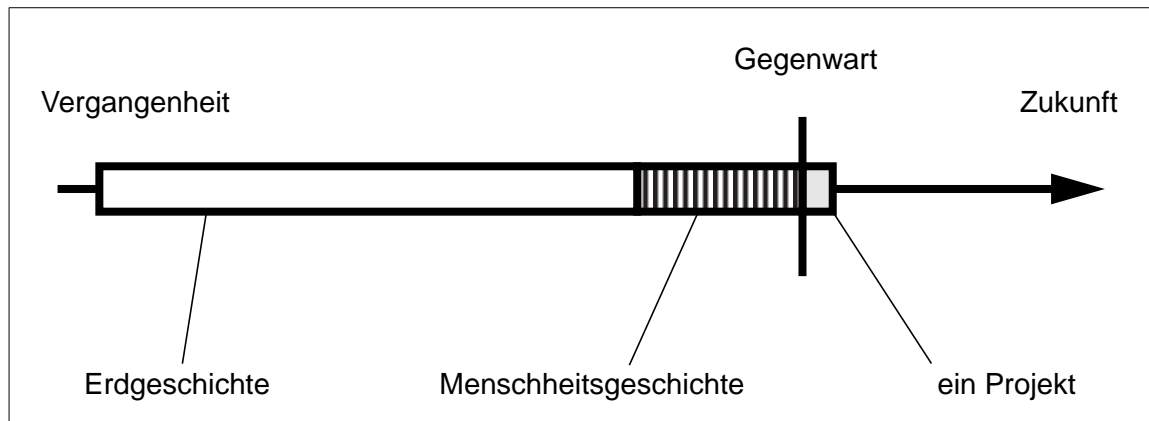


Abb. 4: Größenordnungen der Zeit (schematisch)

### 3.3 Zeitzone

Zeitzone sind größere geographische Gebiete, in denen an Stelle der Ortszeit dieselbe Zeit gilt (Zonenzeit). Die Zeitzone unterscheiden sich je nach ihren geogr. Längenunterschieden gegenüber dem Ortsmeridian von Greenwich (Weltzeit, Coordinated Universal Time, UTC), wobei je 15° geogr. Länge einen Zeitunterschied von 1 Stunde ausmachen. Zeitangaben in UTC werden wegen der Assoziation

nullter Längengrad - zero meridian - z - Zulu (NATO-Alphabet)

auch als „Zulu Time“ bezeichnet.

Die Datumsgrenze ist eine im wesentlichen mit dem 180. Längengrad im Pazifischen Ozean zusammenfallende Linie, bei deren Überschreiten in ost-westl. Richtung ein Datum übersprungen, in umgekehrter Richtung das gleiche Datum wiederholt wird. Hier werden die sich durch Bewegung mit oder gegen die Erdumdrehung ergebenden Zeitdifferenzen ausgeglichen.

### 3.4 Sommerzeit

Aus wirtschaftlichen Gründen wird in einigen Ländern nach Sommerzeit gerechnet. Sie geht gegenüber der betreffenden Zonenzeit meist um eine Stunde vor. Auf diese Weise soll die Ausnutzung des Tageslichts verbessert werden (engl.: *daylight saving time*).

### 3.5 Terminologie

Im Folgenden werden wichtige Begriffe erläutert:

Datum:	Benennung eines Tages nach einem Kalendersystem. Siehe auch Zeitpunkt.
Jahr:	Einheit eines Kalenders. Ganzzahliges Vielfaches von Monaten und Tagen.
Kalender:	System für die Ordnung von Zeiteinheiten die Berechnung von arithmetischen Operationen auf diese. Einheiten eines Kalenders sind in der Regel Tage und deren Vielfache. Es gibt verschiedene Kalendersysteme, deren Benutzung vom jeweiligen Kulturkreis abhängt. Der gregorianische Kalender wird allerdings weltweit als Standard akzeptiert. Die Feiertagsberechnung ist jedoch auf lokale Kalender angewiesen, da spezielle Zyklen zugrunde liegen.
Kalendarium:	Eine Übersicht über aller Ereignisse in einem bestimmten Zeitraum.
Monat:	Einheit eines Kalenders. Ganzzahliges Vielfaches von Tagen.
Schaltjahr:	Jahr, das durch Einfügen/Weglassen eines Tages Unterschiede zwischen dem astronomischen Jahr und dem Kalenderjahr ausgleichen soll.
Schaltsekunde:	Sekunde, die der Synchronisation physikalischer Systeme (z.B. Atomuhren) mit Kalendern dient.
Tag:	Kleinste Einheit eines Kalenders. Die Länge ist durch die Erdrotation festgelegt. Ein Tag hat 24 Stunden.
Stunde:	Zeiteinheit. $1/24$ eines Tages. Eine Stunde hat 60 Minuten.
Minute:	Zeiteinheit. $1/60$ einer Stunde. Eine Minute hat 60 Sekunden.
Sekunde:	Zeiteinheit. $1/60$ einer Minute. Weitere Einteilung in Zehner-Potenzen.
Woche:	Zeitdauer von 7 Tagen zwischen Montag und Sonntag.
Quartal:	Zeitdauer von 3 Monaten, bei einer Aufteilung eines Jahres in 12 Monate wie im gregorianischen Kalender.
Zeitdauer:	Zeit, die zwischen zwei Zeitpunkten verstreicht. Länge eines Zeitintervalles.
Zeitintervall:	Zusammenfassung zweier Zeitpunkte als Begrenzung einer Zeitdauer.
Zeitpunkt:	Ein Punkt auf der Zeitachse. Zwischen Zeitpunkten besteht eine Ordnungsrelation.

## 3.6 Zusammenfassung

Die natürliche Einteilung der Zeit durch die Rotation der Erde und den Stand der Erde zur Sonne bestimmt unser Leben und bildet deshalb die Grundlage unseres Zeitempfindens. Verschiedene Kalendersysteme teilen das Jahr und liefern für jeden fortlaufend nummerierten Julianischen Tag einen eindeutigen Namen. Dies ist der Mechanismus, um Kalendersysteme zu vergleichen.

Da jeder Tag gleich lang ist, kann die Beschreibung einer Tageszeit durch die Zahl vergangener Sekunden seit Tagesbeginn vorgenommen werden.

Eine genaue Festlegung von Terminen kann durch die Kombination von Tagesdatum und Tageszeit geschehen.

Zeitdauern werden als vergangene Zeit dargestellt. Genaue Angaben sind vergangene Stunden oder Tage. Eine zweite, unscharfe Form von Zeitdauern ist die Angabe als Jahre und Monate. Sie können ohne Bezugsdatum nicht exakt in Tage umgerechnet werden, da Jahre bzw. Monate in keinem Kalendersystem gleich lang sind.

Zeitintervalle werden durch zwei absolute Zeitpunkte beschrieben. Die Länge eines Intervalls wird durch die Zeitdauer zwischen den Intervallgrenzen bestimmt.

Eine Beschränkung auf die oben definierten Genauigkeiten ist sinnvoll und deckt weite Anwendungsgebiete ab. Extreme Anforderungen an Zeitmessung in astronomischer bzw. physikalischer Sicht können auch aus technischen Gründen hier nicht abgebildet werden.



## 4.1 Bestehendes Klassensystem

Die zu überarbeitenden Kalendermechanismen der Studienarbeit [Mar95] basieren auf dem Beitrag „Temporal software components“ in [MeNe93] von Lawson, Balthazaar und Gray von der University of Wales, Cardiff. Die Autoren skizzieren eine mögliche Architektur einer objektorientierten Bibliothek für Zeitarithmetik und zeitbezogene Themen.

Die Architektur fußt auf einer Dreiteilung des Zeitbegriffs in Zeitpunkte, Zeitdauern und Zeitintervalle. Abb. 5. zeigt einen Ausschnitt des Klassensystems.

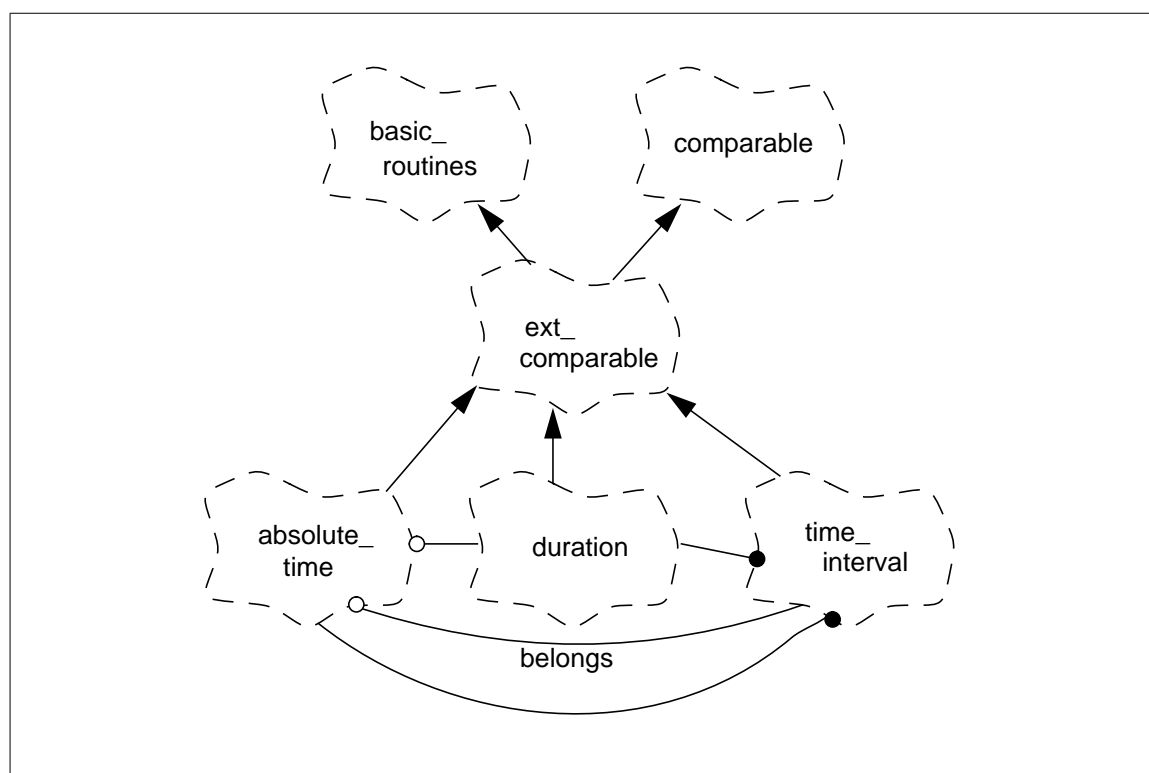


Abb. 5: Primäre Architektur, Kernbeziehungen

Die folgenden abstrakten Basisklassen leiten sich von einer Klasse EXT\_COMPARABLE ab, die mathematische Grundfunktionalitäten wie eine Vergleichbarkeit definiert:

- ABSOLUTE\_TIME
- DURATION
- TIME\_INTERVAL

Sie wurden jeweils in den Granulaten Datum, Uhrzeit und Termin (Datum mit Uhrzeit) konkretisiert, da die Rechenoperationen in diesen Bereichen abgeschlossen sind.

Zusätzlich wurden Klassen für die Beschreibung von Arbeitszeit spezialisiert:

- Eine Klasse WORK\_HMS, um Arbeitszeit an einem Tag zu beschreiben.
- Eine Klasse WORK\_DURATION, um Arbeitsdauern zu beschreiben.
- Eine Klasse WORK\_INTERVAL, um Intervalle von Arbeitszeiten zu beschreiben.
- Eine Klasse WORK\_DT, um Termine zu beschreiben

## 4.2 Kritik

Da die Anforderungen an die vorliegende Bibliothek einem Wandel ausgesetzt sind, muß die Architektur kritisch bewertet werden.

### **Flexibilität der Ein- und Ausgabe**

Das Einlesen und Formatieren von Zeit- und Datumswerten als Zeichenketten ist fest programmiert. Eine Erweiterung über ein Standardformat hinaus ist nur unter Überarbeitung der Klasse bzw. Reimplementierung bei Vererbung möglich. Anzustreben ist ein Mechanismus, der eine einfache Erweiterung zuläßt.

### **Arbeitszeitarithmetik**

Die realisierte Arbeitszeitarithmetik berücksichtigt nur ein einfaches, statisches Arbeitszeitmodell, wie es in nur selten existieren wird. Die Spezialisierung in Arbeitszeitbezogene Klassen ist nicht notwendig sondern eher schädlich, da die gewünschte Flexibilität nicht erreicht wird.

### **Verantwortlichkeiten**

Die bestehende Aufgabenverteilung zwischen den Klassen erzeugt eine zu starke Kopplung. Als Beispiel sei hier die Beziehung zwischen Zeitpunkten und Zeitintervallen genannt: Die Prüfung, ob ein Zeitpunkt in einem Intervall liegt, muß vom Zeitpunkt in das Intervall verlagert werden, da dieses Zeitpunkte schon in seiner Definition benutzt. Auf diese Weise können Zeitpunkte entkoppelt von Intervallen betrachtet werden.

### **Effizienz**

Die Klassen sind hinsichtlich der Speicherausnutzung nicht optimal implementiert.

### **Programming by Contract**

Es werden zahlreiche Marken (flags) gespeichert, die über den Erfolg vorausgegangener Operationen informieren. Dies widerspricht dem „*Programming by Contract*“-Paradigma, bei dem der Benutzer einer Klasse vor Aufruf überprüft, ob die übergebenen Parameter für eine Methode gültig sind [Mey90].

### **Kalendersysteme**

Eine Vergleichbarkeit von Datumsangaben unterschiedlicher Herkunft ist nicht gegeben. Die gesamte Bibliothek ist auf die Verwendung des Gregorianischen Kalenders ausgelegt.

## **4.3 Stand der Technik**

Es ist nicht bekannt, welche Bibliotheken in gängigen Projektplanungssystemen und anderen Softwareprodukten verwendet werden. Wahrscheinlich ist, daß die Hersteller dieser Programme jeweils eigene Klassen verwendeten bzw. Basisfunktionalitäten einkauften und erweiterten. Hierfür bieten sich die am Markt vorhandenen Bibliotheken an.

Bei einer Betrachtung ausgewählter Bibliotheken fällt neben unterschiedlichen Ansätzen der geringe Leistungsumfang auf. Verschiedene Kalendersysteme werden in keiner Bibliothek berücksichtigt.

### **Rogue Wave, C++**

Es existieren zwei Klassen zur Darstellung von Zeitpunkten unterschiedlicher Genauigkeit: `RWDate` repräsentiert ein Tagesdatum und `RWTime` repräsentiert eine Uhrzeit an einem bestimmten Tag seit dem 01.01.1901. Formatierungen übernimmt die abstrakte Basisklasse `RWLocale`. Die Klasse `RWZone` deckt Zeitzonen und Sommerzeit ab [RW96].

### **Microsoft Foundation Classes, C++**

Die Klasse `CTime` stellt Uhrzeit an einem bestimmten Tag dar. Die Klasse `CTimeSpan` steht für Zeitdauern in dieser Auflösung und ist mit `CTime` verrechenbar. Formatierungen funktionieren entsprechend einer Funktion aus der C-Laufzeitbibliothek [MFC96].

### **Cardiff, Tower-Eiffel**

Es existieren Klassen unterschiedlicher Genauigkeit für Zeitpunkte (Datum und Termin) mit einer Werkzeugklasse mit gregorianischen Funktionen. Ein Cluster beinhaltet Klassen für Formatierungen. Die Bibliothek ist nur bedingt mit oben genanntem Artikel [MeNe93] verwandt und gibt nur einen kleinen Teil wieder.

### **Visual Eiffel, SIG**

Die Bibliothek kennt DATE, TIME, und TIMESTAMP (erbt von DATE und TIME). Es sind Zeitdauern verschiedener Genauigkeiten realisiert. Eine Werkzeugklasse TIME\_TOOL liefert einige gregorianische Funktionen.

Die Ausgabe wird durch Formatbeschreibung in einer Zeichenkette definiert.

### **Halstenbach iss-base, erweiterte Standard-Bibliothek für Eiffel**

Diese Bibliothek basiert auf der ISE Eiffel 3 Standardbibliothek [Mey94] und erweitert sie um fehlende Klassen für absolute Zeitpunkte (Zeit/Datum) und vergangene Zeitdauern (Zeit/Datum).

## **4.4 Zusammenfassung**

Das fehlende Angebot objektorientierter Modellierungen für Zeit- und Datumsarithmetik machte es erforderlich, eine eigene Bibliothek zu entwickeln. Die erste Entwicklungsstufe beinhaltet wichtige Basismechanismen, weist aber noch einige Schwächen auf. Erweiterte Anforderungen wie Feiertagsberechnung machen eine tiefgreifende Änderung der Architektur notwendig.

In den folgenden vier Kapiteln werden die einzelnen Teilaufgaben erörtert. Darauf folgt eine Darstellung des resultierenden Klassensystems.

# 5

## Optimierung der Speicherdarstellung

### 5.1 Effizienz

Ein Algorithmus heißt effizient, wenn er ein vorgegebenes Problem in möglichst kurzer Zeit und/oder mit möglichst geringem Aufwand an Betriebsmitteln löst. In der Praxis interessiert man sich meist für die benötigte Laufzeit (bzw. für die Anzahl der auszuführenden Operationen), für die Größe des Speichers oder für die Zahl der Zugriffe auf Hintergrundspeicher. ... [EnHe88]

#### 5.1.1 Speichereffizienz

Die Menge belegten Speichers einer Datenstruktur muß immer im Verhältnis zur Häufigkeit ihres Auftretens zur Laufzeit eines Programms bewertet werden. Der Aufwand einer Optimierung ist häufig nur dann sinnvoll, wenn viele Objekte des betrachteten Typs im Hauptspeicher angelegt bzw. dauerhaft gespeichert werden. In diesem Fall kann durch Reduzierung des Speicherverbrauchs eine Senkung der Kosten einerseits und eine Erhöhung des Systemdurchsatzes andererseits erreicht werden.

#### 5.1.2 Geschwindigkeit des Zugriffs

Eine Reduzierung des Speicherverbrauchs von Objekten kann eine Verschlechterung der Ausführungsdauer bestimmter Zugriffe auf die zugrundeliegende Datenstruktur haben, nämlich dann, wenn Teilinformationen zusammengefaßt werden, um in Basistypen kompakt gespeichert zu werden. Die Effizienz der eingeführten Kodierungsoperationen spielt hierbei eine Rolle. Die relative Häufigkeit bestimmter Zugriffe in einem typischen Programmlauf muß beachtet werden, um in der Summe optimal zu sein.

Die Operationen auf Daten werden in zwei Kategorien eingeteilt: Lesen und Schreiben. Diese entsprechen dem Dekodieren und Kodieren der Teilinformationen und müssen deshalb getrennt betrachtet werden. Ein zusätzlicher Aspekt spielt durch das Modell des abstrakten Datentyps hinein: die interne Repräsentation kann gegebenenfalls ohne Umwandlung in die Komponenten verarbeitet werden.

Ziel ist, eine ausgewogene Lösung zu finden, die Vorteile auf einer Seite nicht mit Effizienzeinbußen auf der anderen Seite bezahlt. Aufwand und Komplexität müssen durch den erzielten Gewinn gerechtfertigt sein. Im Zweifelsfall ist die klare Implementierung der trickreichen vorzuziehen.

### 5.1.3 Vermeidung von Redundanz

Ein weiteres Ziel ist die Vermeidung von Redundanz bei der Speicherung von Objektzuständen. Allerdings kann gerade Redundanz gezielt eingesetzt bei bestimmten Anwendungen die Zugriffsgeschwindigkeit auf Daten deutlich erhöhen. Es ist auch zu beachten, daß Redundanz die Wartungsfreundlichkeit vermindert.

## 5.2 Ausgangssituation

Die Speicherung eines gregorianischen Datums wie auch die Speicherung von Zeit geschieht in der zu überarbeitenden Bibliothek in je einem Basistyp pro Komponente, d.h. für Jahre, Monate und Tage werden jeweils INTEGER-Werte mit einer Genauigkeit von 32-Bit verwendet.

Da Zeit- und Datumswerte manchmal in großer Zahl gespeichert werden müssen, ist eine Optimierung sinnvoll.

## 5.3 Alternative Implementierungen

Im folgenden werden verschiedene Konzepte zur Speicherung von Zeit- und Datumswerten gegenübergestellt.

### 5.3.1 Speicherung in einem Basistyp pro Komponente

Bei einer (heute gängigen) Ganzzahlarithmetik von 32-Bit haben INTEGER-Variablen einen Definitionsbereich von  $-2^{31}$  bis  $+2^{31}$ . Eine komponentenweise Speicherung von Werten im Bereich von 1..12, 1..31 oder -9999..9999 stellt eine durchschnittliche Verschwendung von etwa 75 Prozent Speicher dar. Auf zukünftigen Systemen mit einer erwarteten Ganzzahlarithmetik von 64-Bit sind diese Einbußen noch gravierender.

Der Zugriff auf eine Komponente ist sehr schnell, da keine Umrechnungen notwendig sind.

### 5.3.2 Codierung als Bitfolge

Die Codierung als Bitfolge ist als sehr speichereffizient zu bewerten. In Abb. 6 wird die Codierung zweier kleiner ganzzahliger Werte in eine Variable dargestellt.

In EIFFEL ist eine solche Codierung nicht effizient möglich und müßte durch eine externe C-Funktion realisiert werden. Hierbei läge eine Aushebelung des Typsystems vor. Das EIFFEL-Typsystem läßt absichtlich keine erzwungenen Typkonvertierungen zu. In diesem Fall sind Annahmen über die Implementierung der Klasse INTEGER notwendig, was dem Prinzip der Datenkapsel widerspricht.

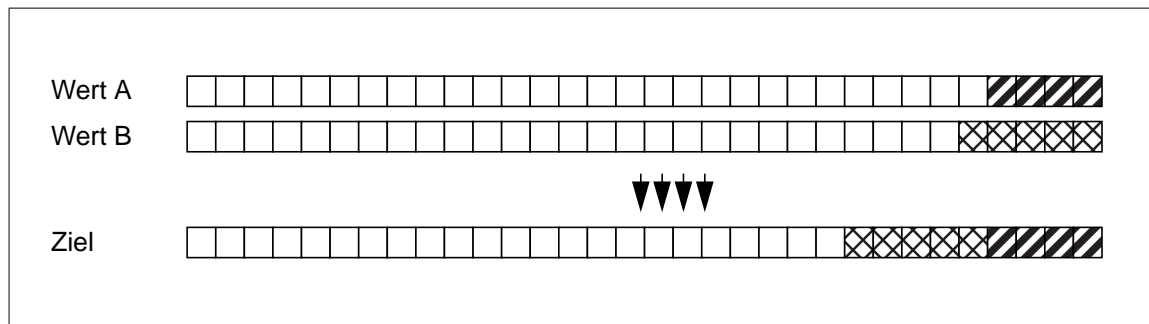


Abb. 6: Binär-Codierung mehrerer INTEGER-Werte in eine einzige Variable

### 5.3.3 Zusammenfassung von Komponenten in Basistypen

Eine andere Möglichkeit der Codierung ist die arithmetische Zusammenfassung. Hierbei werden die einzelnen Komponenten jeweils mit bestimmten konstanten Zehnerpotenzen multipliziert und addiert.

Beispiel 13.03.1996:  $19960313 = 1996 * 10000 + 3 * 100 + 13$

Ganzzahlarithmetik ist in der Regel schnell und kann die geforderten Umwandlungen effizient vornehmen.

Diese Speicherungsart ist fast so speichereffizient wie eine entsprechende Bitfolge. Werden Zweierpotenzen als Faktoren verwendet, entsteht das identische Bitmuster.

Ein wesentlicher Vorteil dieser Darstellung ist die implizite Vergleichbarkeit zweier Codierungen.

### 5.3.4 Speicherung als Julianischer Tag

Bei Datumswerten bietet sich die Speicherung als Julianischer Tag an, da laut Definition ein Kalender für jeden fortlaufend nummerierten Tag eine eindeutige Bezeichnung liefert. Julianische Tagesnummern stellen den fundamentalen Mechanismus zur Konvertierung verschiedener Datumsangaben dar. Sie können in einer Ganzzahlarithmetik von 32-Bit sehr gut dargestellt werden und nutzen den Speicher optimal.

Bei der Verrechnung von Zeitdauern in Tagen und bei der Vergleichsoperation ist diese Methode die effizienteste.

Einzigster Nachteil ist der aufwendigere Umwandlungsmechanismus zwischen der Darstellung als Julianischer Tag und der Darstellung in einer kalenderspezifischen Notation. Diese Umwandlung findet meistens bei der Ein- und Ausgabe statt und ist weniger kritisch zu bewerten als Rechenoperationen bei der Optimierung einer Projektplanung.

## 5.4 Ergebnis

Die Speicherung eines Datums als fortlaufender Wert wie als Julianischer Tag ist bei der gewählten Architektur anderen Konzepten vorzuziehen, da die wichtigsten Operationen sehr effizient implementiert werden können.

Da nur die innere Sicht der Klassen betroffen ist, sind keine Änderungen der Schnittstelle betroffener Klassen bei einer Änderung des Speicherkonzeptes notwendig.

# 6

## Ein- und Ausgabe von Datumswerten

Eine zunehmende Internationalisierung des Lebens führt zu einer Erweiterung des Horizonts und zur Veränderung der individuellen Gewohnheiten. Neue sprachliche Regeln finden Anwendung ohne alte zu ersetzen. Die Akzeptanz einer Software bei ihren Benutzern ist auch davon abhängig, wie sie an deren Fähigkeiten bzw. Gewohnheiten angepaßt ist.

In vielen Softwaresystemen müssen Zeit- und Datumswerte eingelesen oder formatiert werden. Stand der Technik ist, daß von herkömmlichen Werkzeugen zur Softwareentwicklung ein paar wenige Standardformate zur Verfügung gestellt werden. Eine Erweiterung ist in der Regel nicht möglich. Ziel ist, hierbei dem Entwickler mehr Freiheiten zu geben und Erweiterungen der Funktionalität mit vertretbarem Aufwand zu ermöglichen. Objektorientierte Bibliotheken stellen per Definition Grundfunktionalitäten zur Verfügung, die erweitert werden können.

Nach dem Aufzeigen der Vielfalt möglicher Darstellungen und dem Versuch sie zu kategorisieren werden die notwendigen Anforderungen an die Funktionalität gesammelt. Aus dem Vergleich der Entwurfsalternativen folgt die Auswahl einer der möglichen Techniken um diese zu realisieren.

### 6.1 Grundlagen

Im Folgenden werden verschiedene Darstellungen von Zeit und Datum vorgestellt. Zu diesem Zweck wird ein Referenzzeitpunkt eingeführt, der

9. Juni 1931, 3 Uhr 27 Minuten 11 Sekunden.

Notationen, die aus Bequemlichkeit auf die Angabe des Jahrhunderts verzichten, machen im Angesicht der Jahrtausendwende wenig Sinn, da einerseits die maschinelle Sortierbarkeit nicht mehr gegeben ist und andererseits die Zuordnung zu einem Jahrhundert nicht nachvollziehbar ist. Die Verarbeitung von Geburtsdaten sei als Beispiel genannt. Eine 2-stellige Darstellung der Jahreszahl kann nicht mehr problemlos weiter verarbeitet werden.

Es gibt beispielsweise im Bereich der relationalen Datenbanken verschiedene Ansätze, 2-stellige Jahresangaben zu interpretieren und Eingaben entsprechend für die interne Speicherung zu expandieren [INFORMIX, ORACLE]:

- das aktuelle Jahrhundert,
- das 20. Jahrhundert,
- die Zeit zwischen 1950 und 2049 annehmen.

Da hierbei keine Einheitlichkeit herrscht und es für Benutzer nicht offensichtlich ist, welche Technik in der ihm vorliegenden Software verwendet wird, sollte auf die Verwendung 2-stelliger Jahresangaben in Zukunft verzichtet werden.

### 6.1.1 Nationale Darstellungen von Zeit und Datum

Die Vielfalt verschiedener Darstellungen in den Ländern der Welt ist groß. Zusätzlich existieren je Sprachraum unter Umständen mehrere Varianten.

Die Zahl der möglichen Datumsnotationen soll anhand einiger Beispiele verdeutlicht werden:

**9. Juni 1931** bzw. **9. Jun. 31** bzw. **9.6.31** bzw. **09.06.1931**

**09-JUN-1931** bzw. **9-June-1931**

**9/6/31** bzw. **6/9/31** bzw. **1931/06/09** bzw. **19310906**

Alle Darstellungen beziehen sich auf das Referenzdatum.

Die alte englische Darstellung des Tages als zwei Intervalle von 12 Stunden ist nicht mehr gebräuchlich, da sie umständlich und mißverständlich ist. Sie wird deshalb im weiteren Verlauf nicht berücksichtigt.

### 6.1.2 ISO 8601, ein internationaler Standard

Um die Schwierigkeiten, die durch die Vielzahl unterschiedlicher Notationen entstehen zu vermeiden, wurde durch das internationale Normungsgremium ISO eine Darstellung von Zeit und Datum als Standard für internationalen Datenaustausch gewählt. Diese Darstellung besitzt zudem weitere Vorteile im Hinblick auf elektronische Datenverarbeitung. Es folgt ein Überblick über die wichtigsten Darstellungsformen. Weitere Informationen sind in [ISO 88] zu finden.

## Datumsnotation

Ein Datum wird als Gregorianisches Datum in der Form YYYY-MM-DD dargestellt. Hierbei bedeutet YYYY das Jahr in 4 Stellen, MM die Monate in 2 Stellen, DD die Tage in 2 Stellen. Bei Bedarf werden führende Nullen vorangestellt, um die Stellenzahl aufzufüllen.

Als Beispiel folgt das Referenzdatum in ISO 8601 Notation: **1931-06-09**

Die Vorteile dieser Darstellung gegenüber den diversen anderen sind gravierend:

- Sprachunabhängigkeit
- Vergleich und Sortierung über einfache Zeichenkettenoperationen
- Eindeutige Darstellung
- EDV-technisch einfache Realisierung möglich, da keine Tabellen für Monatsbezeichnungen notwendig sind.
- Konsistenz mit der Darstellung der Zeit (absteigende Größe der Einheiten). Deswegen sind auch Zeitpunkte verschiedener Granularität einfach vergleichbar.
- Kurze Notation mit konstanter Länge
- Ein Überlauf nach 1999-12-31 findet nicht statt.
- Die Reihenfolge Jahr, Monat, Tag wird in einigen Nationen schon verwendet. Die Notation ist identisch zur chinesischen Notation (25% der Weltbevölkerung!).

Die Norm erlaubt auch den Verzicht auf die Bindestriche: **19310609**

Gegebenenfalls wird die Genauigkeit reduziert und auf die Angabe des Tages bzw. Monats verzichtet: **1931-06** bzw. **1931**

Desweiteren existiert die wochenorientierte Sichtweise: **1931-W23** bzw. **1931W23**

Tage können wochenorientiert benannt werden, wobei die Wochentage zwischen 1(Montag) und 7(Sonntag) liegen: **1931-W23-3** bzw. **1931W233**

Tage können jahresorientiert benannt werden, wobei die Tage ab dem 1. Januar durchnummeriert werden: **1931-160** bzw. **1931160**

Eine 2-stellige Darstellung der Jahreszahl ist ebenfalls vorgesehen, aus oben genannten Gründen sollte dies jedoch unterlassen werden.

## Zeitnotation

Eine Tageszeit wird in der Form HH:MM:SS dargestellt. HH bedeutet den Stundenanteil, MM den Minutenanteil und SS den Sekundenanteil.

Als Beispiel folgt die Referenzzeit in ISO 8601 Notation: **03:27:11**

Es ist wiederum der Verzicht auf trennende Zeichen erlaubt: **032711**

Gegebenenfalls wird die Genauigkeit reduziert: **03:27** bzw. **0327** bzw. **03**

Um Bruchteile von Sekunden in die Darstellung einzubeziehen wird die Form zu HH:MM:SS.FFFF... erweitert: **03:27:11.025** (Referenzzeit + 25 Millisekunden)

Bei der Kombination von Datums- und Zeitnotation wird als Separator ein T eingefügt.

Der Referenzzeitpunkt sieht in Standard ISO 8601 Notation folgendermaßen aus:

**1931-06-09T03:27:11**

Zeitangaben ohne weitere Kennzeichnung werden als Zeit in der lokalen Zeitzone interpretiert. Um eine Zeitangabe als Weltzeit zu kennzeichnen, wird ein Z angehängt: **03:27Z**

Um die Distanz der lokalen Zeit zur Weltzeit ausdrücklich zu beschreiben kann die entsprechende Zeitdauer an eine lokale Zeitangabe angehängt werden. Als Beispiel folgt eine Zeitangabe in mitteleuropäischer Zeit (MEZ): **03:27+01:00 = 02:27Z**

### Zeitdauernotation

ISO 8601 sieht 4 verschiedene Notationen für Zeitdauern vor:

Zeitdauer als Interval zweier Zeitpunkte: **1931-06-09T03:27:11/1962-03-13T04:36:19**

Zeitdauer als losgelöste Größe, gekennzeichnet mit einem P: **P30Y9M4DT11H9M8S**

Zeitdauer ab einem spezifischen Startpunkt:

**19310609T032711/ P30Y9M4DT11H9M8S**

Zeitdauer bis zu einem spezifischen Endpunkt:

**P30Y9M4DT11H9M8S/1962-03-13T04:36:19**

Die oben beschriebenen Formate kürzen folgende Langform ab:

**30 Jahre, 9 Monate, 4 Tage, 11 Stunden, 9 Minuten, 8 Sekunden**

## 6.2 Analyse

### 6.2.1 Motivation

Die Benutzungsschnittstelle sollte keine Zwänge auf den Benutzer ausüben und ihm weitgehende Freiheiten lassen. Dies erhöht die Akzeptanz eines Programms.

Erweiterungen in der Funktionalität der Klassen, wie sie in objektorientierten Techniken normalerweise gewünscht sind, machen eine Erweiterungsmöglichkeit in der Ein-/Ausgabe notwendig.

Die Vielfalt nationaler und sprachlicher Eigenheiten in der Darstellung von Zeit und Datum ist nicht mit vertretbarem Aufwand erfaßbar. Deshalb muß die Architektur genügend Flexibilität aufweisen.

## 6.2.2 Anwendungsfälle

Die Zahl möglicher Anwendungen für die Konvertierung von Zeit- und Datumswerten zwischen der internen Speicherung und Zeichenketten in verschiedenen Formaten ist groß.

Im Bereich der Eingaben von Zeichenketten sind folgende Fälle denkbar:

- Interaktive Prüfung von Eingaben: Die Eingabe eines Benutzers muß validiert und gegebenenfalls zurückgewiesen werden.
- Alternative Eingabemöglichkeiten: Es sollen verschiedene Datumsformate erkannt werden.
- Darstellung der Eingabemöglichkeiten (z.B. in HTML): Für eine leistungsfähige Online-Hilfe ist es sinnvoll, wenn eine Klasse ihr Eingabeformat selbst darstellen kann.
- Einlesen von Daten von Fremdsystemen (ASCII-Import): Die Kooperation verschiedener unabhängiger Programme ist nur selten über gemeinsame Schnittstellen möglich. In der Praxis wird oft Datenaustausch über Textdateien betrieben, in denen Daten in ihrer Zeichenketten-Repräsentation abgelegt sind. Hier ist es hilfreich, flexible Datentypen zu verwenden, die unterschiedliche Formate kennen.

Bei der Ausgabe von Daten treten folgende Fälle auf:

- Interaktive Ausgaben auf dem Bildschirm: Das Erweitern bzw. Expandieren von Benutzereingaben sowie die Ausgabe von Funktionsergebnissen folgen vielfachen Anforderungen. Die Internationalisierung von Software erfordert die Ausgabe abhängig vom jeweiligen Benutzer.
- Druckformatierung: Die Ausgabe von Daten auf Dokumenten und Listen kann von der Ausgabe auf dem Bildschirm in Form und Länge abweichen.
- Datenaustausch mit Fremdsystemen (ASCII-Export): siehe oben.

## 6.2.3 Anforderungen

Bei einem flexiblen objektorientierten Konzept zur Ein- und Ausgabe von Zeichenketten sind folgende Gesichtspunkte besonders zu beachten:

- Redundanzvermeidung: ein Format soll in beiden Richtungen (Ein- und Ausgabe) gültig sein.
- Modulare Beständigkeit: Kleine Änderungen sollen entsprechend kleine Auswirkungen an der betroffenen Klasse haben.
- Flexibilität und Erweiterbarkeit: der objektorientierte Ansatz impliziert Erweiterbarkeit und verbietet deshalb einen monolithischen Aufbau.
- Klares und verständliches Prinzip: die Erweiterung und Verwendung durch Anwendungsprogrammierer verbietet einen komplexen Ansatz.
- Ein Format muß Metainformationen über sich zur Verfügung stellen.

Zweitrangig sind hier Effizienzüberlegungen, da das Erkennen und Formatieren von Zeichenketten keine Operationen sind, die in großer Anzahl in kurzer Zeit ausgeführt werden müssen.

## 6.2.4 Bekannte Techniken

In Programmiersprachen und Bibliotheken zur Anwendungsprogrammierung werden verschiedene Techniken zur Formatierung von Datumswerten verwendet. Es folgt eine kurze Darstellung mit Bewertung.

### **Fest verdrahtet**

Die überarbeitete Bibliothek verwendete diesen Ansatz. Die Verarbeitung einer Zeichenketten-Eingabe durch eine Methode der jeweiligen Klasse ist sehr wartungsunfreundlich.. Die gesamte Verarbeitung ist durch die Klasse realisiert und deshalb birgt viel Redundanz.

### **Formatmuster**

In Sprachen der vierten Generation (4GL), relationalen Datenbanken [INFORMIX, ORACLE] wie auch in objektorientierten Bibliotheken (siehe Kapitel 4) ist die Formatierung von Zeit- und Datumswerten durch Formatmuster üblich.

Das Format eines gregorianischen Datums in ISO-Norm wird z.B. in folgender Schreibweise definiert: YYYY-MM-DD (siehe auch Abschnitt 6.1.2). Ein Interpreter des Laufzeitsystems erkennt bzw. formatiert den Datumswert entsprechend. Dieses bedeutet nicht, daß das Format zur Laufzeit geändert werden kann.

Dieses Verfahren ist flexibel in den Grenzen der Sprache, in der es definiert ist. Es kann nicht durch neue Codes erweitert werden.

## Eingabe als Sprache

Gültige Eingaben von Zeit- und Datumswerten können als formale Sprachen begriffen werden. Nach der Spezifikation der Grammatik kann nach den Prinzipien des Compilerbaus ein Übersetzer konstruiert werden, der aus den typischen Phasen lexikalische, syntaktische und semantische Analyse besteht [ASU88].

Bei dem Versuch, einen Prototypen zu implementieren, trat ein technisches Problem bei gleichzeitiger Arbeit mit mehreren Parsern aus der Standardbibliothek auf. Da eine Eigenentwicklung im Rahmen der Arbeit nicht möglich war, wurde dieser Weg nicht weiter verfolgt. Die Lösung mit Grammatiken ist mächtig, aber sehr schwer erweiterbar und scheidet schon deshalb aus.

## 6.3 Entwurf

Da gängige Techniken zu viele Nachteile aufweisen, wurde ein eigenes Konzept entwickelt, welches sich jedoch an formale Sprachen anlehnt.

### 6.3.1 Flexibilisierungskonzept

Die Formatierung von Zeit- und Datumswerten beruht auf der Konstruktion von sogenannten Scannern, die als Listen von primitiven syntaktischen Objekten (Terminalen) realisiert sind. Bei einem Umwandlungsvorgang einer Zeichenketten in die interne Repräsentation prüfen alle Scanner eines Objekts nacheinander die Zeichenkette, bis diese als gültige Eingabe erkannt wurde. Das Objekt setzt im Erfolgsfall seine Daten entsprechend, andernfalls wird die Eingabe zurückgewiesen.

Durch die Zusammenfassung von Ein- und Ausgabe wird die redundante Definition eines Formats vermieden.

## 6.4 Ergebnis

Die abstrakte Basisklasse SCANNABLE dient der Definition von Funktionalitäten des Erkennens von Eingaben und Formatieren von Ausgaben als Zeichenketten. Von ihr werden alle Zeit- und Datumsklassen abgeleitet. Solche Nachfahren sind dann formatierbare Objekte und haben einen oder mehrere Scanner mit Listen von Terminalen. Wenn ein Scanner eine Zeicheneingabe erkannt hat, wird der Zustand des formatierbaren Objektes durch sich selbst aus den eingelesenen Daten des Scanners gesetzt. Die abgeleitete Klasse muß deshalb eine Methode zum Setzen des Zustandes aus einem erkennenden Scanner besitzen.

Ein Überblick über die entstandenen Klassen befindet sich in Kapitel 9.



# 7

## Automatische Berechnung von Feiertagen

Für Zeit- und Terminplanung, Koordination von Projekten, Arbeitsvorbereitung, Erstellen von Kalendern und Plänen ist die Berechnung von Feiertagen eine fundamentale Aufgabe. Da die Terminierung nicht jedes Jahr gleich liegt und die zugrundeliegenden Algorithmen teilweise kompliziert sind, ist eine maschinelle Berechnung der manuellen Datenerfassung vorzuziehen.

Feiertage stellen nur eine Teilmenge aller zu berücksichtigenden Ereignisse dar, daraus folgt eine Erweiterung der Aufgabenstellung zur automatischen Berechnung von Datumsereignissen.

### 7.1 Grundlagen

Die Klassifizierung von Datumsereignissen geschieht primär nach ihrer Herkunft:

- Feiertage
- arbeitsfreie Tage
- betriebliche Ereignisse

In zweiter Linie sind Unterschiede in der Art ihrer Berechnung zu machen.

#### 7.1.1 Feiertage

Feiertage sind Tage, an denen die Arbeit ruht. An gesetzlichen Feiertagen sind alle Arbeiten verboten, die geeignet sind, die äußere Ruhe des Tages zu beeinträchtigen. Das Verbot gilt nicht für den Betrieb der Post, Eisenbahn, Straßenbahn und Kraftomnibuslinien sowie u.a. bei unaufschiebbaren Arbeiten in der Landwirtschaft. Welche Tage als Feiertag gelten, unterliegt in Deutschland der landesrechtlichen Regelung. [Kna76]

Diese ältere Definition aus dem Lexikon zeigt die Problematik deutlich auf. Es kann keine allgemeingültige Berechnungsvorschrift für Feiertage geben, da diese in unterschiedlichen Bundesländern schon verschieden geregelt sind. Dieses gilt erst recht für die verschiedenen Staaten der Erde.

Ein Berechnungsverfahren für Feiertage in einem Programm mit dem Anspruch, internationale Termine zu koordinieren, muß flexibel genug sein, um alle Kulturkreise der Erde zu berücksichtigen.

Feiertage sind kirchlicher oder staatlicher Herkunft.

Eine Aufzählung wichtiger Feiertage befindet sich in Anhang A. Diese ist notwendig, um einen Ansatzpunkt für die Analyse zu haben.

### **Kirchliche Feiertage**

Den Schwerpunkt der Aufzählung im Rahmen der Arbeit bilden christliche Feiertage [EAWRE96, KST96]. Sie werden in zwei Kategorien eingeteilt:

- von Ostern abhängig
- nicht von Ostern abhängig.

Ostern als wichtiges christliches Hauptfest spielt eine zentrale Rolle bei der Berechnung von Feiertagen.

### **Berechnung des Osterdatums**

Das wichtigste Fest des christlichen Jahres gilt der Feier der Auferstehung Christi. In vielen Kirchen werden Gottesdienste in der Osternacht gefeiert. Das Osterfest besteht aus dem Ostersonntag und dem Ostermontag

In den Jahren vor 325 wurde Ostern in verschiedenen Regionen an verschiedenen Tagen gefeiert. Diese waren nicht unbedingt Sonntage. Mit dem Konzil von Nizza wurde allerdings festgelegt, daß Ostern in Zukunft an Sonntagen gefeiert würde. Dionysius Exiguus legte 532 eine Methode fest, um das Datum des Osterfestes zu berechnen, die 725 bestätigt und akzeptiert wurde.

Aloisius Lilius ersann das System, daß die Basis des Gregorianischen Kalenders wurde, wie auch die Tabellen, die zur Berechnung des Osterdatums notwendig sind. Christoph Clavius modifizierte diese Tabellen leicht und wurde einer der ersten Verfechter des Gregorianischen Kalenders. Seit dessen Inkrafttreten im Jahre 1582 wird Ostern im Westen nach diesen Tabellen berechnet. Die Orthodoxe Kirche weicht in ihrer Rechenweise ab.

Ostern ist der erste Sonntag nach dem vierzehnten Tag (nach Neumond) nach dem 21. März. Der Zyklus des Osterdatums wiederholt sich nach 5,7 Mio. Jahren.

In vielen Staaten wurde der Gregorianische Kalender erst lange nach 1582 eingeführt, so daß auch die Berechnung des Osterfestes entsprechend abwich. In manchen deutschen Staaten und Schweden wurde trotz Einführung des Gregorianischen Kalenders Ostern lange nach Beobachtungen des Tycho Brahe bestimmt.

Es gibt verschiedene Algorithmen um Ostern zu bestimmen. Hier sollen die bekanntesten und auch anwendungsfreundlichsten erwähnt werden:

- Gauss, gültig ab 1583: Wenn der erste Frühlingsvollmond auf einen Sonntag fällt, verschiebt sich Ostern um eine Woche, um nicht mit dem jüdischen Passahfest zusammen zu fallen. Der Algorithmus wird in Abb. 7 dargestellt.
- Carter, gültig von 1900 bis 2099
- Oudin, gültig ab 1583

```

date_of_easter (year: INTEGER): GREGORIAN_DATE is
  require
    gregorian_year: year > 1582
  local
    i, j, a, b, day, month: INTEGER
  do
    i := year // 100 - year // 400 + 4
    j := i - year // 300 + 11
    a := (((year \ 19) * 19) + j) \ 30
    b := (((year \ 4) * 2 + (4 * year) + 6 * a) + i) \ 7 + a - 9
    if b < 1 then
      day := 31 + b
      month := 3
    else
      if b = 26 or else
        (a = 28 and then b = 25 and then (11 * (j + 1) \ 30) < 19)
      then
        b := b - 7
      end
      day := b
      month := 4
    end
    !!Result.make (year, month, day)
  end -- date_of_easter

//  ganzzahlige Division
\  Modulo-Division

```

Abb. 7: Der Algorithmus von Gauss in EIFFEL [CFD88]

Es spricht einiges dafür, daß die Algorithmen zur Berechnung des Osterfestes in ferner Zukunft ihre Gültigkeit nicht behalten werden. Der primäre physikalische Grund ist die stetige Verlängerung der einzelnen Tage, so daß die Anzahl der Tage über das Jahr hinweg abnimmt. Aus diesem Grunde wird es nötig sein, im 4. oder 5. Jahrtausend ein Schaltjahr auszulassen. Größere Wahrscheinlichkeit hat allerdings die Vermutung, daß das Osterdatum in näherer Zukunft auf ein bestimmtes Datum fixiert wird. Papst Johannes XXIII. erklärte auf dem 2. vatikanischen Konzil, daß es nicht falsch sei, dieses Datum festzulegen. Der Weltkirchenrat signalisiert breite Zustimmung. Laut Encyclopaedia Britannica ist der zweite Sonntag im April favorisiert. Damit wären die von Ostern abhängigen Feiertage immer noch beweglich, aber auf eine Spanne von sieben Zeitpunkten eingegrenzt [EnBri71].

### 7.1.2 Arbeitsfreie Tage

Neben Feiertagen sind andere Tage im Jahr näher zu betrachten. Von den 366 Tagen des Jahres 1996 waren in Westdeutschland durchschnittlich 207 effektive Arbeitstage. In Abb. 8 wird dies näher aufgeschlüsselt [Cost96].

Besonders fällt der Anteil der Sonntage und Samstage bei den arbeitsfreien Tagen auf. Von den 4,2 Millionen abhängig Erwerbstätigen in Baden-Württemberg arbeiten 654000 Sonntags (15,6 Prozent). 323000 Personen tun dies nur gelegentlich, 223000 regelmäßig und 99000 ständig. 80 Prozent der Sonntagsarbeit wird von öffentlichen und privaten Dienstleistern erbracht. Industrielle Arbeit am Sonntag war nach der alten Gewerbeordnung nur aus technischen Gründen zulässig. Die Neuregelung des Arbeitsschutzes am Sonntag durch das Arbeitszeitgesetz im Jahr 1994 ermöglicht jetzt die Begründung mit wirtschaftlichen Interessen. Eine Ausnahmegenehmigung erfordert den Nachweis, daß

- die wöchentlich zulässige Betriebszeit von 144 Stunden weitestgehend ausgeschöpft wird,
- ausländische, namentlich genannte Konkurrenten länger als 144 Stunden pro Woche produzieren,
- die Konkurrenzfähigkeit unzumutbar beeinträchtigt ist und
- Arbeitsplätze gesichert werden.

Diese Regelungen sind in anderen Staaten abweichend ausgeführt. Der internationale Wettbewerb macht eine Flexibilisierung erforderlich.

Das Arbeitsrecht stellt die oberste Instanz der Regelung von arbeitsfreien Tagen dar. Als nächste Stufe ist der Tarifvertrag jeder Branche anzusehen, der z.B. Samstage als arbeitsfrei definieren kann. Außertariflich Angestellte fallen gegebenenfalls nicht unter diese Regel.

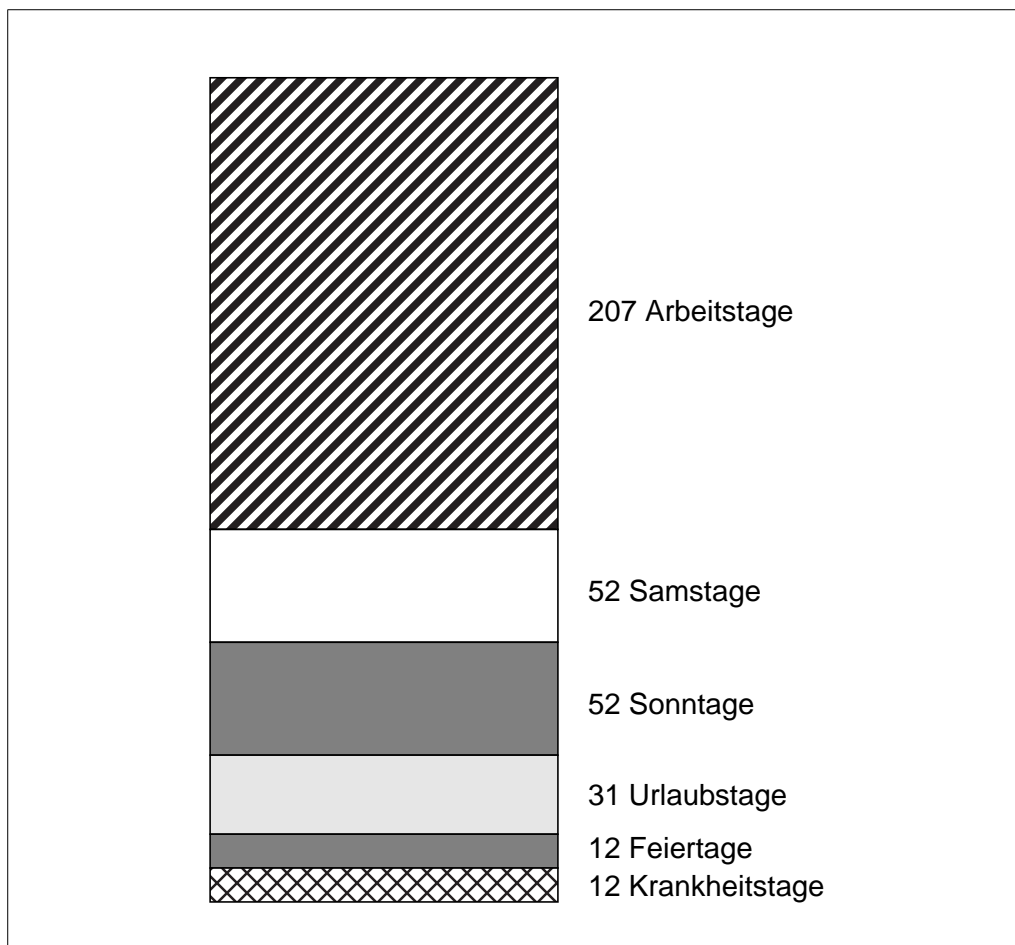


Abb. 8: Durchschnittlicher Anteil der Arbeitstage im Jahr 1996

Betriebliche Vereinbarungen bzw. Festlegungen werden meistens für Urlaub und ähnliches getroffen. So kann ein Teil des Jahresurlaubs aller Mitarbeiter auf einen bestimmten Zeitpunkt (Betriebsurlaub) festgelegt werden. Der Rest ist individueller Urlaub, der über das ganze Jahr zu beliebigen Stückelungen genommen werden kann.

### 7.1.3 Betriebliche Ereignisse

Im betrieblichen Ablauf kann eine Vielzahl von Ereignissen den regulären Gang stören. Diese Ereignisse müssen eingeplant werden, um eine korrekte Abschätzung der vorhandenen Kapazitäten leisten zu können.

Im einzelnen sind hier Inventur, Umzug, Umbau, Betriebsversammlung usw. zu nennen.

### **7.1.4 Sonstige**

Als arbeitsfrei im Sinne von Verfügbarkeit von Ressourcen kann auch Krankheit von Arbeitnehmern gesehen werden (s.a. Abb. 8). Hierbei ist die Kurzfristigkeit von besonderer Bedeutung, da sie oft für Terminüberschreitungen verantwortlich ist.

Die Teilnahme an Fortbildungsmaßnahmen ist ebenfalls von Bedeutung, da in diesem Moment die normale Arbeit unterbrochen wird und keine direkte Verfügbarkeit besteht.

## **7.2 Analyse**

Die umfangreiche Darstellung möglicher Datumseignisse bildet die Grundlage für eine eingehende Analyse des Gebiets.

### **7.2.1 Charakterisierung**

Im Folgenden werden Datumseignisse charakterisiert und gemäß ihrer Eigenschaften in Kategorien eingeteilt.

#### **Häufigkeit des Auftretens**

In der einfachsten Form tritt ein Datumseignis genau einmal auf. Die Mehrzahl der Fälle betrifft allerdings mehrfaches Auftreten, wobei hier unterschieden werden muß, ob regelmäßig oder unregelmäßig.

Regelmäßige Ereignisse werden entsprechend ihrer Periodizität durch die Angabe einer Zeitdauer beschrieben. Denkbar sind Tage, Wochen, Monate, Quartale, Jahre, Jahrzehnte usw.

Unregelmäßiges Auftreten wird am besten durch Aufzählung beschrieben.

#### **Dauer des Ereignisses**

Als Dauer eines Datumseignisses kann nicht nur ein Tag angenommen werden. Es müssen auch Vielfache berücksichtigt werden. Im Allgemeinen wird diese Dauer fest sein, trotzdem sind auch bewegliche Dauern denkbar.

#### **Unterscheidung nach dem Bezugssystem**

Insbesondere die Berechnung der kirchlichen Feiertage ist vom zugrunde liegenden Kalendersystem abhängig. Der Gregorianische Kalender tritt hiebei anteilsbezogen in den Hintergrund. In orthodox geprägten Staaten wird der Julianische Kalender verwendet, andere wichtige Kalender sind der jüdische, der mohammedanische und der chinesische Kalender.

Betriebe richten sich im wesentlichen zwar nach dem kalendarischen Jahr, können aber eigene Zyklen definieren, die davon abweichen.

Ein weiteres Bezugssystem sind Maschinenkalender, die Wartungsintervalle von Maschinen wiedergeben.

Es wird deutlich, daß ein Mechanismus notwendig ist, um Vergleichbarkeit von Datumsangaben verschiedener Kalendersysteme herzustellen. Hierfür bietet sich die Projektion eines jeden Kalenders auf eine Folge von Tagen an.

### **Rhythmus**

Datumsereignisse können bezüglich dem Kalenderrhythmus fest verankert oder beweglich sein. Im zweiten Fall ist die Angabe einer Berechnungsvorschrift notwendig.

### **Autonomie**

Ein Großteil der Datumsereignisse ist selbständig und unabhängig von anderen zu berechnen. Insbesondere für zahlreiche christliche Feiertage trifft dies nicht zu, da sie von Ostern abhängig sind. Hierbei wird die Berechnung an ein anderes Ereignis gekoppelt.

### **Algorithmus**

Die Berechnung beweglicher Datumsereignisse kann durch die Angabe einer Liste geschehen. Dieses ist eine pragmatische und manchmal gerechtfertigte Methode, die allerdings aufwendig und fehlerträchtig ist. Besser ist die Beschreibung fundamentaler, unveränderlicher Datumsereignisse durch einen Algorithmus. In wenigen Fällen kann der Algorithmus sehr komplex werden (s.a. Abb. 7).

Viele Termine werden flexibel, interaktiv durch Benutzer nach Prüfung freier Tage festgelegt. Hier kann nur eine Listenform angewendet werden.

Die Berechnung von Wartungsintervallen von Maschinen ist von deren Benutzung abhängig, d.h. hier entsteht eine Forderung nach Iteration. Zyklische Abhängigkeiten müssen vor dem Hintergrund der Implementierung im Entwurf besonders betrachtet werden.

### **Bedeutung und Herkunft**

Zwischen Datumsereignissen bestehen neben technischen auch inhaltliche Unterschiede. Zum einen ist die Herkunft relevant, d.h. ob staatlich oder betrieblich bedingt. Zum anderen muß die Bedeutung im arbeitsrechtlichen Zusammenhang gesehen werden.

Die Wirkung eines Datumsereignisses ist nicht Teil seiner Definition, sondern kann je nach Kontext bei seiner Verwendung festgelegt werden.

## 7.2.2 Anforderungen

Es gibt zwei wesentliche Anforderungen an eine Bibliothek zur Berechnung von Datumsereignissen:

- Erzeugen einer Liste aller Ereignisse in einem bestimmten Zeitraum. Diese Funktion wird bei der Erstellung von Kalendarien benötigt.
- Abfrage zu einzelnen Tagen, ob bestimmte Ereignisse eingetreten sind. Diese Funktion wird bei der Arbeitszeitberechnung benötigt.

Das Verteilen von Zeitdauern, wie bei der Berechnung von Zielterminen und bei der Ressourcenplanung notwendig, läßt sich auf eine Kombination beider Fälle zurückführen.

Die für die Berechnung von Datumsereignissen benötigten Daten werden im Allgemeinen erst durch den Anwender bestimmt. Ein hoher Anteil an interaktiver Änderung von Parametern zur Laufzeit ist zu erwarten.

## 7.2.3 Zusammenfassung

Datumsereignisse werden durch ihre Dauer und die Art ihrer Berechnung beschrieben. Aus dieser Beschreibung ergibt sich die Häufigkeit ihres Auftretens in einem Zeitraum.

# 7.3 Entwurf

## 7.3.1 Mengenmodell

Datumsereignisse können je nach Herkunft zu Mengen zusammengefaßt werden, um eine gemeinsame Verwaltung zu ermöglichen. Die beliebige Kombination dieser Mengen durch Vereinigung ergibt Obermengen, die für weitere Berechnungen verwendet werden können. Als pragmatische Erweiterung ist auch ein Ausschluß einzelner Ereignisse aus solchen Obermengen möglich.

## 7.3.2 Attributierung

Um der generellen Forderung nach Flexibilität gerecht zu werden und um verschiedene Wirkungen des gleichen Datumsereignisses in verschiedenen Kontexten zu ermöglichen, wird auf eine Objektsicht bei der Attributierung verzichtet und statt dessen eine Mengensicht verwendet, d.h. Attributierung durch Zusammenfassung.

Erst bei der Berechnung von Kalendarien (z.B. arbeitsfreie Tage) wird entschieden, ob eine Menge von Ereignissen dazu gehört.

### 7.3.3 Spezialisierung bei Ereignissen

Alle fundamentalen Operationen auf Datumseignisse werden von einer abstrakten Basisklasse definiert und durch geeignet erweiterte, abgeleitete Klassen implementiert.

### 7.3.4 Abhängigkeitsmechanismus

Um eine möglichst lose Kopplung zwischen den Klassen dieses Bereichs zu erreichen, ist ein dynamischer Aktualisierungsmechanismus notwendig. Ein zyklensfreier, gerichteter Graph beschreibt die entstehenden Abhängigkeiten zwischen Datumseignissen, Ereignismengen und weiteren Strukturen. Zuerst werden Verknüpfungen zwischen Objekten nach einem bestimmten Protokoll erzeugt. Nach Änderung eines beteiligten Objektes werden alle abhängigen Objekte informiert und für ungültig erklärt. Bei einem Zugriff auf ein solches Objekt muß dann zuerst die Gültigkeit hergestellt werden.

Die „faule“ Aktualisierung eines geänderten Objekts hat den Vorteil, daß es nur bei Bedarf neu berechnet werden muß.

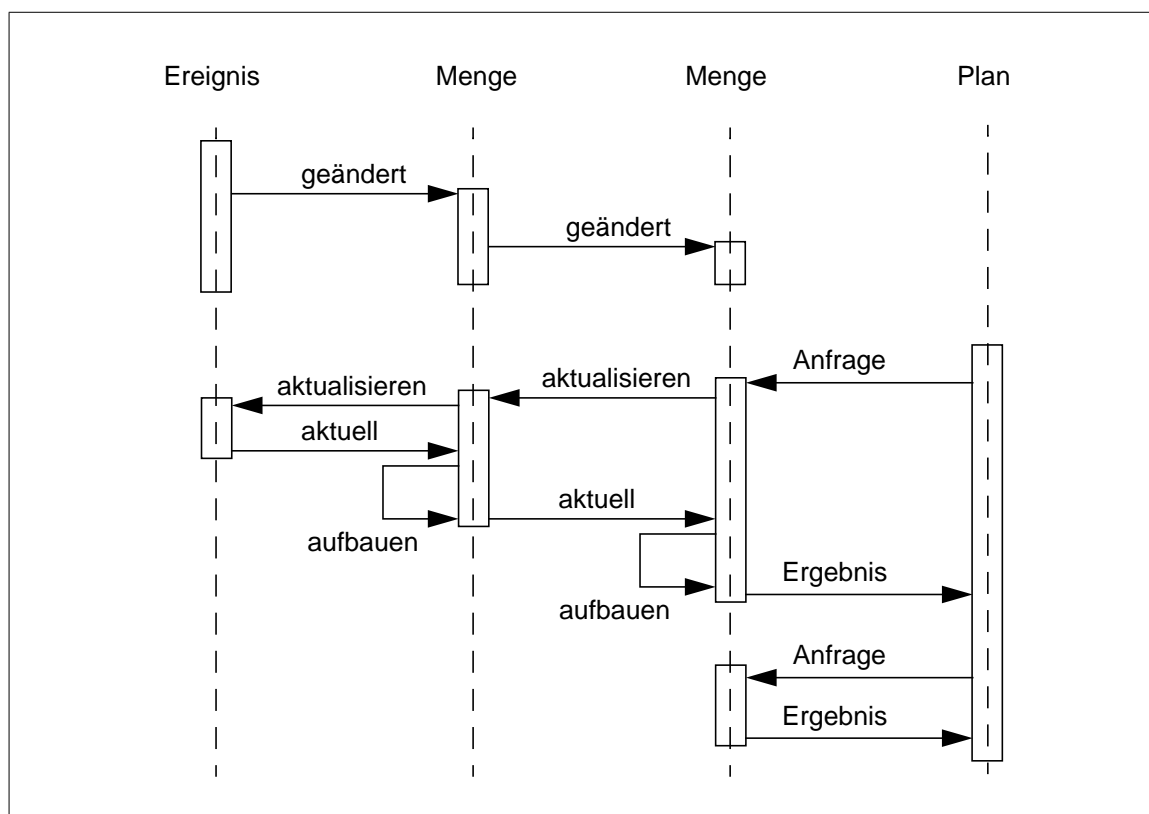


Abb. 9: Benachrichtigungsschema bei Änderungen an Datumseignissen

Das Interaktionsdiagramm in Abb. 9 beschreibt die Aktivitäten Ändern, Benachrichtigen, Berechnen und Aufbauen in einer angenommenen Konfiguration zur Laufzeit.

### 7.3.5 Dynamische Bereichserweiterung

Die Berechnung der Datumsereignisse muß aus Effizienzgründen initial auf einen bestimmten Datumsbereich beschränkt werden, der bei Bedarf erweitert wird.

### 7.3.6 Datenstruktur für effiziente Abfragen

Aus der Analyse der Anforderungen (Abschnitt 7.2.2) wird die Notwendigkeit einer Liste aller Tage in einem Datumsbereich deutlich, um Aussagen über Datumsereignisse an einzelnen Tagen treffen zu können.

Hier sind zwei alternative Realisierungen möglich. Die erste Lösung verwendet eine Bitliste um binäre Aussagen über einzelne Tage treffen zu können, hat allerdings den Nachteil der aufwendigeren Berechnung der konkreten Datumsereignisse. Die zweite Lösung verwendet für jeden Tag eine (meistens leere) Liste der zutreffenden Datumsereignisse. Dieses Datenstruktur ist weit weniger speichereffizient, hat aber dafür eine sehr gute Zugriffsgeschwindigkeit. Geht man davon aus, daß in der Regel nur Zeiträume von einigen Jahren berechnet werden, ist diese Lösung vorzuziehen.

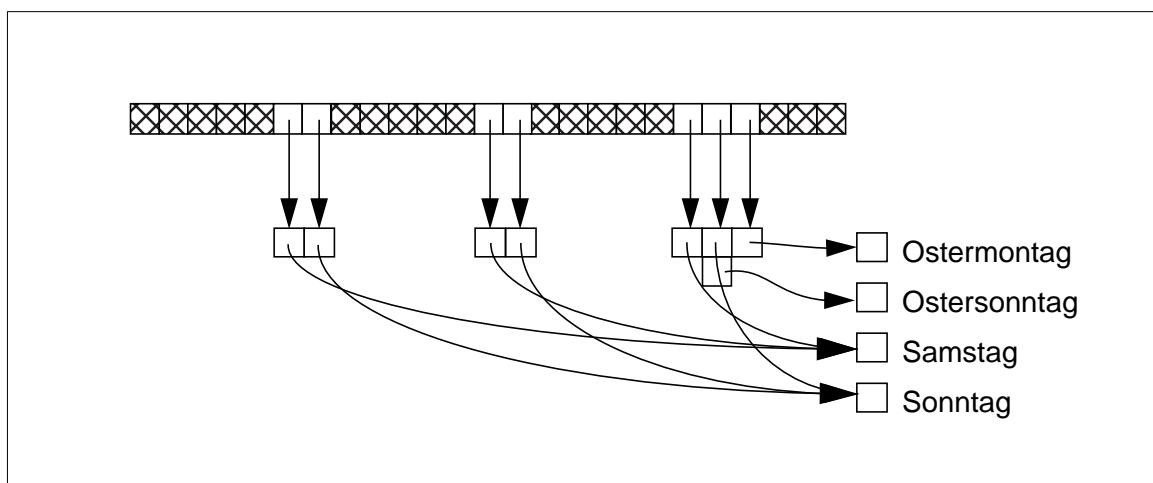


Abb. 10: Datenstruktur für eine Liste von Datumsereignissen

Abb. 10 zeigt eine solche Liste von Datumsereignissen am Beispiel mehrerer Wochen um Ostern. Die schraffierten Tage beinhalten keine Datumsereignisse und würden somit beispielsweise als Arbeitstage interpretiert.

## 7.4 Ergebnis

Abstrakte Datumsereignisse stellen den zentralen Mechanismus des Konzeptes zur Feiertagsberechnung dar. Konkrete Nachfahren dieser Klasse speichern die Definition eines Datumsereignisses, die aus dem Namen und einem Algorithmus zur Berechnung der Periodizität besteht. Ein Datumsereignis kann eine Liste aller Termine seines Eintreffens in einem bestimmten Zeitraum erzeugen. Es tut dies ausgehend vom ersten, indem jeweils das nächste berechnet wird.

Datumsereignisse werden als Mengen verwaltet, um verschiedene Kategorien kombinieren zu können, d.h. diese Mengen werden miteinander verknüpft und bilden dann die Vereinigungsmenge.

Erst bei der ersten konkreten Anfrage nach bestimmten Tagen wird aus der Menge der Datumsereignisse eine Liste von Tagen gebildet, in der alle „Treffer“ markiert sind.

Unter der Mengenstruktur liegt ein Abhängigkeitsmechanismus, der für eine Neuberechnung aller Datenstrukturen sorgt.

Ein Überblick über die entstandenen Klassen befindet sich in Kapitel 9.



# 8

## Erweiterung der Arbeitszeitarithmetik

Arbeitszeit ist die Zeit, die ein Arbeitnehmer in seinem Betrieb verbringt, abgerechnet die bewilligten Ruhepausen. Auf Grund gewerkschaftlicher Bestrebungen wurde gegenüber den Anfängen der Industrialisierung die Arbeitszeit in allen Ländern im Laufe eines Jahrhunderts von wöchentlich 69 (England) bis 83 Stunden (Deutschland) auf 40 bis 48 Stunden gesenkt. Gemäß der Arbeitszeit-Ordnung vom 30.4.1938 ist in Deutschland in der gewerblichen Wirtschaft der Achtsturentag gesetzlich festgelegt. Auch in Ausnahmefällen (Vor- und Nacharbeit, Notfälle) dürfen 10 Stunden täglich nicht überschritten werden. Sie müssen in Tarifverträgen geregelt oder vom Gewerbeaufsichtsamt genehmigt sein. ... [Kna76]

### 8.1 Grundlagen

#### 8.1.1 Definitionen

##### **Arbeitstage**

Arbeitstage sind alle Tage, die nicht arbeitsfrei sind.

Gesetzliche Feiertage und Sonntage sind keine Arbeitstage, Samstage sind durch Tarifverträge geregelt. Ausnahmen sind prinzipiell möglich, da gesetzlich vorgesehen, und deshalb in einem Konzept zur Berechnung von Arbeitszeiten zu berücksichtigen. Die Feiertage aus der Feiertagsberechnung können durch „spezielle Tage“ aus einer anderen Menge von Datumereignissen in diesen Fällen neutralisiert werden. In Kapitel 7 wird auf die Berechnung von Feiertagen (allgemein: Datumereignisse) eingegangen.

Das Arbeitsrecht setzt zusätzliche Randbedingungen:

- Mindestanzahl freier Sonntage/Feiertage
- generelles Arbeitsverbot an bestimmten Feiertagen

Die Menge der Arbeitstage hat geographisch nur eine begrenzte Gültigkeit, da Feiertage nur lokal gültig sind.

## **Arbeitszeit**

Arbeitszeit ist eine Zeitdauer, die von einem Arbeitnehmer für die Erledigung von Aufgaben einem Betrieb zur Verfügung gestellt wird. Dieses Verhältnis wird im Rahmen einer Projektplanung auch als Verfügbarkeit einer Humanressource bezeichnet. In diesem Zusammenhang ist folgendes von Interesse:

- die Summe aller Arbeitszeiten in einem Zeitraum,
- der Termin, an dem eine bestimmte Menge Arbeitszeit vergangen ist.

Eine Unterbrechung der effektiven Arbeitszeiten eines Tages (etwa durch Pausen) ist je nach Anspruch an die Genauigkeit des Ergebnisses zu berücksichtigen. So kann die Arbeitszeit eines Arbeitstages aus verschiedenen Teilen zusammengesetzt sein oder als ein Block gesehen werden, der einen Pausenanteil beinhaltet.

Weiter ist zu berücksichtigen, daß der Arbeitsbeginn und das Arbeitsende an verschiedenen Tagen liegen können.

## **Arbeitszeitmodell**

Da Arbeitsvorgänge eine Koordination von Teilprozessen erfordern, unterliegen Arbeitszeiten gewissen Regeln. Diese Regeln werden in Arbeitszeitmodellen formuliert.

### **8.1.2 Statisches Arbeitszeitmodell**

Das häufig anzutreffende statische Arbeitszeitmodell ist das zugleich einfachste.

Der Arbeitnehmer hat feste Arbeitszeiten, die an jedem Arbeitstag gleich sind. Die Zahl der Pausen ist geregelt und unter Umständen ist deren Zeitpunkt vorgeschrieben.

### **8.1.3 Dynamische Arbeitszeitmodelle**

In der betrieblichen Praxis treten eine Vielzahl von Variationen auf. Ohne Anspruch auf Vollständigkeit werden hier die wichtigsten Komponenten dargestellt.

#### **Arbeitsbeginn und -ende flexibel**

Besonders bei Angestellten, die nicht in starre Produktionsvorgänge eingebunden sind, sind Arbeitsbeginn und -ende in Grenzen flexibel. Arbeitgeber verlangen die Anwesenheit in festgelegten Kernzeiten und das Erfüllen einer vertraglichen täglichen Arbeitsdauer. Die Planbarkeit erfordert eine Festlegung einer festen (Plan-)Arbeitszeit wie im statischen Arbeitszeitmodell, an die der Arbeitnehmer gebunden ist wenn Termine existieren.

### **Arbeitsdauer flexibel**

Die tägliche Arbeitsdauer ist nicht einheitlich und kann frei gewählt werden oder unterliegt betrieblichen Anforderungen.

### **Wechselschicht**

In Produktionsbereichen, die einen hohen Maschinenauslastungsgrad erfordern, ist die Arbeit in Wechselschicht üblich. Die betroffenen Arbeitnehmer werden in Gruppen (Schichten) eingeteilt. Die Arbeitszeit ändert sich dabei in z.B. wöchentlichem Rhythmus, um alle Schichten gleichmäßig zu belasten. Im Dreischichtbetrieb ist es möglich, Produktionsanlagen 24 Stunden am Tag auszulasten. Nach einer Nachtschicht folgt in der Regel eine Freischicht, um dem menschlichen Körper Erholung zu gewähren.

### **Sonderschichten**

Um die Kapazität zu erhöhen kann es notwendig sein, Sonderschichten in den regulären Ablauf einzubauen. Hierzu zählen auch geplante Überstunden, die die tägliche Arbeitszeit verlängern.

### **Teilzeitarbeit**

Eine große Zahl Arbeitnehmer ist nur in Teilzeit-Arbeitsverhältnissen beschäftigt. Hierzu zählen auch freie Mitarbeiter. Es sind anteilige Arbeitszeiten an jedem Arbeitstag, aber auch volle Arbeitszeit an ausgewählten Tagen denkbar.

### **Adaptive Modelle**

Wenn auch gesetzliche Regelungen die Möglichkeiten einschränken, sind doch Arbeitszeitmodelle zu berücksichtigen, die die Festlegung von Arbeitszeiten von äußeren Einflüssen abhängig machen.

## **8.1.4 Recht**

Die rechtlichen Grundlagen für Arbeitszeitmodelle sind in Deutschland in verschiedenen Bundesgesetzen niedergelegt [REFA71]:

- das Arbeitsrecht basiert auf dem Grundgesetz (GG), dem Bürgerlichen Gesetzbuch (BGB), dem Handelsgesetzbuch (HGB) und der Gewerbeordnung (GewO),
- Tarifverträge nach dem Tarifvertragsgesetz (TVG) regeln Mindestarbeitsbedingungen,
- Arbeitszeiten gehorchen der Arbeitszeitordnung (AZO),
- einzelne Gruppen wird ein besonderer Schutz durch das Mutterschutzgesetz (MuSchG), das Schwerbehindertengesetz (SchwbG) und das Jugendarbeitsschutzgesetz (JugArbSchG) zugestanden,

- weiteren Einfluß haben das Berufsbildungsgesetz (BerufsbG) und das Betriebsverfassungsgesetz (BetrVG).

Betriebsvereinbarungen und Einzelverträge regeln das individuelle Verhältnis zwischen Arbeitgebern und Arbeitnehmern.

## 8.2 Analyse der Anforderungen

Eine Bibliothek zur Berechnung von Arbeitszeiten muß jedem Arbeitstag eine Menge von Arbeitszeiten zuordnen. So können folgende Anfragen beantwortet werden:

- Ist ein Termin Arbeitszeit? Diese Fragestellung taucht bei der Koordination von Terminen zwischen beliebigen Ressourcen auf. Bei der Festlegung von Synchronisationspunkten in einem Arbeitsprozeß müssen alle beteiligten Ressourcen verfügbar sein.
- Wann ist eine bestimmte Zeit verstrichen? Für einen Arbeitsvorgang wird eine Vorgangsdauer veranschlagt. Diese Zeitdauer muß auf die Arbeitszeit des Bearbeiters umgelegt werden. Als Ergebnis entsteht ein Zieltermin.
- Wieviel Arbeitszeit ist zwischen zwei Terminen verfügbar? Bei der Frage nach der Erfüllbarkeit einer Aufgabe mit einem bestimmten Mitteleinsatz müssen einzelne Ressourcen entsprechend geprüft werden.

Bei Berücksichtigung der Vielfalt möglicher Arbeitszeitmodelle ist es notwendig, Flexibilität in den Vordergrund zu stellen. Durch austauschbare Arbeitszeitmodelle kann dies gewährleistet werden.

Ein weiterer Punkt ist die Effizienz bei der Berechnung. Die Planung von Projekten beinhaltet Probleme mit a priori exponentieller Komplexität. Die verwendeten Algorithmen sind zwar optimiert, berechnen aber häufig mehrere Alternativen, um ein hinreichend gutes Ergebnis zu erreichen. Die Verrechnung von Arbeitszeiten ist ein Programmteil, der mehrfach durchlaufen wird, deshalb müssen die betreffenden Operationen besonders effizient sein.

Eine Berücksichtigung von Feiertagen ist bei der Festlegung von Arbeitszeitmodellen in der Regel nicht notwendig, allerdings ist sie theoretisch denkbar. Arbeitsfreie Tage müssen bei der Berechnung von Vorgangsdauern hingegen enbezogen werden.

## 8.3 Ergebnis

Der Kern des Konzeptes zur Arbeitszeitberechnung ist ein abstraktes Modell zur Erzeugung von Zeitintervallen (Arbeitszeiten) in einem bestimmten, datumsabhängigen Rhythmus. Einige Standardmodelle werden von der Bibliothek angeboten. Die Vielfalt der möglichen konkreten Arbeitszeitmodelle bedingt deren Implementierung bei Bedarf.

Ein konkretes Arbeitszeitmodell erzeugt für jeden Arbeitstag eine Menge von Arbeitszeiten. Für jede Arbeitszeit ist ein abzurechnender Pausenanteil angegeben. Ist dieser Pausenanteil leer, so ist die Arbeitszeit als effektiv anzusehen. Beim Rechnen mit Arbeitszeiten sind grundsätzlich nur effektive Arbeitszeiten zu verwenden.

Die Berechnung der arbeitsfreien Tage wird unabhängig vorgenommen (Kapitel 7). Der Benutzer der Bibliothek hat einen Arbeitszeitplan zu implementieren, indem er ein Kalendarium aller Arbeitstage aufstellt und dieses mit einem Arbeitszeitmodell verknüpft. Der Arbeitszeitplan steht dann für einzelne oder Gruppen von Humanressourcen zur Verfügung. Abb. 11 zeigt diesen Aufbau.

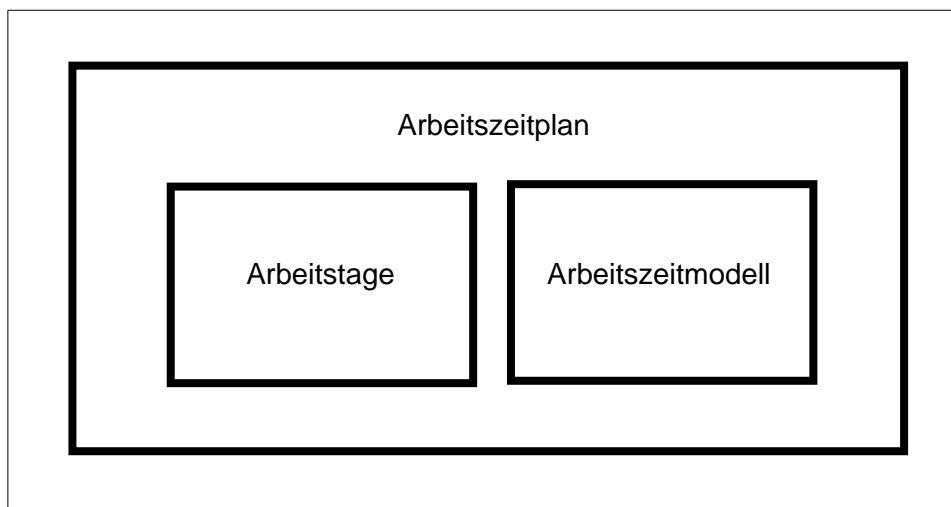


Abb. 11: Aufbau eines Arbeitszeitplans

Ein Überblick über die entstandenen Klassen befindet sich in Kapitel 9.



# Resultierendes Klassensystem

Das bestehende Klassensystem wurde in seiner Architektur und Implementierung an wesentlichen Stellen geändert. Die Klassen für Arbeitszeitarithmetik mußten vollständig ersetzt werden. Die Formatierung von Zeit- und Datumswerten wurde ganz aus den betroffenen Klassen heraus gelöst.

## 9.1 Übersicht

In Abb. 12 wird die Verwendungsbeziehung der einzelnen Teillösungen als Hierarchie dargestellt. An zentraler Stelle steht die Klassenkategorie für grundlegende Zeit- und

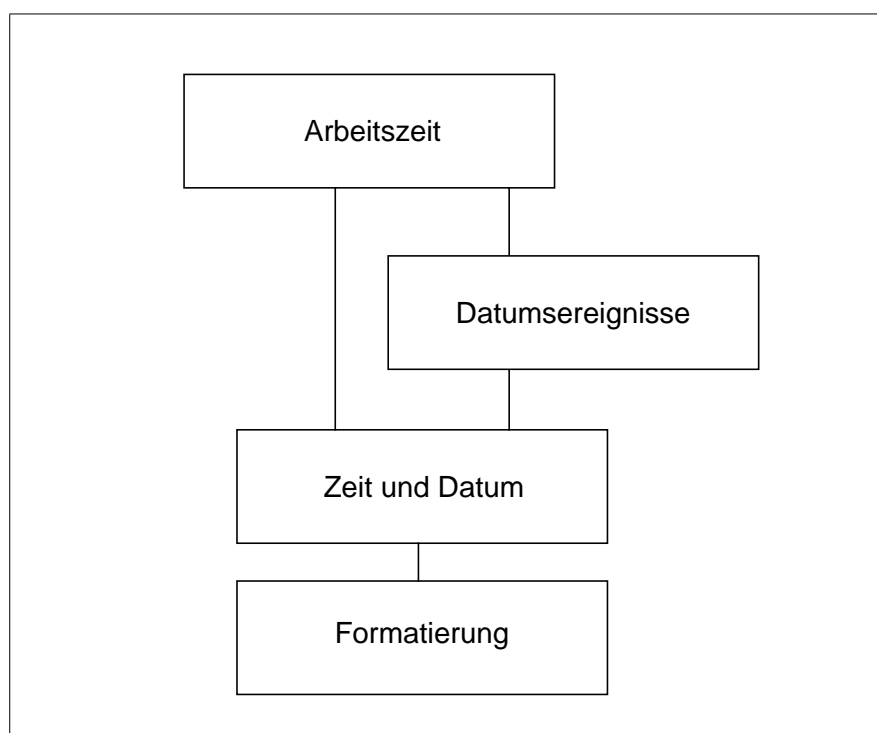


Abb. 12: Hierarchie der Klassenkategorien

Datumfunktionalität. Sie benutzt Klassen aus der Klassenkategorie für Formatierung.

Die Bereiche Datumsereignisse und Arbeitszeit stellen Anwendungen dieser Basisstrukturen dar. Teile der Kategorie Arbeitszeit verwenden die Kategorie Datumsereignisse.

## 9.2 Klassenkategorien

Bei den folgenden Klassendiagrammen wurde auf die Darstellung der Klassen aus der Basisbibliothek verzichtet.

### 9.2.1 Klassenkategorie für Zeit und Datum

Diese Klassenkategorie wird durch den EIFFEL-Cluster TIME\_GENERAL dargestellt. Die enthaltenen Klassen beschreiben grundlegende, anwendungsunabhängige Funktionalitäten. Abb. 13 enthält eine Darstellung der wichtigsten Beziehungen.

Die Klasse COMPARABLE stammt aus der Basisbibliothek und ermöglicht eine Ordnungsrelation auf Objekte einer abgeleiteten Klasse.

Die Klasse SCANNABLE stammt aus der Klassenkategorie für Formatierung und bringt die Möglichkeit der flexiblen Ein- und Ausgabe zu erbenden Klassen.

#### **Klasse DATE**

Diese Klasse beschreibt ein Tagesdatum, dargestellt als Julianischer Tag. Dafür kann jeder beliebige INTEGER-Wert verwendet werden. Es existiert eine einfache Arithmetik für die Addition und Subtraktion von beliebigen Tagen.

#### **Klasse TIME**

Diese Klasse beschreibt eine Tageszeit. Es können die Werte Stunde, Minute, Sekunde und Millisekunde gesetzt werden.

#### **Klasse DATE\_AND\_TIME**

Diese Klasse ist eine Verbindung der Klassen DATE und TIME. Sie beschreibt einen Termin, also eine bestimmte Uhrzeit an einem bestimmten Tag.

#### **Klasse DURATION**

Die Klasse ist eine abstrakte Basisklasse für die Definition wichtiger Funktionalitäten einer Zeitdauer.

#### **Klasse HMS\_DURATION**

Diese Klasse definiert eine Zeitdauer in Stunden, Minuten, Sekunden für die Verrechnung mit Tageszeiten (TIME).



### **Klasse DT\_DURATION**

Diese Klasse definiert eine Zeitdauer in Tagen, Stunden, Minuten, Sekunden für die Verrechnung mit allgemeinen Terminen (DATE\_AND\_TIME).

### **Klasse INTERVAL**

Diese Klasse hat einen generischen Parameter vom Typ **COMPARABLE**, um ein Intervall von beliebigen, vergleichbaren Klassen zu beschreiben. Sie bietet Operationen und Relationen auf Intervalle an [All83, MeNe93].

Die Klasse ist auch außerhalb der Bibliothek für beliebige andere Intervalle verwendbar.

### **Klasse GREGORIAN\_DICTIONARY**

Diese Klasse definiert datumsrelevante Bezeichnungen aus dem Gregorianischen Kalender.

### **Klasse GREGORIAN\_CALENDAR**

Diese Klasse ist ein wichtiges Werkzeug für Umrechnungen im Gregorianischen Kalendersystem.

### **Klasse GREGORIAN**

Diese abstrakte Klasse wird von anderen gregorianischen Klassen ererbt, um einen GREGORIAN\_CALENDAR als ONCE-Objekt zu liefern.

### **Klasse GREGORIAN\_DATE**

Diese Klasse definiert das gregorianische Datum in der Form Jahr, Monat und Tag..

### **Klasse GREGORIAN\_DATE\_AND\_TIME**

Diese Klasse spezialisiert DATE\_AND\_TIME für gregorianische Datumsangaben.

### **Klasse GREGORIAN\_DURATION**

Diese Klasse definiert eine Zeitdauer in Jahren, Monaten und Tagen.

### **Klasse GREGORIAN\_DT\_DURATION**

Diese Klasse definiert eine Zeitdauer in Jahren, Monaten, Tagen, Stunden, Minuten, Sekunden.

## 9.2.2 Klassenkategorie für Datumseignisse

Diese Klassenkategorie wird durch den EIFFEL-Cluster TIME\_DATE\_EVENTS dargestellt. Die enthaltenen Klassen beschreiben Datumseignisse und zugehörige Ver-

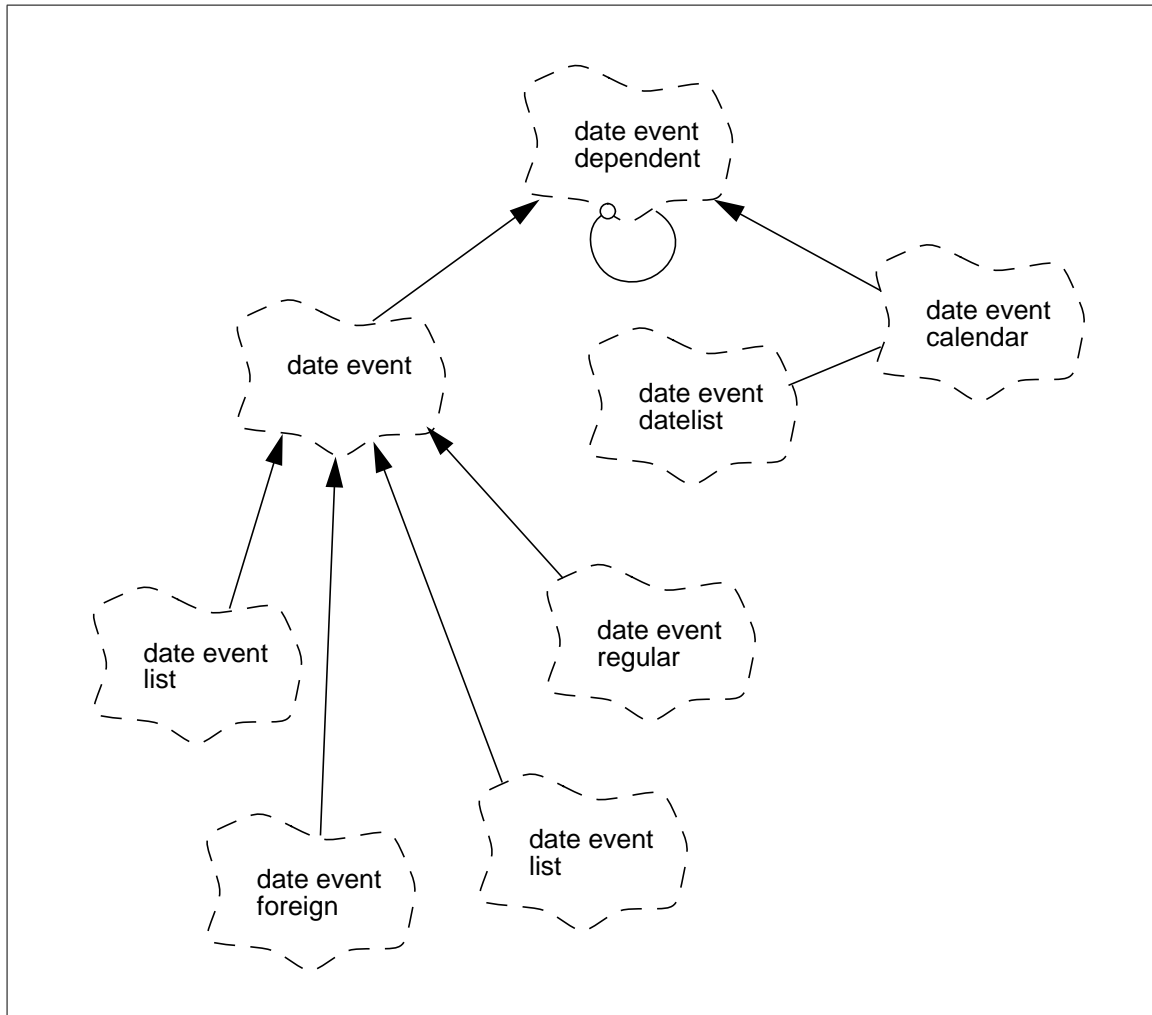


Abb. 14: Klassen in TIME\_DATE\_EVENTS

waltungsstrukturen (siehe Kapitel 7).

### Klasse DATE\_EVENT\_DEPENDENT

Diese abstrakte Basisklasse implementiert den Abhängigkeitsmechanismus. Beliebige weitere Klassen können durch Vererbung und Registrierung daran teilnehmen und werden bei Änderungen entsprechend benachrichtigt.

### **Klasse DATE\_EVENT**

Die abstrakte Basisklasse beschreibt das allgemeine Verhalten eines Datumseignisses. Eine Sortierung erfolgt über die Bezeichnung. Objekte dieser Klasse nehmen am Abhängigkeitsmechanismus teil.

### **Klasse DATE\_EVENT\_CALENDAR**

Diese Klasse ist für die Verwaltung von Mengen (Kategorien) von Datumseignissen zuständig. Eine Sortierung erfolgt über die Bezeichnung. Objekte dieser Klasse nehmen am Abhängigkeitsmechanismus teil.

### **Klasse DATE\_EVENT\_DATELIST**

Diese Klasse verwaltet eine Liste mit berechneten Datumsangaben aller Ereignisse eines bestimmten Zeitraums.. Der Umfang der Liste hängt von den konkreten Anforderungen ab, d.h. die Liste wächst, wenn ihre Grenzen überschritten werden.

### **Klasse DATE\_EVENT\_LIST**

Diese Klasse ist ein konkreter Nachfahre der abstrakten Klasse für Datumseignisse (DATE\_EVENT). Es wird eine beliebige Menge fester Termine gespeichert, die jedoch alle zu dem selben Ereignis gehören.

### **Klasse DATE\_EVENT\_REGULAR**

Die Klasse ist ein weiterer Nachfahre der abstrakten Klasse für Datumseignisse (DATE\_EVENT). Es wird eine Menge von Terminen beschrieben, die zwischen einem Anfangstermin und einem (optionalen) Endtermin liegen und im Abstand einer festen Zeitdauer aufeinander folgen.

### **Klasse DATE\_EVENT\_EASTER**

Diese Klasse ist ein konkreter Nachfahre der abstrakten Klasse für Datumseignisse (DATE\_EVENT). Es wird das Osterfest beschrieben, welches einer komplizierten Berechnungsvorschrift folgt und nicht durch einfache Parametrierung definiert werden kann.

### **Klasse DATE\_EVENT\_FOREIGN**

Diese Klasse ist ein konkreter Nachfahre der abstrakten Klasse für Datumseignisse (DATE\_EVENT). Es wird eine Menge von Terminen beschrieben, die von einem anderen Datumseignis abhängen und im gleichen Rhythmus wie dieses, jedoch um eine konstante Zeitdauer verschoben, stattfinden. Das zugrundeliegende fremde Ereignis wird in den Abhängigkeitsmechanismus eingebunden.

### 9.2.3 Klassenkategorie für Arbeitszeitarithmetik

Diese Klassenkategorie wird durch den Eiffel-Cluster TIME\_WORKTIME dargestellt. Die enthaltenen Klassen beschreiben Arbeitszeitarithmetik (siehe Kapitel 8).



Abb. 15: Klassen in TIME\_WORKTIME

#### Klasse WORKTIME

Diese Klasse ist ein Interval aus Zeitpunkten und beschreibt Arbeitszeit. Ein Pausenanteil wird ebenfalls ausgewiesen.

#### Klasse WORKTIME\_MODEL

Diese Klasse stellt ein abstraktes Arbeitszeitmodell dar. Konkrete Modelle werden abgeleitet und erzeugen für jedes definierte Datum eine Menge von Arbeitszeiten.

### **Klasse WORKTIME\_MODEL\_STATIC**

Diese Klasse definiert das statische Arbeitszeitmodell.

### **Klasse WORKTIME\_MODEL\_SHIFT**

Diese Klasse definiert das Schichtmodell

### **Klasse WORKTIME\_PLAN**

Diese Klasse definiert einen Arbeitszeitplan, der verschiedene Berechnungen auf Basis einer Festlegung der Arbeitstage und des Arbeitszeitmodells erlaubt.

## **9.2.4 Klassenkategorie für Formatierung**

Diese Klassenkategorie wird durch den EIFFEL-Cluster TIME\_SCANNING dargestellt. Die enthaltenen Klassen beschreiben formatierte Ein- und Ausgabe von Zeit- und Datumswerten (siehe Kapitel 6).

### **Klasse SCANNABLE**

Diese abstrakte Basisklasse dient der Definition von Funktionalitäten des Erkennens von Eingaben und Formatieren von Ausgaben als Zeichenketten.

### **Klasse SCANNER**

Diese abstrakte Basisklasse dient der Definition von Formatbeschreibungen für die Ein- und Ausgabe. Ein Scanner enthält eine Liste von Terminalen.

### **Klasse SCANNER\_YMD**

Diese abstrakte Basisklasse ist von SCANNER abgeleitet und dient der Beschreibung von Formaten für Klassen, die über Jahr, Monat und Tag definiert sind. Sie beinhaltet drei Variable, die von geeigneten Terminalen gefüllt werden.

### **Klasse SCANNER\_YMD\_ISO**

Diese Klasse beschreibt das ISO-Format YYYY-MM-DD für Datumseingaben.

### **Klasse SCANNER\_YMD\_GERMAN**

Diese Klasse beschreibt das ausgeschriebene deutsche Format für Datumseingaben wie z.B. „13. März 1997“.

### **Klasse SCANNER\_JD**

Diese Klasse beschreibt das Format für Julianische Tage.

### **Klasse SCANNING\_TERMINAL**

Diese Klasse beschreibt ein abstraktes Terminal, d.h. ein syntaktisches Objekt.

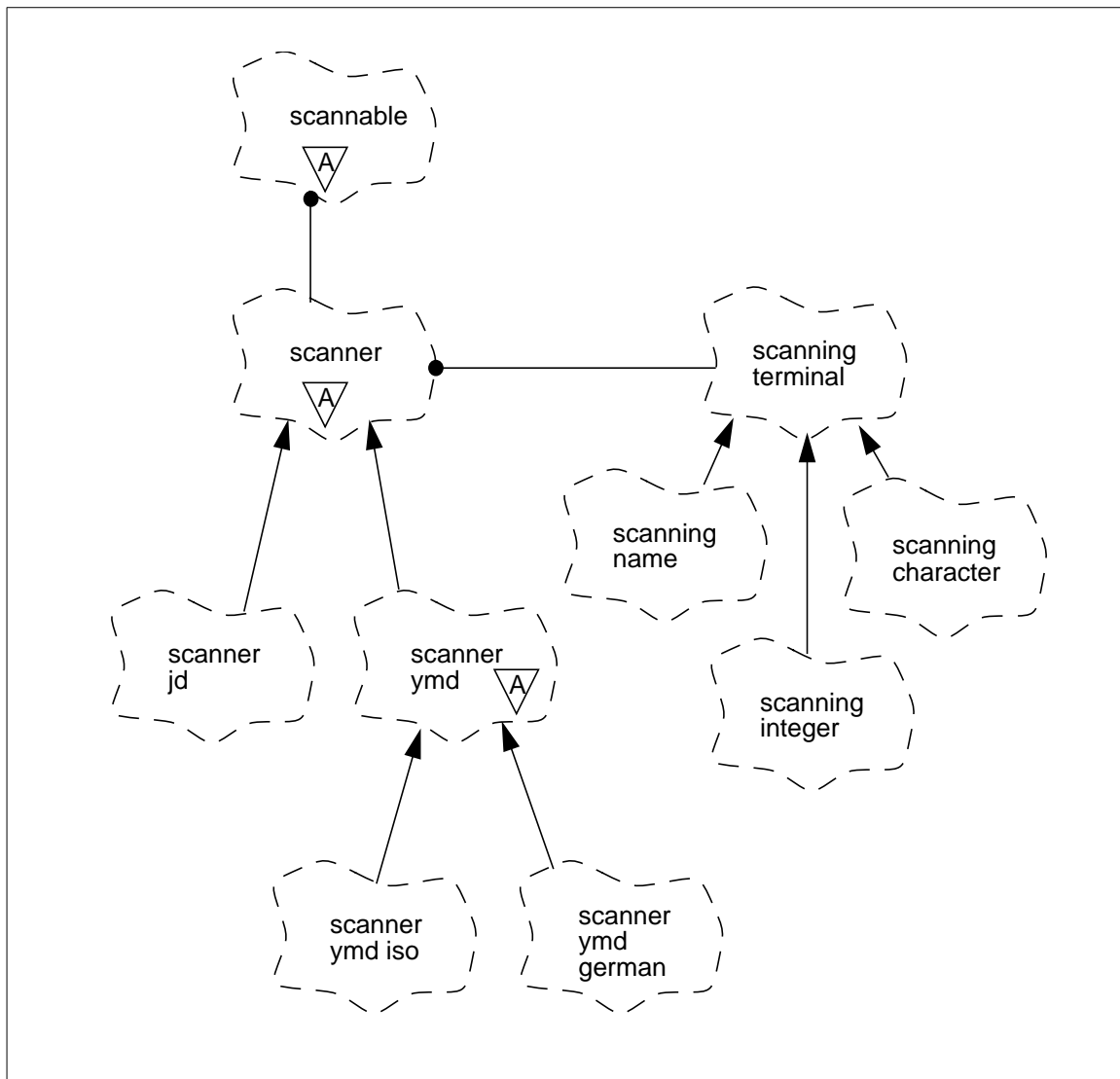


Abb. 16: Klassen in TIME\_SCANNING

Das Terminal kann optional sein, dann wird es gegebenenfalls vom Scanner ignoriert, wenn er eine Zeichenkette sonst nicht erkennen kann.

### Klasse SCANNING\_NAME

Diese Klasse beschreibt ein konkretes Terminal welche Namen erkennt.

### Klasse SCANNING\_INTEGER

Diese Klasse beschreibt ein konkretes Terminal welches Zahlen erkennt.

### Klasse SCANNING\_CHARACTER

Diese Klasse beschreibt ein konkretes Terminal welches Zeichen erkennt.



## Zusammenfassung

Neben einer Restrukturierung der Bibliothek nach den Gesichtspunkten Klarheit und Semantik wurde eine Reimplementierung zur Steigerung der Effizienz durchgeführt.

Aus der Erweiterung der Formatierung von Zeit- und Datumswerten, der automatischen Berechnung von Feiertagen und der Ersetzung der Arbeitszeitarithmetik durch ein flexibles, dynamisches Konzept entstanden drei zusätzliche Module.

Die Berechnung von Feiertagen bedingte eine Verallgemeinerung des Datumsbegriffs. So entstand die Ablösung von der rein gregorianischen Sichtweise, welche in allen betrachteten Bibliotheken vertreten ist. Der Julianischer Tag nach Scaliger als gemeinsame Basis für die Konvertierung zwischen Kalendersystemen erwies sich zudem als effiziente Speicherdarstellung eines Datums.

Im Rahmen der Feiertagsberechnung wurde ein einfaches Mengenmodell mit einem transparenten Abhängigkeitsmechanismus (change-update) verknüpft.

Für die Realisierung der Arbeitszeitarithmetik wurde ein abstraktes Arbeitszeitmodell entwickelt. Die Anpassung an die reale betriebliche Welt kann gegebenenfalls durch Vererbung und Erweiterung geschehen. Zusätzliche Leistungsanforderungen können später aufgesetzt werden.

### Fazit

Leitgedanke war immer, die Bibliothek durch pragmatische Erweiterungen leistungsfähiger zu machen anstatt sie zu verkomplizieren. Die einfache Anwendung der Bibliothek stand im Vordergrund.

Nicht gelöst werden konnte das Problem dynamische Berechnungsvorschriften für Datumseignisse zu definieren, da dieses eine objektorientierte Modellierung allgemeiner Algorithmen erfordert hätte, was den Rahmen der Arbeit übertraf.

Zukünftige Versionen der EIFFEL-Basisbibliothek erlauben möglicherweise mehrere Parser gleichzeitig, daß auch im Bereich der Ein- und Ausgabe von Zeit- und Datumswerten neue Wege gegangen werden können.

Der entwickelte Formatierungsmechanismus wurde nicht auf seine allgemeine Verwendbarkeit für andere Gebiete der Anwendungsentwicklung geprüft.

Das Auffinden von weiteren Standard-Arbeitszeitmodellen ist ebenso offen.

Analyse und Design wurden nach der Methode von Grady Booch durchgeführt. Die bewußte Aufteilung in Phasen schafft beim Entwickler eine Perspektive mit mehr Abstand von der Implementierung. Der notwendige Kompromiß zwischen Abstraktion und Pragmatik erfordert ein eher intuitives Vorgehen, das nicht formalisiert werden kann. Hier konnte der Entwickler seine Erfahrung einbringen.

Die Booch-Notation ist eine einfache, klare Darstellungsmethode für Klassenhierarchien. Sie ist im wesentlichen vergleichbar mit anderen Notationen und wird deshalb schnell verinnerlicht.

Die Implementierung in EIFFEL liefert schnelle Ergebnisse bei hoher Qualität des Codes. Die Kapselung der Klassen ist sehr gut und wird auch von der Entwicklungsumgebung optimal unterstützt. Vor- und Nachbedingungen sowie Klasseninvarianten helfen bei der Verifikation des Ergebnisses. Die Implementierung wird zur Nebensache und setzt Kapazitäten für Analyse und Design frei.

### **Ausblick**

Die umfangreichen Änderungen an der Bibliothek, die auch die Architektur und die Schnittstellen betrafen, machen Änderungen in PLASMA notwendig.

Da die objektorientierte Entwicklung von Software als Spirale gesehen wird, bleibt die Resonanz auf diese Windung und zukünftige Anforderungen abzuwarten.

# Anhang A

## Feiertage

### Von Ostern abhängige christliche Feiertage

Als Referenzdatum zur Berechnung abhängiger Feiertage wird immer der Ostersonntag verwendet.

- Beichtdienstag:** Auch Pfannkuchentag. Ein Tag der Buße zur Vorbereitung auf die Fastenzeit. Volkstümlich in manchen Regionen mit Vergnügungen und Pfannkuchenessen vor dem Fasten begangen (Fastnacht). Termin ist 47 Tage vor Ostern.
- Aschermittwoch:** Beginn der Fastenzeit/Passionszeit, so benannt nach dem katholischen Brauch, im Gedenken an die Sterblichkeit des Menschen die Stirn der Gläubigen mit einem Aschenkreuz zu versehen. Termin ist 46 Tage vor Ostern.
- Palmsonntag:** Der erste Tag der Heiligen oder Karwoche erinnert an den triumphalen Einzug Jesu in Jerusalem. Palmwedel (welche die Menschen damals zur Begrüßung schwangen) werden zuweilen in den Kirchen verteilt, und Kreuze aus Palmstroh schmücken manches Haus. Termin ist 1 Woche vor Ostern. Zwischen Palmsonntag und Ostersonntag liegt die Karwoche, die ernsthafteste Woche im christlichen Jahr, in der Christen der letzten Woche im Leben Jesu gedenken.
- Gründonnerstag:** Erinnert an die Einsetzung des Heiligen Mahles, der Eucharistie, des Abendmahles und an Jesu Gebot bei der Fußwaschung. Termin ist 3 Tage vor Ostern.
- Karfreitag:** Erinnert an die Kreuzigung Jesu, das Beispiel höchster Selbstopferung. Termin ist 2 Tage vor Ostern.
- Karsamstag:** Stiller Samstag. Mit Zeiten der Stille und Gottesdiensten, ohne Glocken und Orgel begangen, gilt der Tag der Besinnung zuweilen im Gedenken an die Taufe, das Ende der Fastenzeit oder Passionszeit. Termin ist 1 Tag vor Ostern.
- Ostermontag:** Termin ist 1 Tag nach Ostern.

- Christi Himmelfahrt:** Dieser Feiertag erinnert an die letzte irdische Erscheinung des auferstandenen Christus, gedenkt seiner darauf folgenden Himmelfahrt und seiner irdische Begrenzungen überschreitenden Gegenwart für Christen. Termin ist 39 Tage nach Ostern.
- Pfingsten:** So benannt nach dem jüdischen Fest, an welchem die Nachfolger Jesu zuerst den Heiligen Geist empfangen und das Evangelium verkündeten, deshalb oft als „Geburtstag der Kirche“ bezeichnet. Termin ist 7 Wochen nach Ostern (So/Mo).
- Dreieinigkeitsfest:** Auch Trinitatis. Gewidmet an die christliche Lehre der Trinität, nach der Gott unteilbar Einer ist, doch offenbart in den klar unterschiedenen Gestalten von Vater, Sohn und Heiligem Geist. Termin ist 8 Wochen nach Ostern.
- Fronleichnam:** Katholische Fest zur Feier des Eucharistiesakraments, fröhlicher gestaltet als das Gedenken an die Einsetzung des Abendmahls am Gründonnerstag. Termin ist 60 Tage nach Ostern.

#### **Nicht von Ostern abhängige christliche Feiertage**

- Neujahr:** Marienitag, auch Beschneidung/Namengebung Jesu. Erinnert Christen an die Herkunft Jesu und der christlichen Religion aus der jüdischen Tradition. Termin ist der 1. Januar.
- Erscheinungsfest:** Epiphania, Fest der ersten Erscheinung Christi, zuweilen begangen als Anbetung der Weisen, Dreikönigstag, Taufe Christi oder Tag des Wunders zu Kana. Termin ist der 6. Januar.
- Lichtmeß:** Feier der Darstellung Jesu im Tempel. Der Name kommt von Lichterprozessionen, die in vielen Kirchen begangen werden. Das Licht ist Symbol für Jesus als das Licht für die Völker. Termin ist der 2. Februar.
- St. Josephstag:** Fest zu Ehren des Mannes, von dem die Kindheitsgeschichten Jesu berichten, der zusammen mit Maria für die Erziehung Jesu verantwortlich war. Termin ist der 19. März.
- Mariä Verkündigung:** Fest der Ankündigung des Engels Gabriel an Maria, daß sie ein Kind bekommen wird. Termin ist der 25. März.
- Verklärung:** Erinnert an das Leuchten von Jesu Antlitz und Kleidern, eine Reminiszenz an die Geschichte vom leuchtenden Antlitz Moses nach seinem Gespräch mit Gott. Termin ist der 6. August.

---

<b>Mariä Himmelfahrt:</b>	Tag Unserer Hohen Frau, feiert die Aufnahme von Leib und Seele Mariens in den Himmel. In vielen katholischen Gemeinden werden Prozessionen veranstaltet. Termin ist der 15. August.
<b>Mariä Geburt:</b>	Geburt Marias, der Mutter Jesu. Termin ist der 8. September.
<b>Erntedankfest:</b>	Nach regionaler Überlieferung mit Gottesdiensten gefeiert (die Daten variieren), für die Altar und Kirche mit Früchten aus Feld und Garten geschmückt sind, die später an Alte und Bedürftige verteilt werden. Termin ist regional verschieden im September/Oktober.
<b>Reformationstag</b>	Evangelischer Feiertag. Termin ist der 31. Oktober.
<b>Allerheiligen:</b>	Eine Gelegenheit, alle Heiligen anzurufen und denen zu danken, für die kein eigener Gedenktag besteht. Termin ist der 1. November.
<b>Allerseelen:</b>	An diesem Tag wird besonders der Dahingegangenen gedacht und für sie gebetet. Termin ist der 2. November.
<b>Buß- und Betttag:</b>	Der Mittwoch vor dem letzten Sonntag im Kirchenjahr, also zwischen 16. und 22. November, da der erste Sonntag des neuen Kirchenjahres der 1. Advent ist, der letzte Sonntag des alten Kirchenjahres also zwischen dem 20. und dem 26. November angesiedelt ist.
<b>1. Advent:</b>	Beginn des Kirchenjahres. Vier Sonntage vor dem Christfest dienen der Vorbereitung auf diesen Höhepunkt. Adventskalender und Adventskränze bezeichnen den Übergang vom Dunkel ins Licht. Termin ist der Sonntag an oder nach dem 27. Nov.
<b>2. Advent:</b>	Termin ist der Sonntag an oder nach dem 4. Dez
<b>Mariä Empfängnis:</b>	Die unbefleckte Empfängnis Mariens. Feier der katholischen Lehre, daß Maria frei von der Erbsünde empfangen und geboren wurde, um das Jesuskind sündenfrei auszutragen. Termin ist der 8. Dezember.
<b>3. Advent:</b>	Termin ist der Sonntag an oder nach dem 11. Dez
<b>4. Advent:</b>	Termin ist der Sonntag an oder nach dem 18. Dez
<b>Heiligabend:</b>	Beginn des Christfestes. Termin ist der 24. Dezember.
<b>1. Weihnachtstag:</b>	Feier der Geburt Jesu. Termin ist der 25. Dezember.
<b>2. Weihnachtstag</b>	Zweiter Feiertag zur Geburt Jesu. Termin ist der 26. Dezember.

### **weltliche Feiertage**

Valentinstag: 14. Februar

Muttertag: 2. Sonntag im Mai. Wenn dieser der Pfingstsonntag ist, dann schon eine Woche früher

### **nationale Feiertage**

Tag der dt. Einheit: 3. Oktober

Maifeiertag: 1. Mai

### **internationale Feiertage**

Es werden nachstehend einige amerikanische Feiertage aufgeführt:

New Year's Day: 1. Januar

President's Day: dritter Montag im Februar

Armed Forces Day: dritter Samstag im Mai

Memorial Day: letzter Montag im Mai

Flag Day: 14. Juni

Independence Day: 4. Juli

Labor Day: erster Montag im September

Columbus Day: zweiter Montag im Oktober

Election Day: Dienstag an oder nach dem 2. November

Veterans Day: 11. November

Thanksgiving Day: vierter Donnerstag im November

# Literatur

- ASU88** Aho, Sethi, Ullman  
*Compilerbau, Teil 1*  
Addison-Wesley, 1988
- All83** James F.Allen  
*Maintaining Knowledge about temporal intervals*  
Communications of the ACM, 26 (11): 832-843, 1983
- Ber95** *Das Bertelsmann Lexikon*  
Verlagshaus Stuttgart GmbH, 1995
- Boo94** Grady Booch  
*Objectoriented Analysis & Design 2. Edition*  
Benjamin/Cummings Publishing Company, Readwood City, CA 1994
- CFD88** Cornelius Caesar, Georg Fritsche, Axel Dittes  
*D wie Denkste, Kalenderabhängige C-Funktionen*  
c't, Magazin für Computertechnik, Heise Verlag Hannover, 9/88
- Cost96** Hilde Cost  
*Sonntagsarbeit: „Und am siebten Tage sollst Du ruhen“*  
Magazin Wirtschaft 12/96  
Industrie- und Handelskammer, Region Stuttgart
- CoYo90** Peter Coad, Ed Yourdon  
*Object-Oriented Analysis*  
Prentice Hall International (UK), Hertfordshire, 1990
- CoYo91** Peter Coad, Ed Yourdon  
*Object-Oriented Design*  
Prentice Hall International (UK), Hertfordshire, 1991
- EAWRE96** Europ. Arbeitsgemeinschaft für Weltreligionen in der Erziehung  
*Kalender der Feste der Religionen 1997*  
Wartburg Verlag Weimar, 1996
- EnBri71** *Encyclopaedia Britannica, 1971*  
Encyclopaedia Britannica Inc., London
- EnHe88** Engesser, Hermann [Hrsg.]  
Claus, Schwill [Bearb.]  
*DU DEN Informatik*  
Dudenverlag, Mannheim Wien Zürich, 1988

- Gam95** Erich Gamma et. al.  
*Design Patterns: elements of reusable object-oriented software*  
Addison Wesley, Reading, Massachusetts, 1995
- INFORMIX** *INFORMIX SQL-Sprachbeschreibung 5.0*  
Informix Deutschland GmbH
- ISO88** *ISO 8601:1988 Date/Time Representations.*
- KaGo77** K. M. Kahn, A. G. Gorry  
*Mechanizing temporal Knowledge*  
Artificial Intelligence 9, 2, 87-108, 1977
- Kna76** *Knaurs Lexikon*  
Knaur Verlag, 1976
- KST96** Reinhard Kirste, Herbert Schulze, Udo Tworuschka  
*Die Feste der Religionen*  
Gütersloher Verlagshaus, 1996
- Mar95** Elfe Marzell  
*Entwicklung von Mechanismen zur Verwaltung von Hardware- und Humanressourcen in offenen integrierten CAE-Systemen*  
Universität Stuttgart, Fakultät Informatik, 1995  
Studienarbeit 1320
- MeNe93** Bertrand Meyer, Jean-Marc Nerson  
*Object-oriented Applications*  
Prentice Hall International (UK), Hertfordshire, 1993
- Mey90** Bertrand Meyer  
*Objektorientierte Software-Entwicklung*  
Carl Hanser Verlag, München, 1990
- Mey92** Bertrand Meyer  
*Eiffel: The Language*  
Prentice Hall International (UK), Hertfordshire, 1992
- Mey94** Bertrand Meyer  
*Reusable software: the base object oriented component libraries*  
Prentice Hall International (UK), Hertfordshire, 1994
- MFC96** *Microsoft Foundation Classes, Visual C++ 4.2, Online Help*  
Microsoft Corporation, 1996
- ORACLE** *ORACLE 7 SQL Language Reference Manual*  
Oracle Corporation, Redwood City (USA)
- REFA71** Verband für Arbeitsstudien - REFA - e.V. Darmstadt  
*Methodenlehre des Arbeitsstudiums*  
Carl Hanser Verlag, München, 1971

- REFA74**    Verband für Arbeitsstudien - REFA - e.V. Darmstadt  
*Methodenlehre der Planung und Steuerung*  
Carl Hanser Verlag, München, 1974
- Ryba96**    Michael Ryba  
*Planung und Durchführung methodischer Entwurfsaktivitäten*  
Verlag Dr. Kovac, Hamburg, 1996
- RW96**      *Tools.h++, Version 7*  
*Foundation Class Library for C++ Programming*  
Rogue Wave Software
- Wec88**     Siegfried Weckmann  
*C wie Calendar*  
c't, Magazin für Computertechnik, Heise Verlag Hannover, 3/88



# Internet-Referenzen

Das Internet war bei der Literaturrecherche sehr hilfreich. Als Einstieg empfiehlt sich die Verwendung von Suchdiensten wie YAHOO oder ALTAVISTA. Gute Suchwörter sind z.B. eiffel, calendar, date of easter. Bei hinreichend genauer Einschränkung des Suchbegriffs hält sich die Zahl der gefundenen Referenzen in Grenzen.

Als Quellenangabe ist das Netz nicht geeignet, da die notwendige Verlässlichkeit und Dauerhaftigkeit der Informationen nicht garantiert werden kann. Als Ausgangspunkt für weiteres Forschen sind die WWW-Seiten jedoch eine interessante Quelle

In der Hoffnung, die Referenzen mögen lange gültig bleiben, hier eine kleine Auswahl:

<ftp://login.dknet.dk/pub/ct/calendar.faq>

Häufig gestellte Fragen (FAQ) zu Kalendern, von Claus Tondering.

<http://astro.nmsu.edu/~lhuber/leaphist.html>

Calendars, by L. E. Doggett

Eine ausführliche Darstellung kalenderbezogener Themen.

<http://www.genealogy.org/~scottlee/calconvert.cgi>

Calendar Conversions, Scott E. Lee

<http://cssa.stanford.edu/~marcos/ec-cal.html>

Calculation of the Ecclesiastical Calendar

<http://cssa.stanford.edu/~marcos/ushols.html>

The American Secular Calendar

Eine Darstellung amerikanischer Feiertage.

<http://www.smiley.cy.net/bdecie/>

The World Wide Holiday and Festival Page

<http://sadira.gb.nrao.edu:80/~rfisher/Ephemerides/times.html>

Astronomical Times

<ftp://uni-erlangen.de/pub/doc/ISO/ISO8601.ps.Z>

ISO 8601:1988 Date/Time Representations.

Gary Houston, 1993-01-31

<http://www.ft.uni-erlangen.de/~mskuhn/iso-time.html>

Eine Zusammenfassung des ISO-Standards für Zeit und Datum.

Markus Kuhn, 1996-10-04

<http://www.aavso.org/jd.html>

Explanation of Julian Date, Grant Foster, AAVSO

<http://tycho.usno.navy.mil/mjd.html>

Modified Julian Date

Gernot M. R. Winkler, formerly with U.S. NAVAL OBSERVATORY

<http://www.eiffel.com>

Die EIFFEL-Homepage, ein Muß für jeden EIFFEL-Interessierten. Hierüber ist auch Bertrand Meyer, der Entwickler der Programmiersprache, erreichbar.

<ftp://ftp.cs.cf.ac.uk/pub/eiffel/Tower>

anonymous ftp server in the Department of Computer Science in the University of Wales, Cardiff.

Hier kann die Bibliothek aus Cardiff heruntergeladen werden.

<http://cssa.stanford.edu/htbin/ushols.cgi>

American Holidays and Dates of Interest

# Glossar

Allerheiligen	All Saint's Day
Allerseelen	All Soul's Day
Aschermittwoch	Ash Wednesday
Beichtdienstag	Shrove Tuesday, Pancake Day
Beschneidung Jesu	Circumcision of Jesus
Christi Himmelfahrt	Ascension Day
Dreieinigkeitsfest	Trinity Sunday
Erntedankfest	Harvest Festival
Erscheinungsfest	Epiphany
Fronleichnam	Corpus Christi
Gründonnerstag	Maundy Thursday, Thursday of Holy Week
Heiligabend	Christmas Eve
Karfreitag	Good Friday, Friday of Holy Week
Karsamstag	Holy Saturday, Easter Eve
Lichtmeß	Candlemas
Mariä Empfängnis	Immaculate Conception of The Blessed Virgin Mary
Mariä Geburt	Birth of The Blessed Virgin Mary
Mariä Himmelfahrt	Assumption of The Blessed Virgin Mary
Mariä Verkündigung	Annunciation, Lady Day
Mariätag	Solemnity of Mary
Neujahr	New Years Day
Ostersonntag	date of easter, Easter Sunday
Palmsonntag	Palm Sunday
Pfingsten	Pentecost, Whitsuntide
Verklärung	Transfiguration



Hiermit versichere ich, daß ich diese Arbeit selbständig verfaßt und bei der Erstellung nur die angegebenen Hilfsmittel verwendet habe.

---

Johannes Fasolt

