Anwendung der BON

Problemstellung

Implementierung eines einfachen POP3-Clients

Anforderungsbeschreibung

Es soll ein einfacher POP3-Kommandozeilen-Client entworfen werden, welcher über eine Internet-Verbindung EMail von einem POP3-Server holt und diese auf dem Client-Rechner in einzelnen Dateien ablegt. Der Ablauf sieht dabei folgendermaßen aus:

Meldung des POP3-Servers:

```
+OK aixrs1.hrz.uni-essen.de POP3 3.3(18) w/IMAP2 client (Comments to
MRC@CAC.Washington.EDU) at Fri, 23 Jan 1998 17:08:20 +0100 (MEZ)
```

Anmeldung:

```
user USERNAME
+OK User name accepted, password please
pass PASSWORD
+OK Mailbox open, 1 messages
```

Information über die Anzahl, Nummern und Größen der Mails anfordern:

```
list
+OK Mailbox scan listing follows
1 1314
.
```

Mail anfordern:

```
retr 1
+OK 1314 octets
Received: from uni-essen.de (aixrs5.hrz.uni-essen.de [132.252.3.15])
by aixrs1.hrz.uni-essen.de (8.8.5/8.7) with ESMTP id RAA18690 for
.
.
.

.
```

Verbindung beenden:

```
quit
+OK Sayonara
```

Die Verbindung selbst wird über eine TCP/IP-Socket-Verbindung hergestellt, die IP-Adresse des Servers ist hierbei konstant und die Port-Nummer ist 110. Die Benutzerdaten (Name und Paßwort) sind ebenso fest vorgegeben. Die eingegangenen Mails werden in dem Verzeichnis, in dem der POP3-Client gestartet wurde, abgelegt. Die Dateinamen sollen dabei fortlaufende Nummern sein. Es werden immer alle Mails geholt, egal, ob diese in einer vorigen Session schon geladen wurden.

Finden der Systemgrenze

Das Umfeld des Systems stellt sich folgendermaßen dar: Das Programm wird per Kommandozeile aufgerufen, kommuniziert über eine Socket-Verbindung mit dem Server und legt die Mails in einzelnen Dateien im aktuellen Verzeichnis ab. Die Schnittstellen des Systems zum Umfeld stellen gleichzeitig initiale Subsysteme dar, wobei die Schnittstelle zur Kommandozeile aufgrund geringer Komplexität in diesem System kein eigenständiges Subsystem ist:

POP3-Client: Cluster mit den Klassen für das eigentliche System

Dateisystem: Interface für das Dateisystem

Socket-Verbindung: Ansteuerung des Socket-Interfaces

Metaphern (unabängige Konzepte) aus dem Anwendungsgebiet (Vokabular des Benutzers) sind unter anderem:

- Datei
- Mail
- Server

Der einzige Benutzungsfall (Use-Case) ist oben geschildert und beschreibt den Ablauf der Verbindungsaufnahme, Datenübertragung und Verbindungsbeendigung.

Finden potentieller Klassen

Ausgehend von den Metaphern existieren noch weitere unabgängige Konzepte im Anwendungsgebiet, die potentielle Klassen darstellen:

- Benutzer
- Verzeichnis
- Verbindung
- Mailbox

Auswahl und Gruppierung von Klassen

Aus den aufgezählten Klassen und Metaphern lassen sich folgende Analyseklassen bilden:

Datei: Kann erzeugt und in ein Verzeichnis gelegt werden, Zeichen können hineingeschrieben werden, trägt einen Namen

Verzeichnis: Liste von Dateien

Verbindung: Kann zu einer bestimmen TCP/IP-Adresse aufgebaut und wieder geschlossen werden, Zeichen können herausgelesen werden

Mailbox: Enthält die Nummern und Längeninformation der einzelnen Mails, kann Mails über eine Verbindung zeichenweise auslesen

POP3-Client: Koordiniert Verbindungsaufbau, Lesen der Mailbox-Daten, Anlegen der Dateien, Lesen der Mails und Schließen der Verbindung

Die Konzepte *Benutzer* und *Server* sind hier keine eigenständigen Klassen, da sich ihnen kein besonderes Verhalten zuordnen läßt. In einem größerem System oder einer Weiterentwicklung wäre es aber denkbar, diese Konzepte als Klassen zu modellieren, um z.B. Benutzerdaten konfigurierbar zu machen oder Server-Adressen zu verwalten.

Den oben identifizierten Subsystemen können die Klassen wie folgt zugeteilt werden:

POP3-Client: Mailbox, POP3-Client

Socket-Verbindung: Verbindung

Dateisystem Datei, Verzeichnis

Kollaboration zwischen Klassen (also Kunde-Anbieter-Beziehungen) bestehen zwischen:

POP3-Client und Verbindung, Datei, Verzeichnis sowie Mailbox

Verzeichnis und Datei

Mailbox und Verbindung

Definition der Klassen, Beschreibung des Systemverhaltens, Definition exportierter Features

Wird hier nicht betrachtet

Verfeinern des Systems

An dieser Stelle treten zunehmend Design-Klassen in den Vordergrund, u.a. aus folgenden Bereichen:

- Zeichen, Zeichenketten
- Pattern-Matching
- Arrays, Listen, Mengen

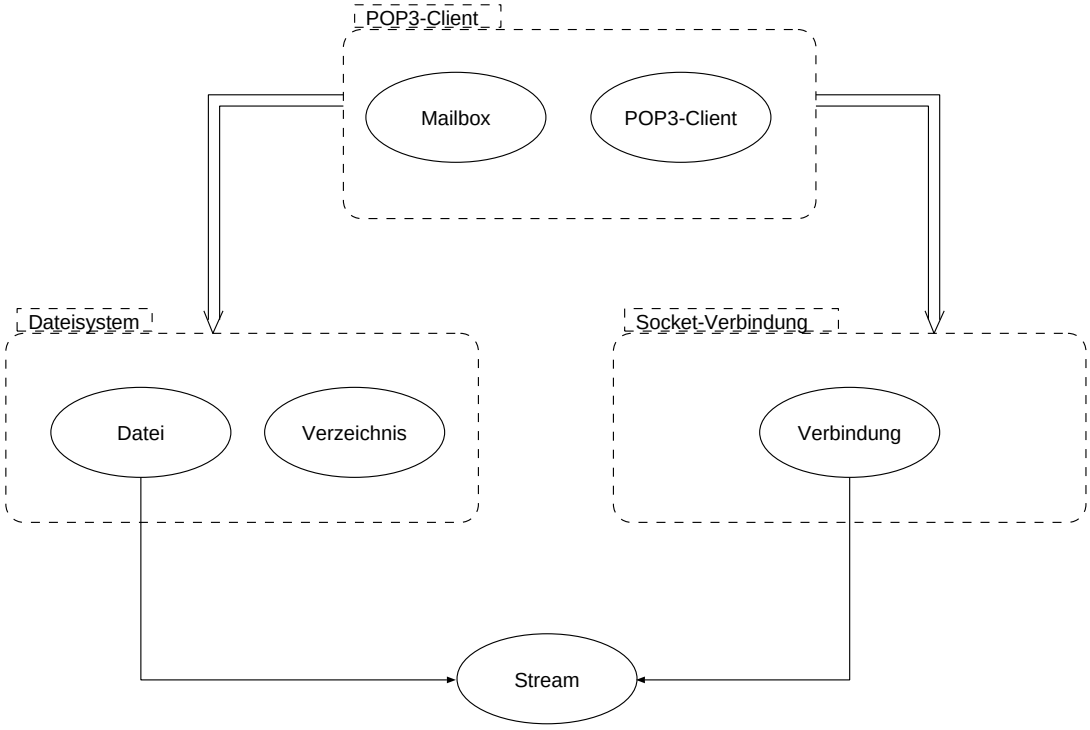
Generalisierung

Bei der Generalisierung fällt auf, daß die Klassen *Datei* und *Verbindung* insofern gemeinsames Verhalten aufweisen, als daß sie Zeichenströme beschreiben. Demnach kann hier eine neue Oberklasse *Stream* gebildet werden, die dieses Verhalten beschreibt.

Komplettierung und Review

Wird hier nicht betrachtet

Cluster-Diagramm



Problemstellung

Entwurf eines Auftragsbearbeitungssystems

Anforderungsbeschreibung

Es soll ein System entworfen werden, welches bei einem Versender per Brief, Fax, EMail oder Telefon eingehende Bestellungen verwaltet. Bestellungen sollen dabei neu angelegt, bearbeitet und gelöscht werden können. Die Bestelldaten umfassen die Kundendaten (Name, Anschrift), die bestellte Ware (Anzahl, Bezeichnung) und die Zahlungs- und Lieferkonditionen. Daneben werden keine Kunden- oder Artikeldaten im System gespeichert. Das System soll eine graphische Benutzerschnittstelle erhalten, es wird auf mehreren Rechnern gleichzeitig laufen und die Daten werden zentral in einer Datenbank gehalten.

Beschreiben sie die Schritte

- Finden der Systemgrenze
- Finden potentieller Klassen
- Auswahl und Gruppierung der Klassen
- Verfeinern des Systems
- Generalisierung

Problemstellung

Entwurf eines Web-Servers

Anforderungsbeschreibung

Der Web-Server dient der Übermittlung von Dateien, welche durch einen Anfragebefehl angefordert werden. Die Kommunikation mit den Client-Programmen erfolgt dabei über eine TCP/IP-Socket-Verbindung. Der Web-Server empfängt die Anfragebefehle über diese Verbindung und verschickt darüber auch die angeforderten Dateien. Ein Anforderungsbefehl besteht aus einer einzelnen Zeile mit folgendem Format:

```
GET <Dateiname>
```

Die Antwort des Servers besteht mindestens aus einer Zeile mit einer Statusmeldung. Bei erfolgreicher Anfrage (d.h. wenn die angeforderte Datei gefunden wurde), folgt darauf eine Zeile mit der Beschreibung des MIME-Typen der Datei sowie der Inhalt der Datei:

Erfolgsfall:

```
OK
Content-Type: <MIME-Type>
<Dateiinhalt>
```

Datei nicht gefunden:

```
Not Found
```

Die MIME-Typen der zurückgegebenen Dateien werden nach der Endung der angeforderten Datei zugeordnet und in einer Konfigurationsdatei abgelegt. Diese Konfigurationsdatei hat somit folgenden Aufbau:

```
<Endung1> <MIME-Type1>
<Endung2> <MIME-Type2>
...
```

Beispiel:

```
.html text/html
.gif image/gif
```

Problemstellung

Entwurf eines EMail-Servers

Anforderungsbeschreibung

Der EMail-Server dient dem autorisierten Zugriff auf die Mailboxen verschiedener Benutzer. Er erlaubt die Abfrage und das Löschen aller Nachrichten in dem Mailboxen. Die Kommunikation mit den Client-Programmen erfolgt dabei über eine TCP/IP-Socket-Verbindung. Der EMail-Server empfängt die Benutzerinformation und Befehle über diese Verbindung und verschickt darüber auch die angeforderten Daten. Ein Anforderungsbefehl besteht aus drei Zeilen mit folgendem Format:

```
USER <Username>
PASS <Paßwort>
RETR
```

Ein Löschbefehl hat folgendes Format

```
USER <Username>
PASS <Paßwort>
DELE
```

Die Antwort des Servers besteht mindestens aus einer Zeile mit einer Statusmeldung. Bei erfolgreicher Anfrage (d.h. wenn der Benutzer sich korrekt identifiziert hat), folgt bei der Anforderung von Mail der Inhalt der Mailbox, wobei hier einzelne Mails nicht getrennt werden. Beim Löschen von Mail wird außer der Statusmeldung nichts zurückgegeben.

Erfolgsfall bei Mail-Anforderung:

```
OK
<Mailboxinhalt>
```

Erfolgsfall beim Löschen von Mail:

```
OK
```

Benutzer nicht korrekt identifiziert:

```
Access Denied
```

Die Mailboxen sind einfache Dateien, welche jeweils den Namen des entsprechenden Benutzers tragen und immer existieren (d.h. es wird beim Löschen der Mail nur der Inhalt der Datei gelöscht und nicht die Datei selbst). Die Benutzerdaten (Name und Paßwort) sind in einer Paßwortdatei mit folgendem Aufbau abgelegt:

```
<Name> <Paßwort>
```

Beispiel:

```
Mueller computer123
```