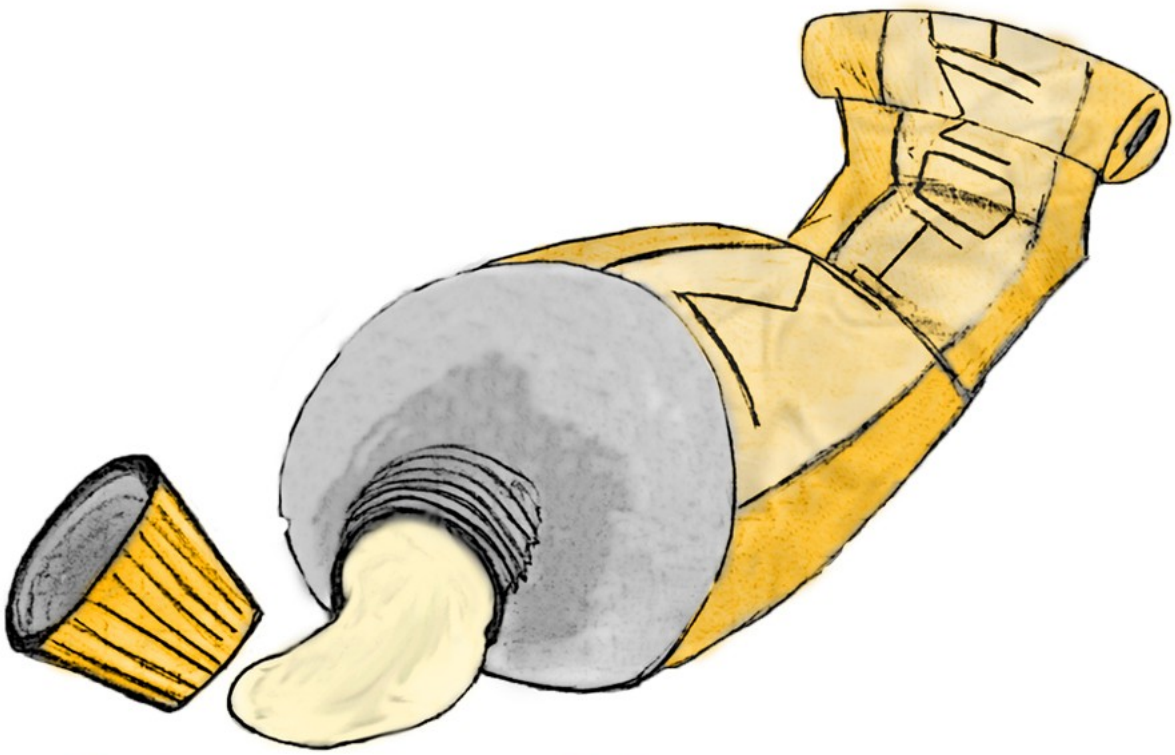
Entwurfsdokument

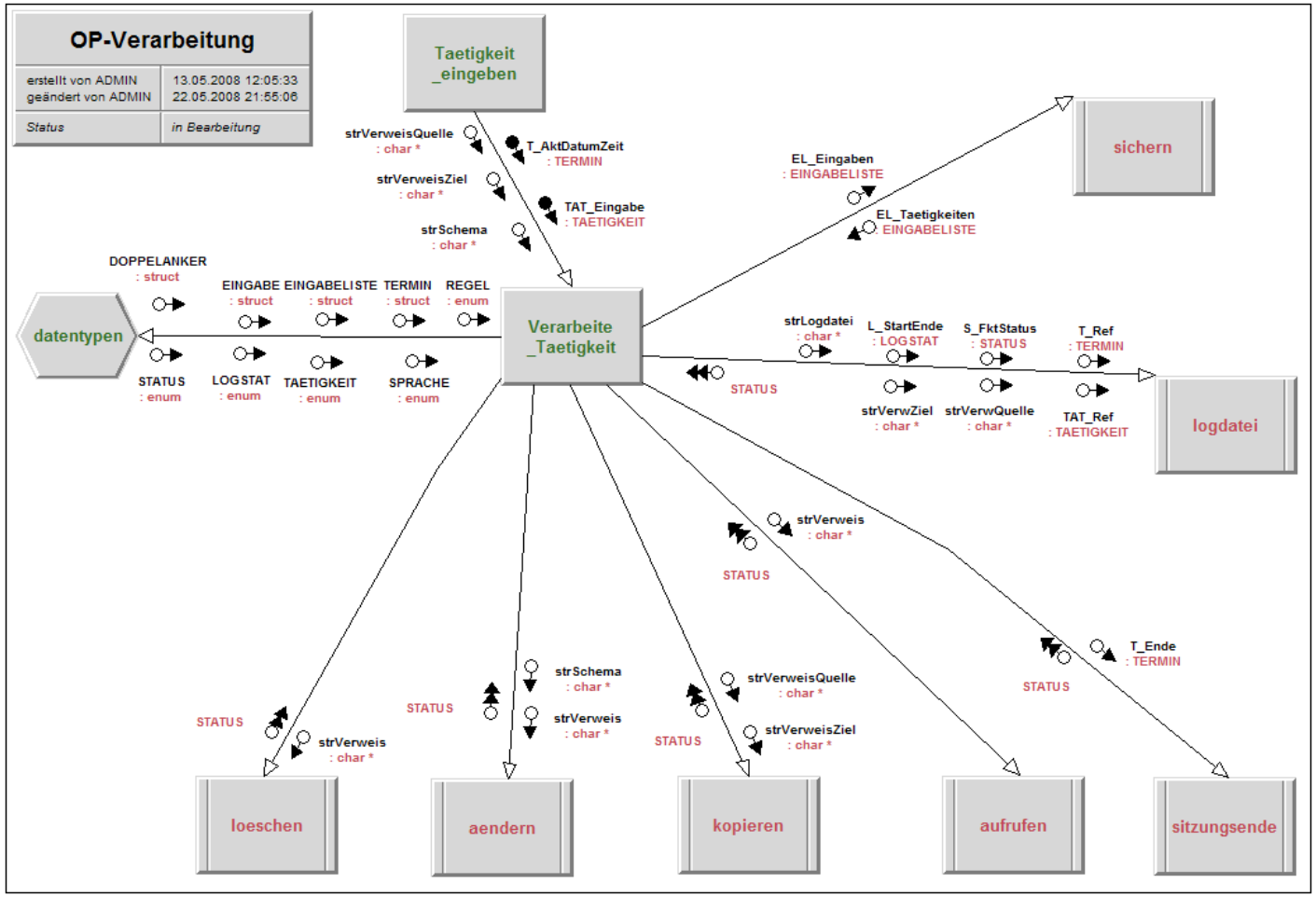
„MathoMadaja“



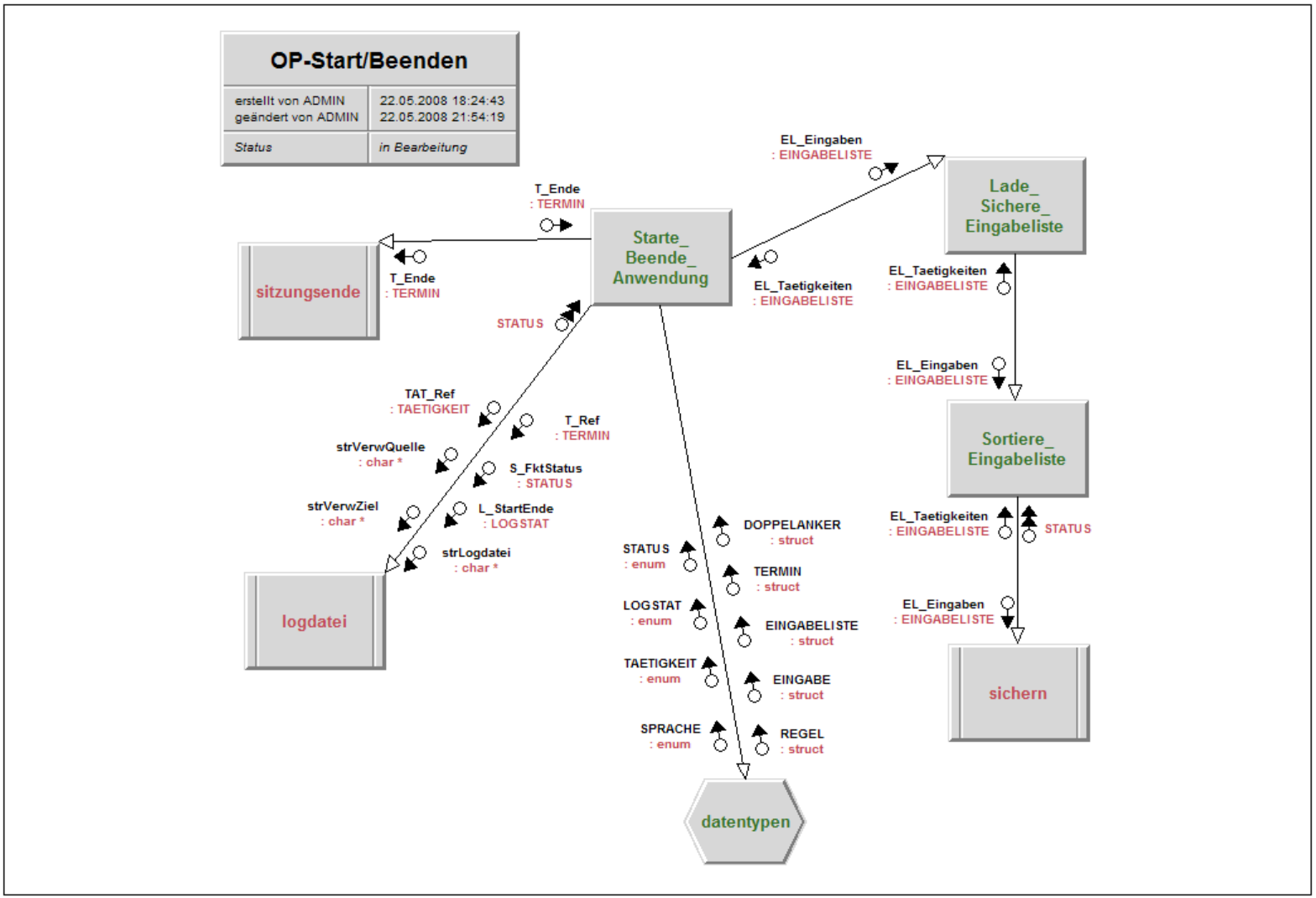
Inhaltsverzeichnis

1 Operationsdiagramme

1.1 Verarbeitung



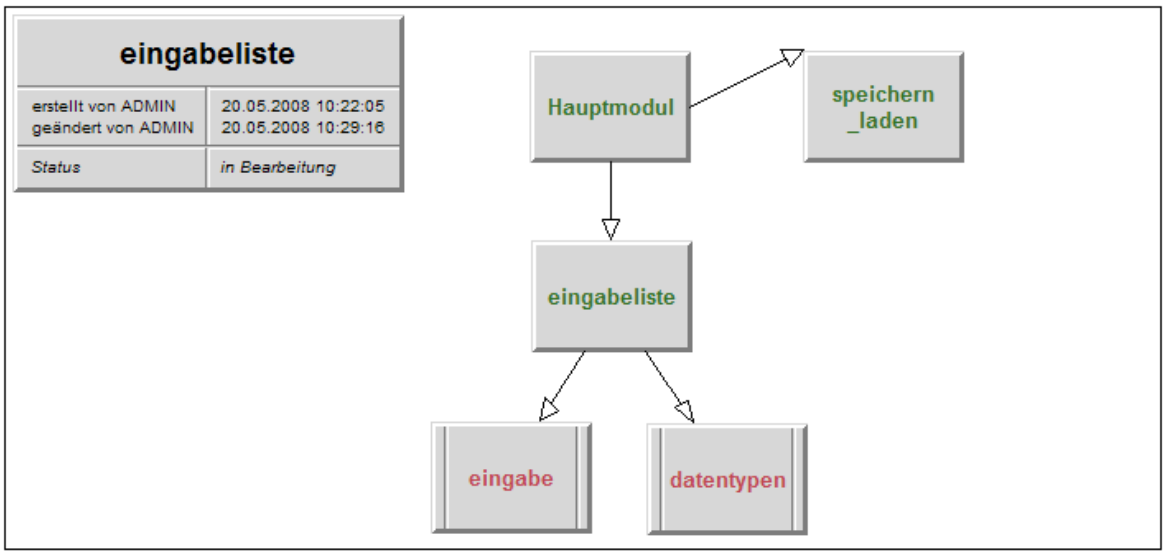
1.2 Start / Beenden



2 Module

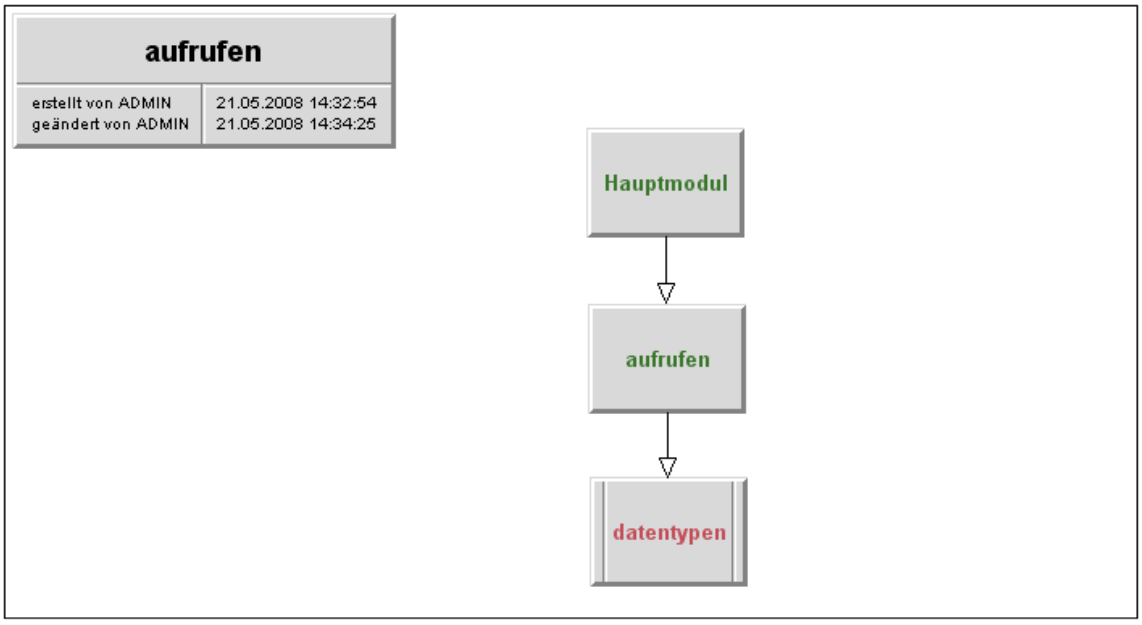
2.1 Modul „eingabeliste"

In diesem Modul werden die zu verarbeitenden Daten vom Benutzer eingegeben und dann als Zeichenkette (strVerweisQuelle und strtVerweisZiel oder strSchema) ans Hauptmodul weitergegeben.



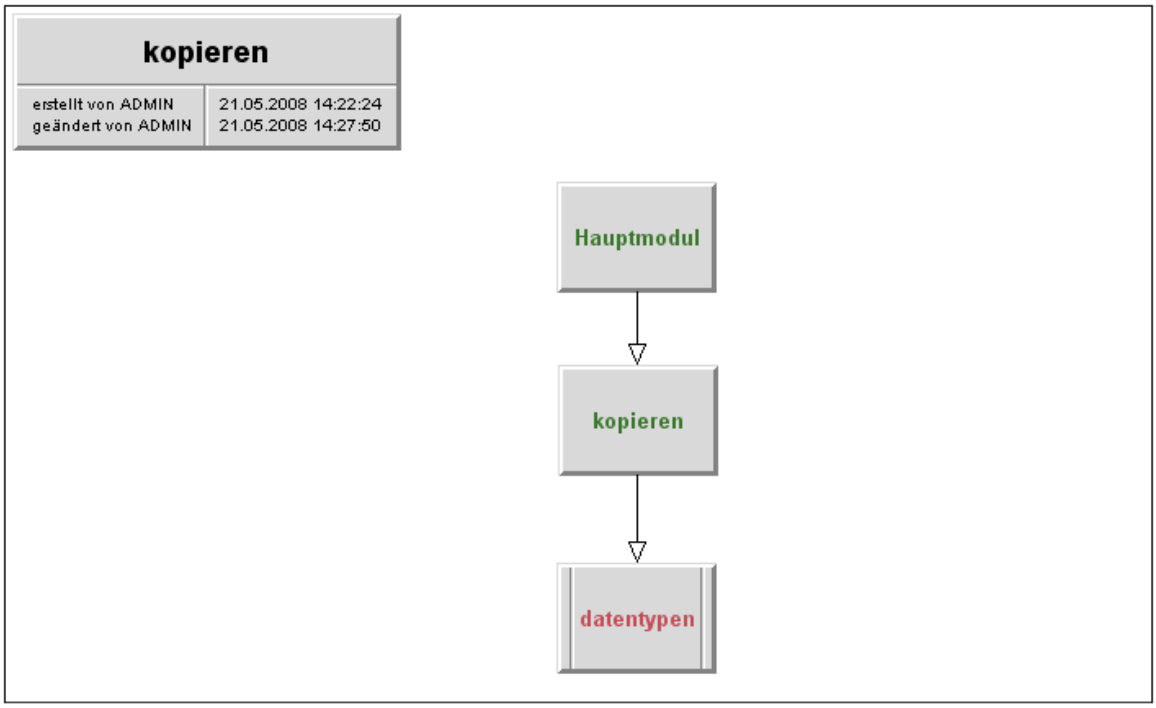
2.2 Modul „aufrufen"

Durch dieses Modul wird die vom Hauptmodul übergebene Zeichenkette strVerweis (die einen Pfad enthält) verarbeitet. Dadurch wird der Pfad (z.B. ausführbare Datei) der Zeichenkette aufgerufen.



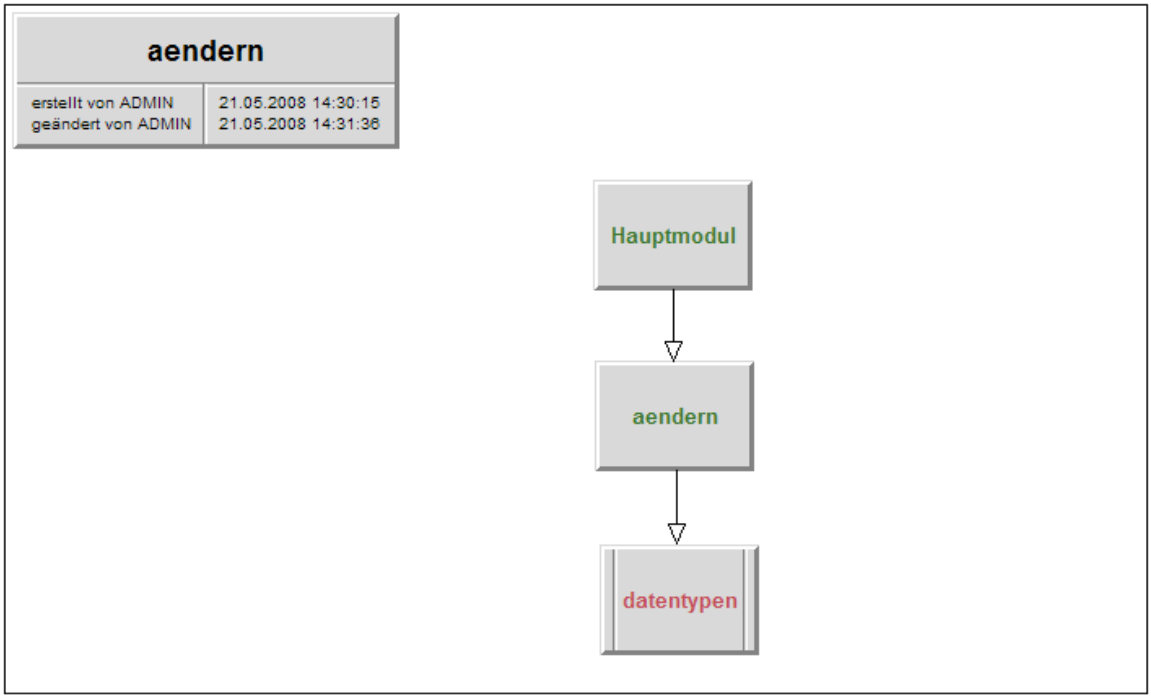
2.3 Modul „kopieren"

Das Hauptmodul übergibt zwei Zeichenketten an das Modul „kopieren". Dabei enthält strVerweisQuelle den Ursprungspfad (Information über Dateien / Ordner die kopiert werden sollen), und strVerweisZiel den Zielpfad (Information über den Ordner in den kopiert werden soll).



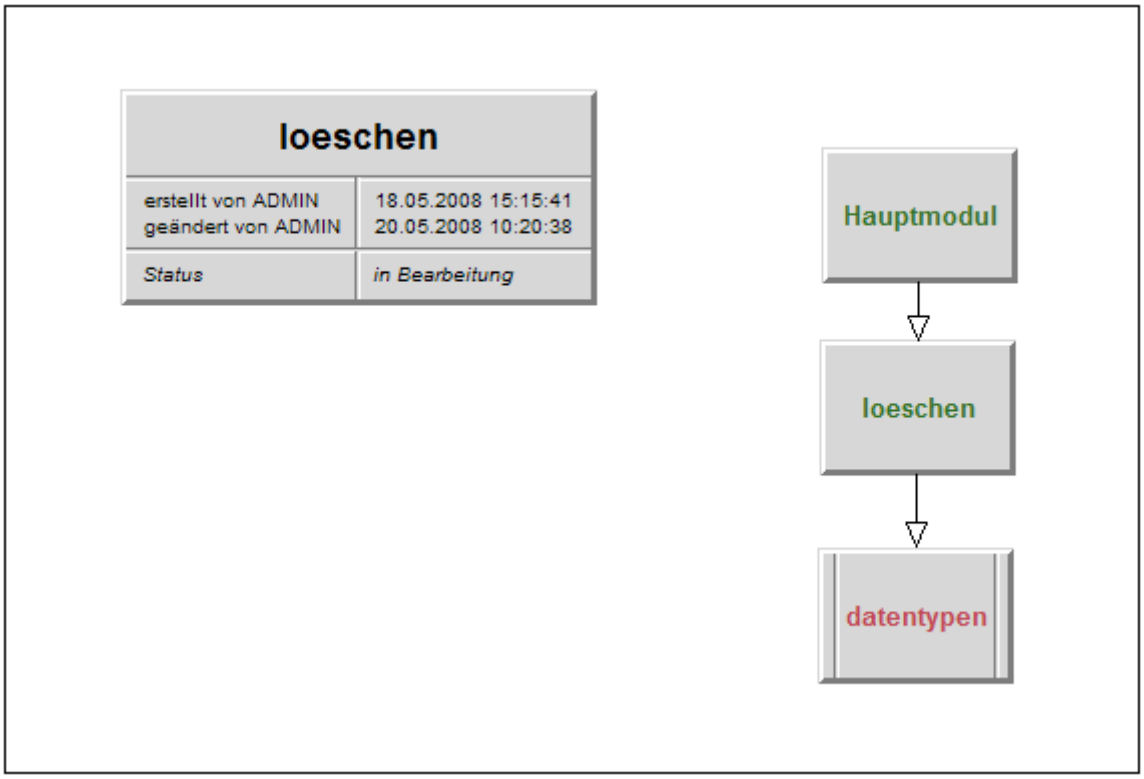
2.4 Modul „aendern"

Das Hauptmodul übergibt zwei Zeichenketten (strSchema und strVerweis) an das Modul „aendern". Diese enthalten Informationen über die ursprüngliche Bezeichnung der Datei bzw. des Ordners sowie über die über neue Bezeichnung.



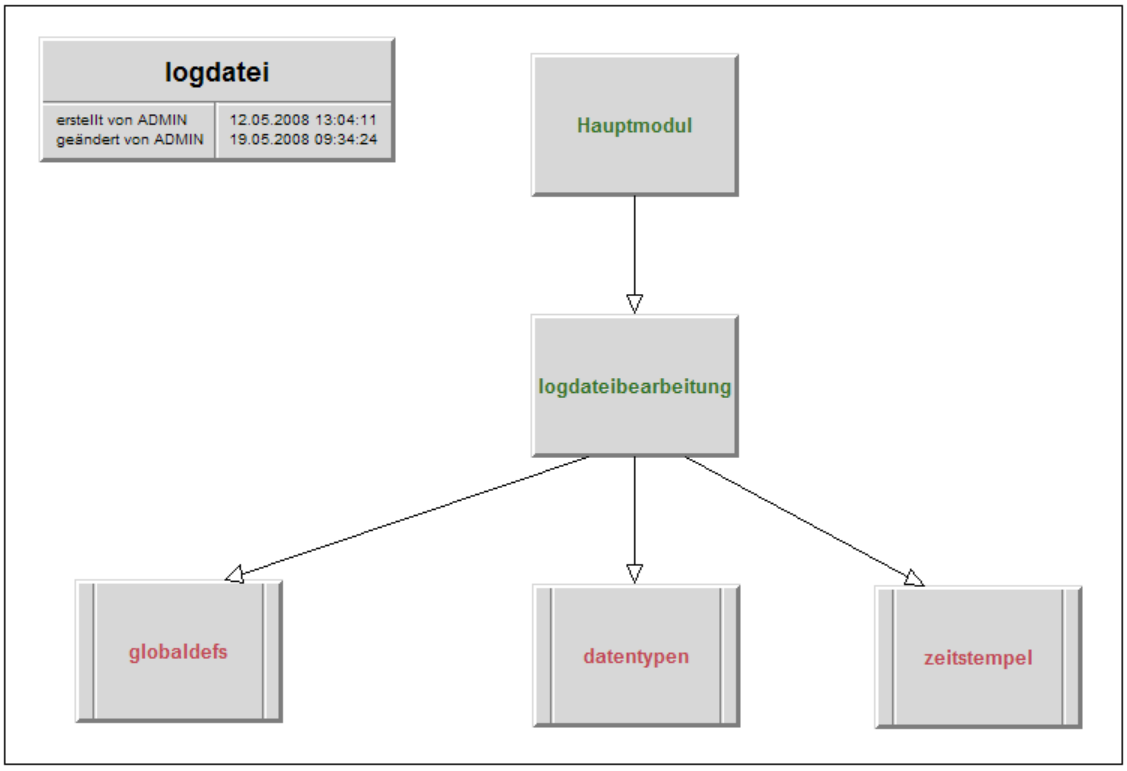
2.5 Modul „loeschen"

Dieses Modul ist verantwortlich für das Löschen von Dateien bzw. Verzeichnissen. Die vom Hauptmodul übergebene Zeichenkette strVerweis enthält den Pfad der zu löschenden Datei bzw. des zu löschenden Ordners.



2.6 Modul „logdatei"

Dieses Modul ist verantwortlich für die Erstellung der Log-Datei. Hierfür übergibt das Hauptmodul die Zeichenkette strLogdatei (Name der Datei), den Datentyp LOGSTAT (Programmstart/-ende, Modulstart/-ende), den Datentyp S_FktStatus (Status des Untermoduls) sowie die Zeichenkette E_Taetigkeit (eingegebene Tätigkeit).

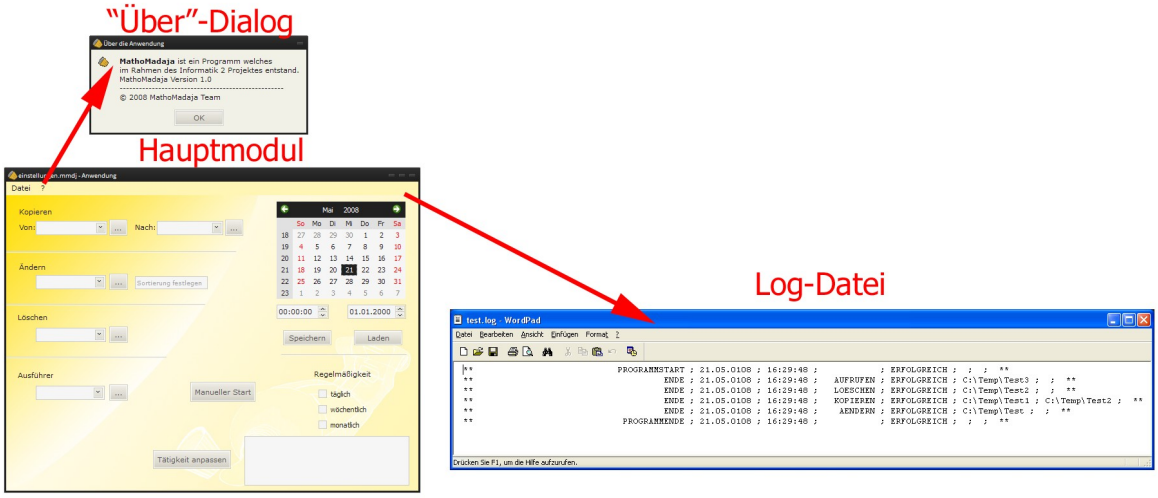


2.7 Zuordnung der Module

- Hauptmodul
  - Marco Bezzon, David Garus
- Module „eingabeliste, logdatei"
  - Marco Bezzon
- Modul „aufrufen"
  - Jan Ripper
- Modul „kopieren"
  - Thomas Harr
- Modul „aendern"
  - Markus Schüler
- Modul „loeschen"
  - David Garus

3 Beschreibung der Daten

3.1 Bildschirmausgabe



3.2 Datei „datentypen.h"

Diese Datei enthält globale Definitionen sowohl für das Hauptmodul als auch für die einzelnen Untermodule.

Taetigkeit, welche es Auszuführen gilt:

```
typedef enum { TFEHLER, TKEINE ,AUFRUFEN, LOESCHEN, AENDERN,
KOPIEREN } TAETIGKEIT;
```

Regelmäßigkeit: Wie oft soll die Tätigkeit ausgeführt werden

```
typedef enum { KEINE, TAEGLICH, WOECHENTLICH, MONATLICH } REGEL;
```

Log-Datei Status:
Wird der Funktion Log-Datei übergeben, um entsprechendes auszugeben

```
typedef enum { LFEHLER, START, ENDE, PROGSTART, PROGENDE } LOGSTAT;
```

Rückgabestatus der "Haupt"-Funktionen, ob erfolgreich abgearbeitet wurde:

```
typedef enum { SFEHLER, ERFOLG } STATUS;
```

Sprache-Auswahl, momentan noch nicht weiter berücksichtigt:

```
typedef enum { DEUTSCH } SPRACHE;
```

Termin-Datentyp enthält Datum, Zeitpunkt und Regelmäßigkeit wann was ausgeführt werden soll:

```
typedef struct {
                int iW_Tag;
                REGEL R_Regelmaessigkeit;
                DATUM D_Datum;
                ZEIT  Z_Zeit;
                } TERMIN;

typedef struct {
                int iTag;
                int iMonat;
                int iJahr;
                } DATUM;


typedef struct {
                int iSekunde;
                int iMinute;
                int iStunde;
                } ZEIT;
```

Dieser Datentyp enthält die eigentlichen Eingaben eines Benutzers und kann so gespeichert werden.

- T_Termin: Enthält Datum und Uhrzeit, wann eine Tätigkeit ausgeführt werden soll.
- strQuelle: String, der enthalten sein MUSS, gültig für alle Tätigkeiten.
- strZiel: String, der enthalten sein KANN, nur beim Kopieren notwendig.
- strSchema: String, der enthalten sein KANN, nur beim Ändern notwendig
- TAT_Funktion: Enthält die Tätigkeit, die ausgeführt werden soll.

```
typedef struct Eingabe {
                TERMIN T_Termin;
                long long int lli_SortMuster;
                char strQuelle[ANZ_CHAR + 1];
                char strZiel[ANZ_CHAR + 1];
                char strSchema[ANZ_SCHEMA + 1];
                TAETIGKEIT TAT_Funktion;
                } EINGABE;
```

Dieser Datentyp enthält eine Liste, bestehend aus mehreren EINGABEN:

```
typedef struct EingabeListe {
                EINGABE *E_Taetigkeit;
                struct EingabeListe *EL_Naechste;
                struct EingabeListe *EL_Vorherige;
                } EINGABELISTE;
```

Der Datentyp Doppelanker besteht aus einem Anfangsanker der Eingabeliste auf ihr erstes Element und einen Schlussanker auf das letzte Element der Liste. Der Doppelanker hat den Vorteil, dass durch ihn ein sortieren bzw. durchsuchen der Eingabeliste vereinfacht wird.

```
typedef struct doppelAnker {
        EINGABELISTE * EL_Erst;  /* Erster in Liste */
        EINGABELISTE * EL_Letzt; /* Letzter in Liste*/
}DOPPELANKER;
```