Softwarepraktikum 1998

Projektdokumentation

(Teilbeleg Nr. 1)

Aufgabe: Entwicklung eines Tools zum grafischen Entwurf von VHDL-Beschreibungen elektronischer Schaltungen

Bearbeiter: Jan Horbach

Praktikumsbetreuer: Dr. Martin Würkert

Chemnitz, den 18.03.2009

1. Spezifikation der funktionellen Anforderungen

1.1. Produktbeschreibung

Die Entwicklungsumgebung VisualVHDL unterstützt den VHDL-Programmierer (besonders den Studenten) bei der Generierung von VHDL-Quellcode aus einer grafischen Strukturbeschreibung. Dazu bietet das System einen grafischen Editor, in dem die Komponenten zu Strukturarchitekturen zusammengesetzt werden. Im Editor werden die Komponenten und Signalleitungen interaktiv miteinander verbunden. Dabei kann der Entwickler auf schon vorhandene Entwürfe zurückgreifen, die alle in einer multiuserfähigen Datenbank abgelegt sind, die sich auf einem Server befindet. Von ihr werden Entities, Architectures und Packages verwaltet. Um die Sicherheit der Daten zu gewährleisten, werden allen Datenobjekten spezielle Zugriffsrechte auf Nutzer- und Nutzergruppenebene zugeteilt.

Nutzungsumgebung: Das System kann auf jedem Rechner mit einer Java 1.1 kompatiblen Java Virtual Machine und Swing 1.0 betrieben werden. Netzanbindung ist empfohlen, der Server kann aber auch lokal betrieben werden.

Nutzergruppen: Die Nutzer von VisualVHDL sind hauptsächlich in zwei Gruppen aufgeteilt: Die Administratoren, die Rechte festlegen können, und die „normalen“ Nutzer, die dann noch weiter aufgeteilt werden können.

1.2. Funktionelle Anforderungen

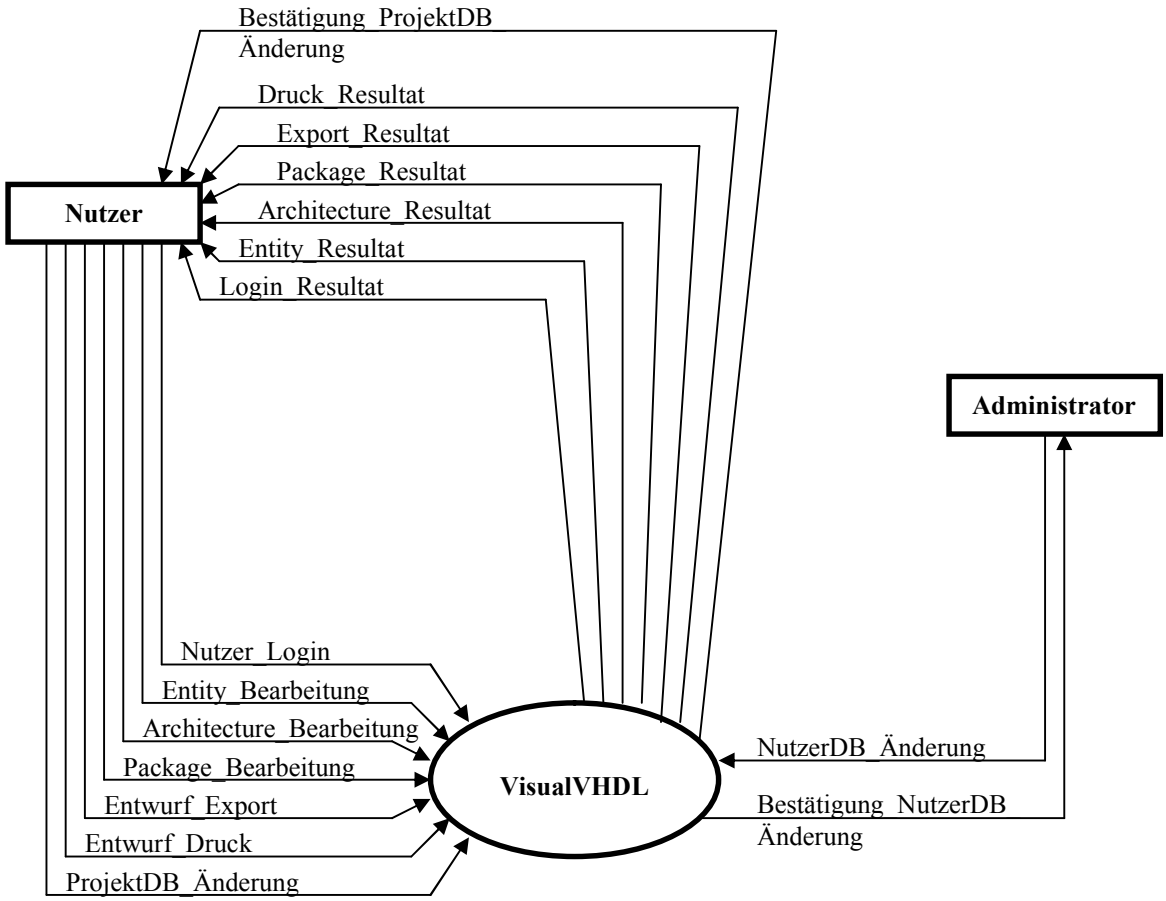
1.2.1. Umgebungsmodell

Ereignistabelle:

Legende: N: Nutzer
A: Administrator

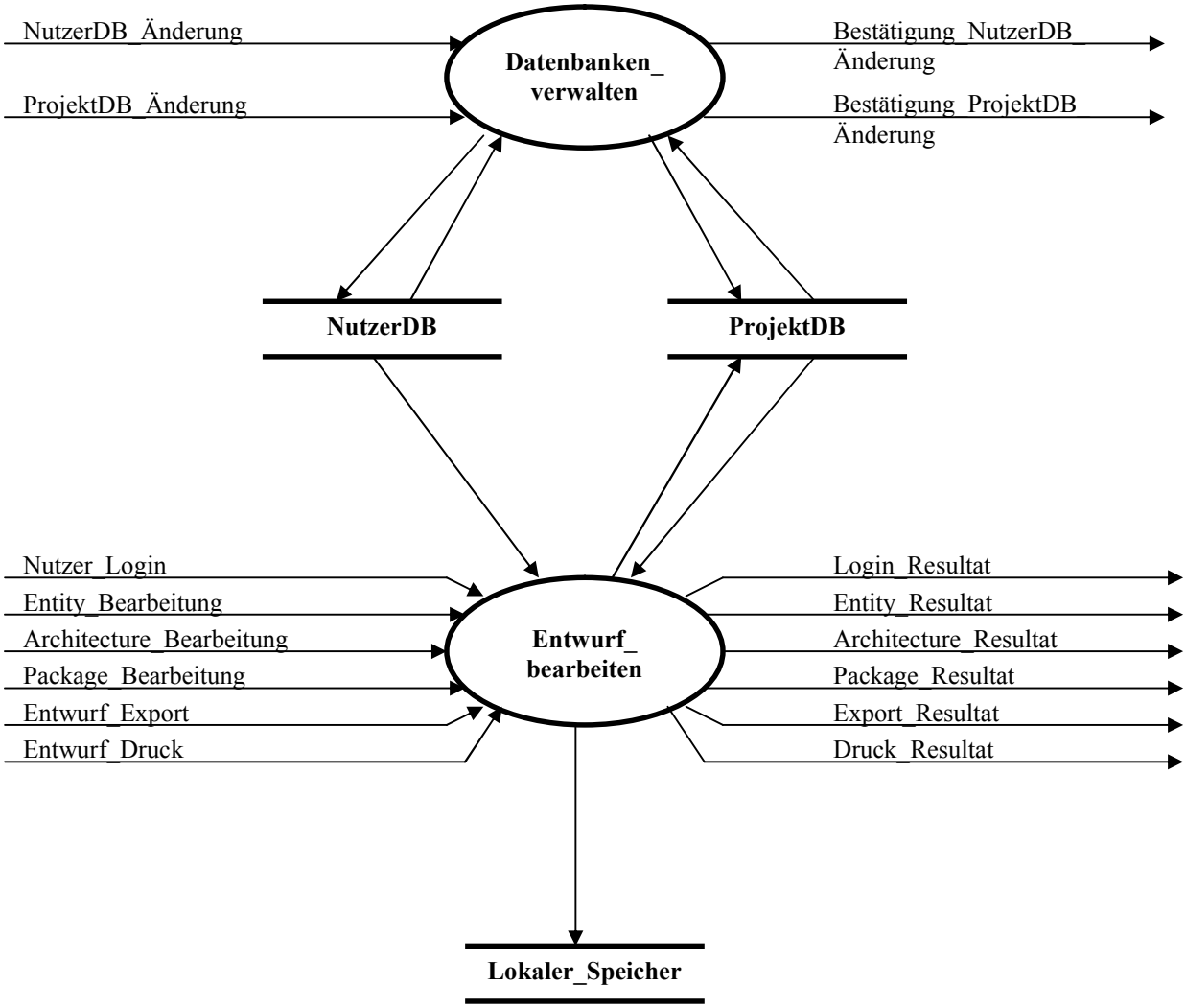
Nummer	Ereignis	Datenfluß zum System	Antwort des Systems
1.	N: Der Nutzer will sich einloggen.	Nutzer_Login	Login_Resultat; Nutzer
2.	N: Der Nutzer möchte eine Entity bearbeiten.	Entity_Bearbeitung	Entity_Resultat; Entity
3.	N: Der Nutzer möchte eine Architecture bearbeiten.	Architecture_Bearbeitung	Architecture_Resultat; Architecture
4.	N: Der Nutzer möchte eine Package bearbeiten.	Package_Bearbeitung	Package_Resultat; Package
5.	N: Der Nutzer will seinen Entwurf exportieren.	Entwurf_Export	Export_Resultat
6.	N: Der Nutzer will seinen Entwurf drucken.	Entwurf_Druck	Druck_Resultat
7.	A: Der Administrator will die Nutzerdatenbank verwalten.	NutzerDB_Änderung	Bestätigung_NutzerDB_Änderung
8.	N: Der Nutzer will die Projektdatenbank verwalten.	ProjektDB_Änderung	Bestätigung_ProjektDB_Änderung

Kontextdiagramm:



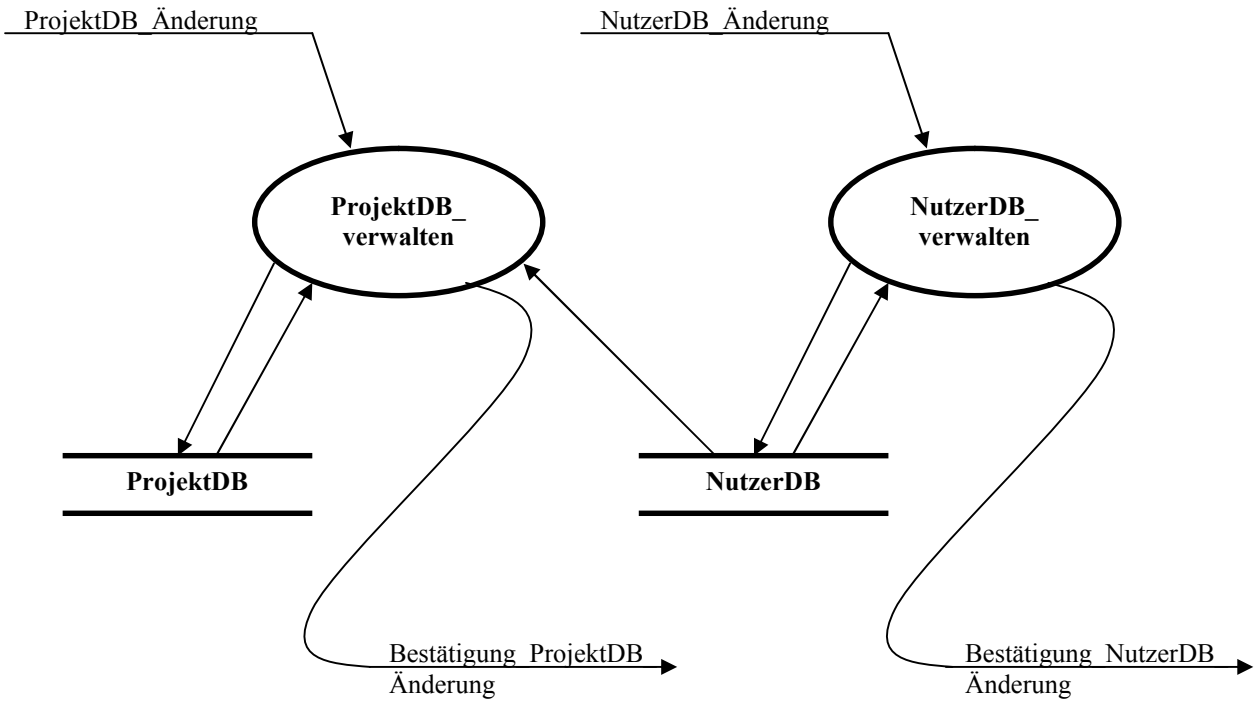
1.2.2. Verhaltensmodell

1.2.2.1. Grobes Verhaltensmodell (vergröbertes primäres Verhaltensmodell)

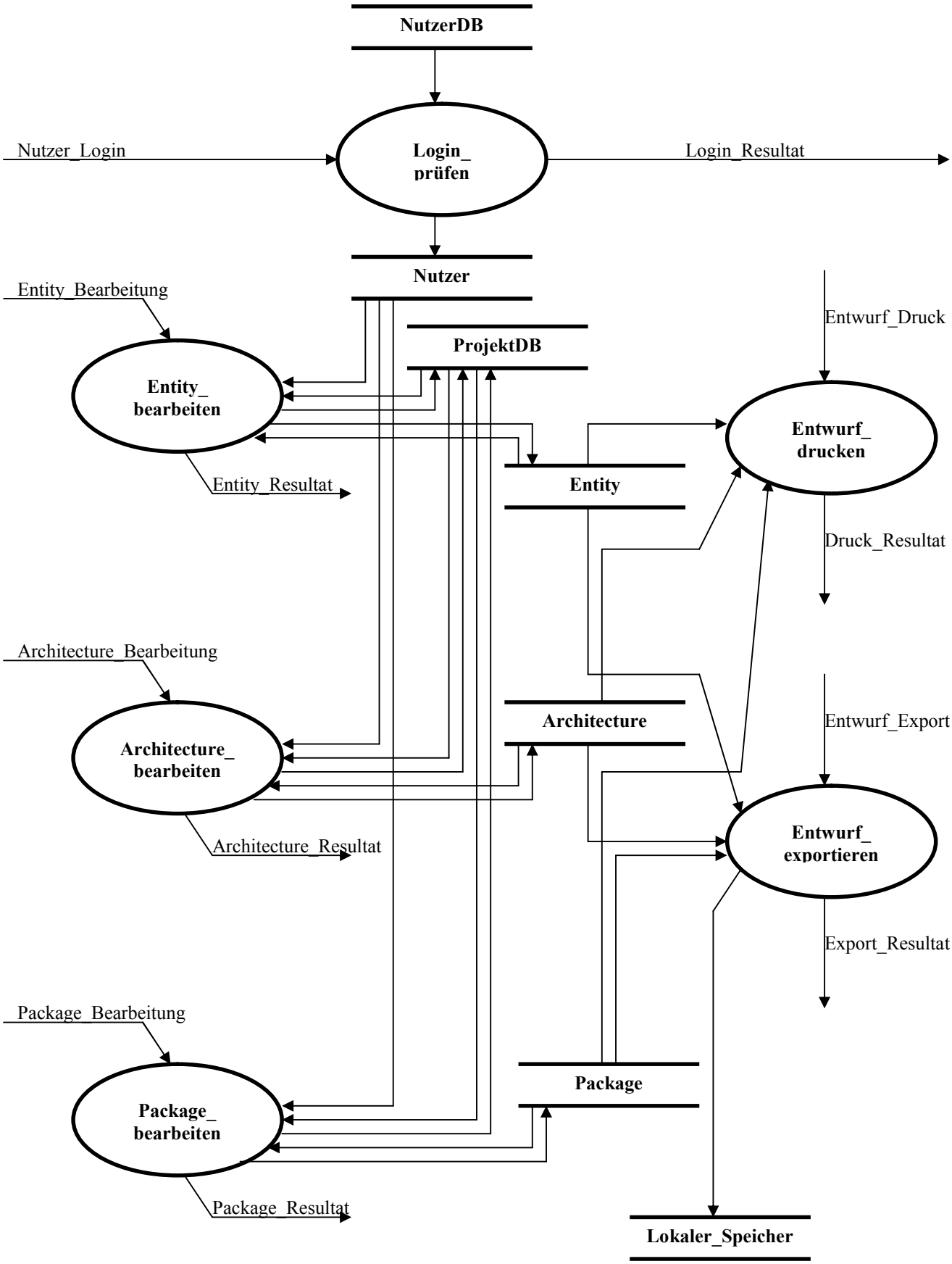


1.2.2.2. Primäres Verhaltensmodell

Teilmodell Datenbanken_verwalten:



Teilmodell Entwurf_bearbeiten:



1.2.2.3. Datenkatalog

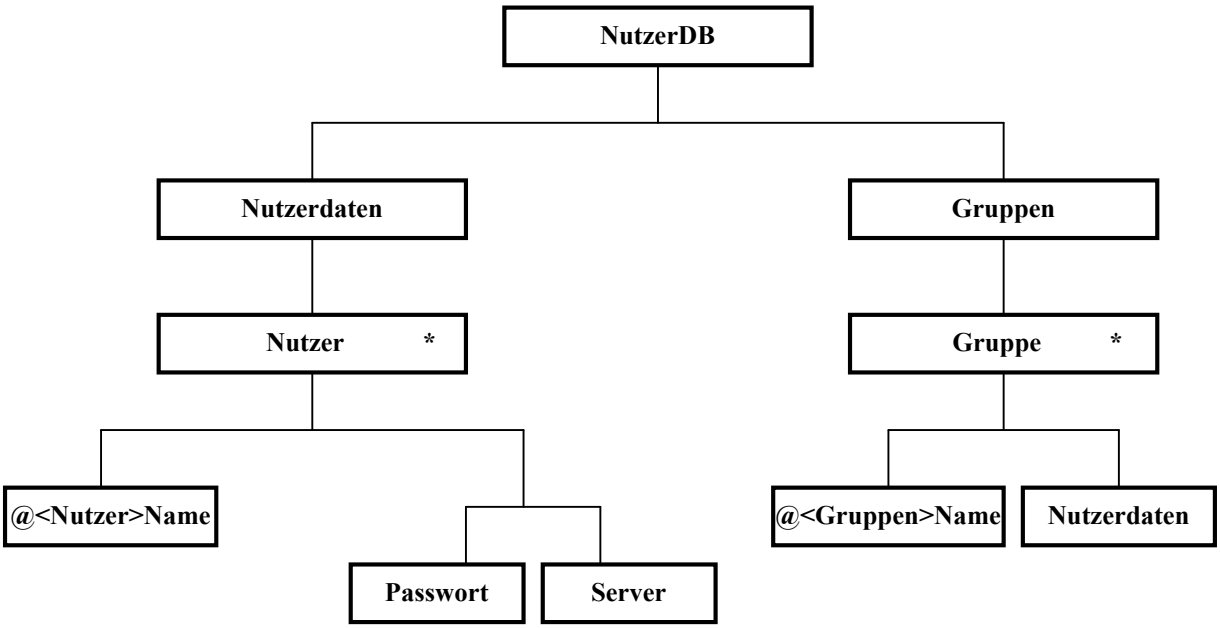
Element	Strukturbeschreibung
NutzerDB	**Nutzerdaten + Gruppen**
Nutzerdaten	{Nutzer}
Nutzer	@<Nutzer>Name + Passwort + Server
<Nutzer>Name	Zeichenkette10
Passwort	Zeichenkette20
Server	Zeichenkette100
Gruppen	{Gruppe}
Gruppe	@<Gruppen>Name + Nutzerdaten
<Gruppen>Name	Zeichenkette10
ProjektDB	**Verzeichnisse**
Verzeichnisse	{Verzeichnis}
Verzeichnis	@<Verzeichnis>Name + Verzeichnisse + Entities + Packages + ACL
<Verzeichnis>Name	Zeichenkette50
ACL	{Nutzerrechte} + {Gruppenrechte} * insgesamt mindestens einer *
Nutzerrechte	Nutzer + Rechte
Gruppenrechte	Gruppe + Rechte
Rechte	(“read”) + (“write”) + (“changeACL”) * “read” schließt Verzeichniswechsel ein * * “write” schließt Löschen mit ein *
Entities	{Entity}
Entity	@<Entity>Name + Ports + Generics + Architectures + Packages
<Entity>Name	Zeichenkette50
Ports	{Port}
Port	@<Port>Name + Modus + <Port>Datentyp + <Port>Wert
<Port>Name	Zeichenkette50
Modus	[“in” / “out” / “inout” / “buffer” / “register”] * legt Art des Signalflusses fest *
<Port>Wert	Zeichenkette1000 * kann von beliebigem VHDL-Typ sein, z.B. “10 ns” *
Generics	{Generic}
Generic	@<Generic>Name + <Generic>Datentyp + <Generic>Wert
<Generic>Name	Zeichenkette50
<Generic>Wert	Zeichenkette1000 * kann von beliebigem VHDL-Typ sein, z.B. “10 ns” *
Architectures	{Architecture}
Architecture	@<Architecture>Name + <gehört_zu>Entity + [Verhalten / Struktur]
<Architecture>Name	Zeichenkette50
Verhalten	Zeichenkette * enthält einen VHDL-Quelltext, sollte 64K nicht überschreiten *
Struktur	Components + Leitungen
Components	{Component}
Component	@Entity + Position + Größe * eine grafische Komponente *
Position	<Position>x + <Position>y
<Position>x	gZahl * x-Position auf dem Bildschirm *
<Position>y	gZahl * y-Position auf dem Bildschirm *
Größe	<Größe>x + <Größe>y
<Größe>x	gZahl * x-Ausdehnung auf dem Bildschirm *
<Größe>y	gZahl * y-Ausdehnung auf dem Bildschirm *
Leitungen	{Leitung}
Leitung	<Leitung>von + <Leitung>nach
von	Entity + Port
nach	Entity + Port

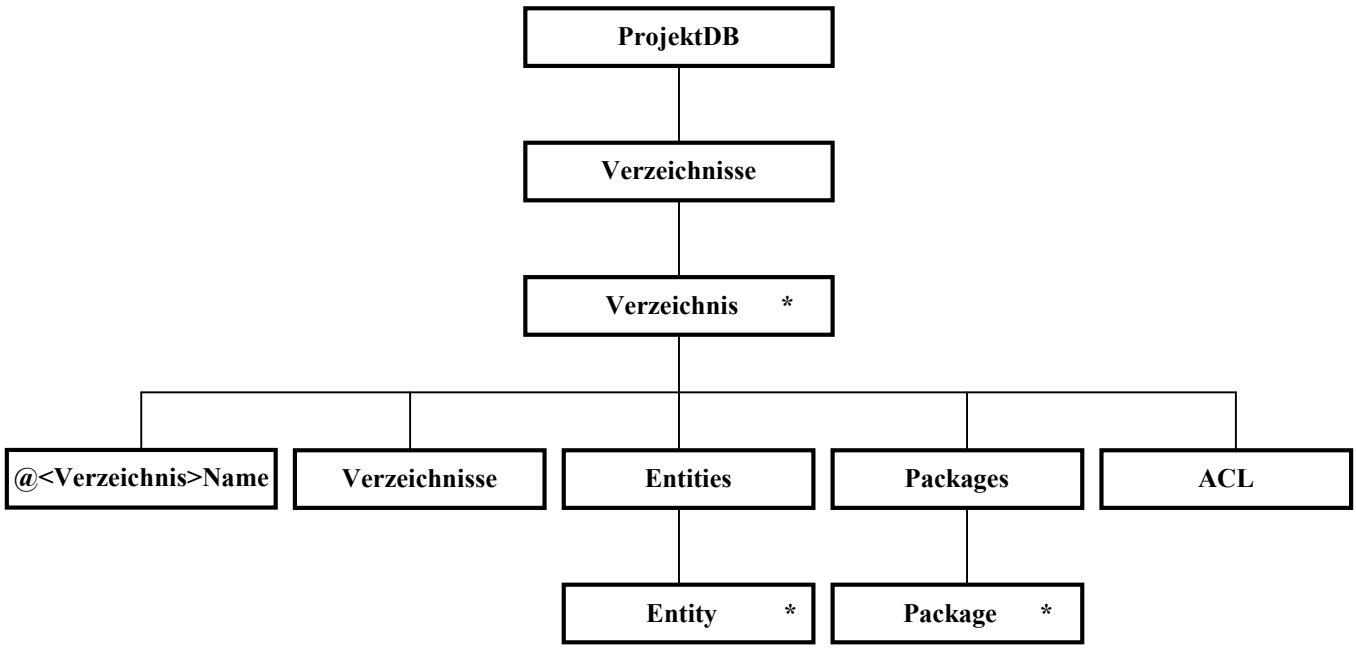
Element	Strukturbeschreibung
Packages	{Package}
Package	@<Package>Name + Datentypen + Konstanten
<Package>Name	Zeichenkette50
Datentypen	{Datentyp}
Datentyp	[Typ / Subtyp]
Typ	@<Typ>Name + <Typ>Definition * selbstdefinierte Datentypen *
<Typ>Name	Zeichenkette50
Definition	Zeichenkette100 * z.B. Aufzähltyp *
Subtyp	@<Subtyp>Name + <Ober>Datentyp + Ausdruck * abgeleitete Datentypen *
<Subtyp>Name	Zeichenkette50
Ausdruck	Zeichenkette100 * z.B. Bereichseingrenzung *
Konstanten	{Konstante}
Konstante	@<Konstanten>Name + Datentyp + <Konstanten>Wert
<Konstanten>Name	Zeichenkette50
Wert	Zeichenkette100 * kann von beliebigem VHDL-Typ sein, z.B. “10 ns” *

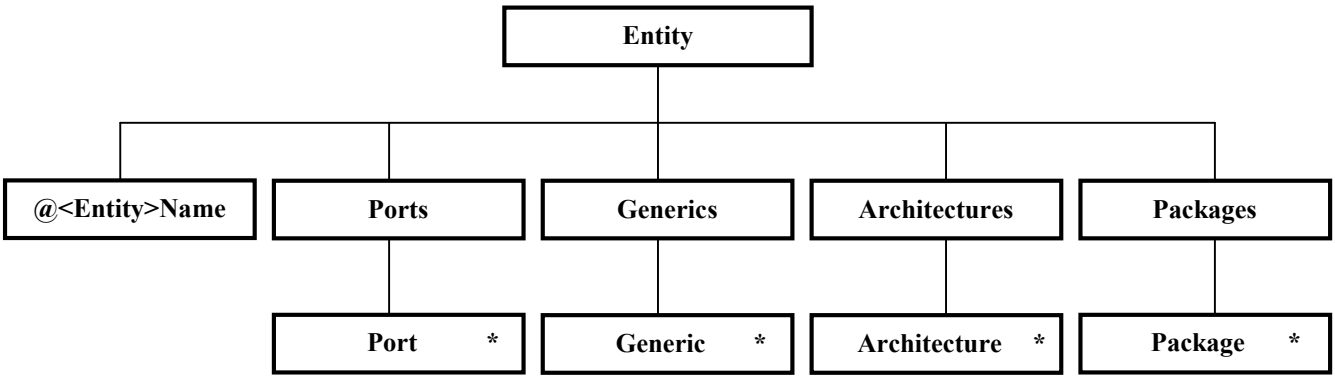
Element	Strukturbeschreibung
Dynamische Daten	
Nutzer_Login	Nutzer
Login_Resultat	[“successful” / “login incorrect” / “unknown server”]
Entity_Bearbeitung	[“new” / “load” + Entity / “save” + Entity / “edit” + Entity]
Entity_Resultat	[“successful” / “error”]
Architecture_Bearbeitung	[“new” / “load” + Architecture / “save” + Architecture / “edit” + Architecture]
Architecture_Resultat	[“successful” / “error”]
Package_Bearbeitung	[“new” / “load” + Package / “save” + Package / “edit” + Package]
Package_Resultat	[“successful” / “error”]
Entwurf_Export	“export” + Architecture
Export_Resultat	[“successful” / “error”]
Entwurf_Druck	“print” + Architecture
Druck_Resultat	[“successful” / “error”]
NutzerDB_Änderung	[“add” / “del” / “edit”] + [Nutzer / Gruppe]
Bestätigung_NutzerDB_ Änderung	[“successful” / “error”]
ProjektDB_Änderung	[[“create” / “del” / “edit”] + Verzeichnis / “del” + [Verzeichnis / Entity / Architecture / Package]]
Bestätigung_ProjektDB_ Änderung	[“successful” / “error”]

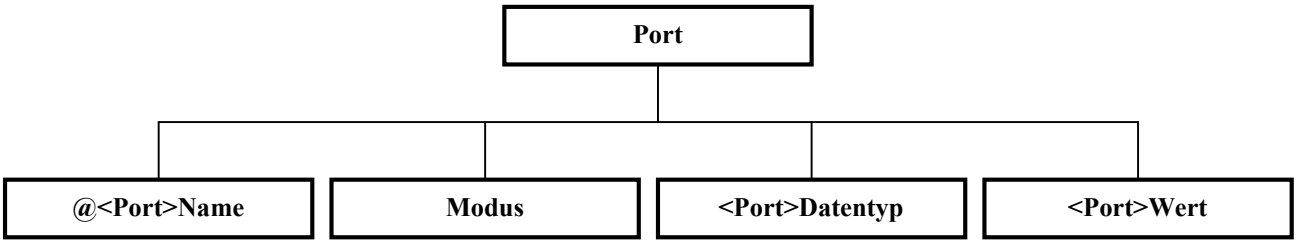
Anmerkung: Die Rückgabewerte “error” werden bei eventuellen Fehlern gesetzt, auch wenn diese in der Prozeßspezifikation zum Teil vernachlässigt werden.

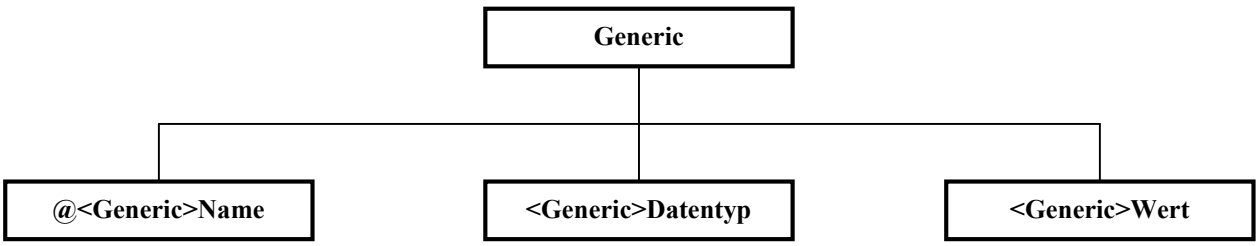
Struktur der Speicher (grafische Ergänzung zu Inhalten des Datenkatalogs):

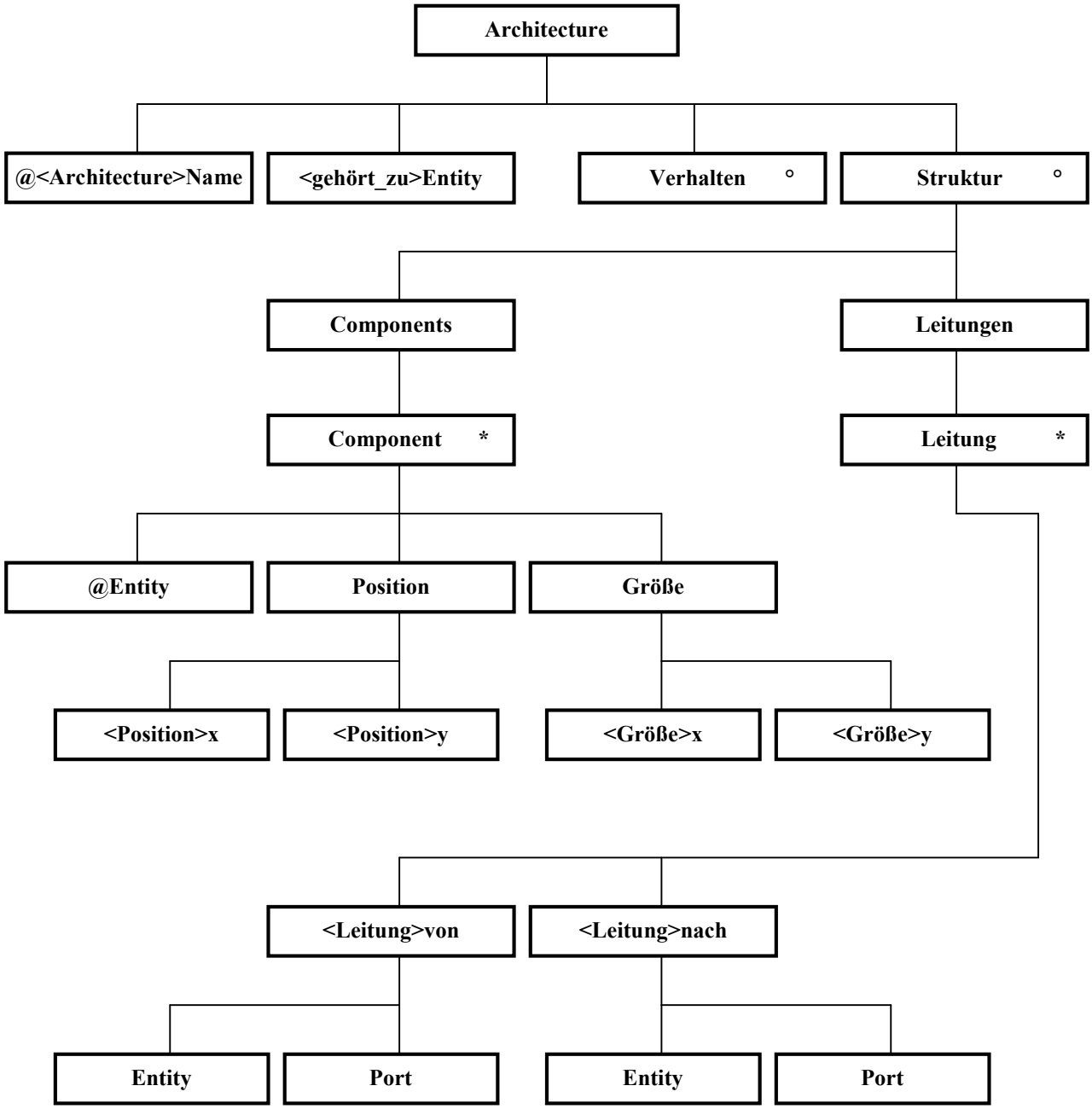


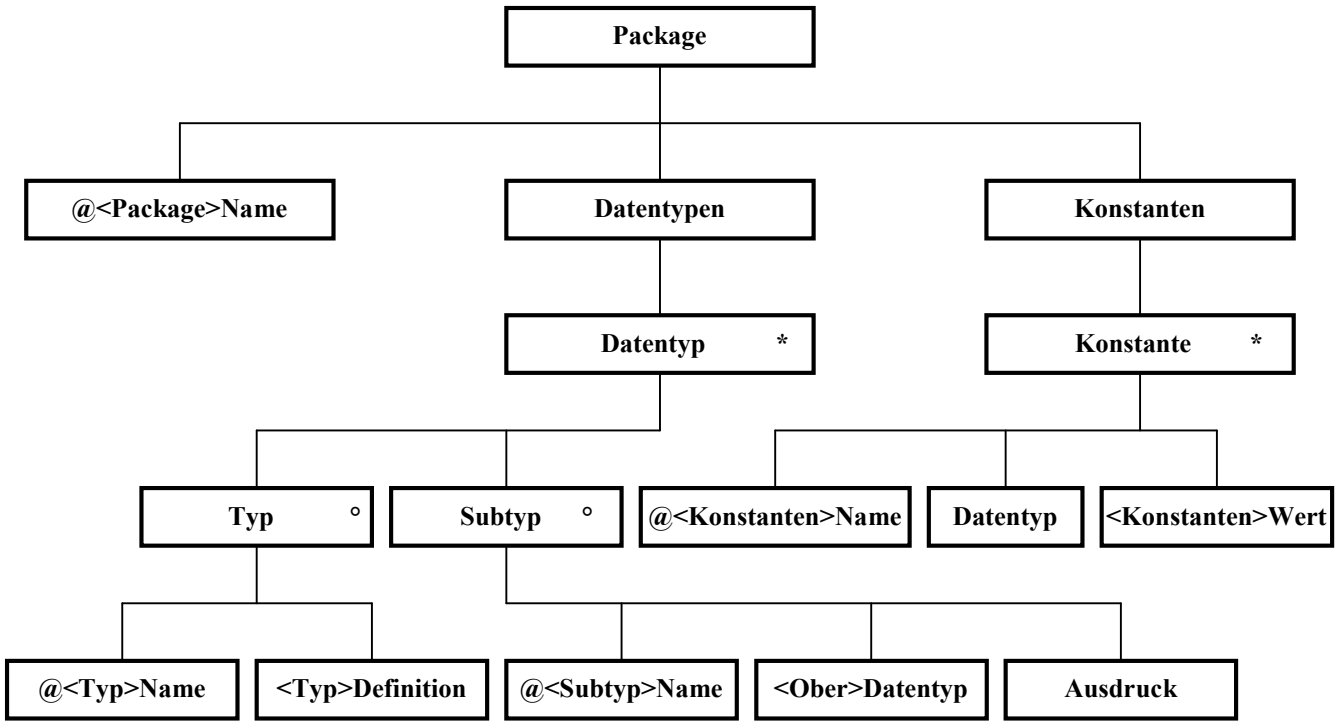



Entity
@<Entity>Name
Ports
Generics
Architectures
Packages
Port *
Generic *
Architecture *
Package *

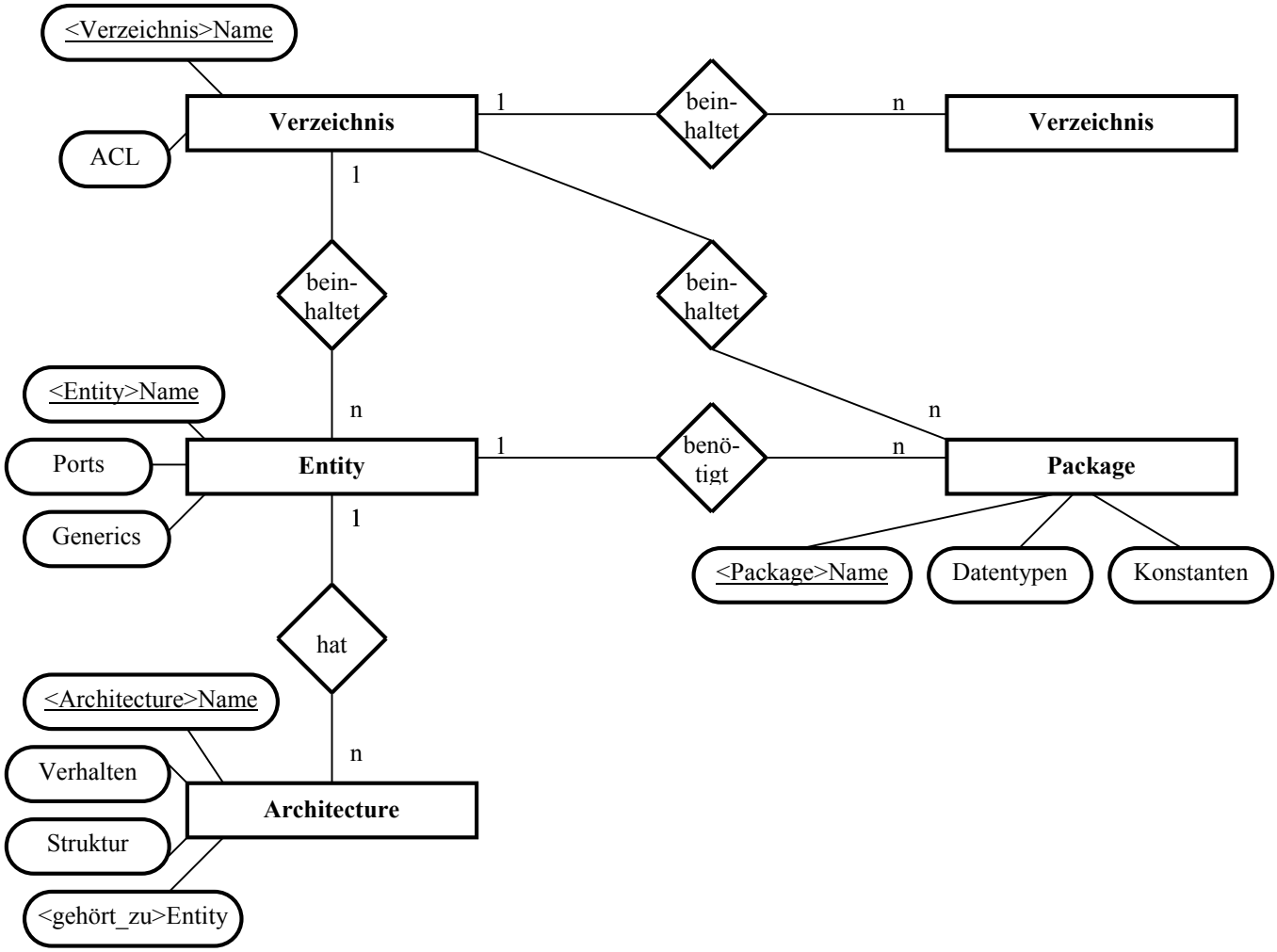

Port
@<Port>Name
Modus
<Port>Datentyp
<Port>Wert


Generic
@<Generic>Name
<Generic>Datentyp
<Generic>Wert


Architecture
@<Architecture>Name
<gehört_zu>Entity
Verhalten °
Struktur °
Components
Leitungen
Component *
Leitung *
@Entity
Position
Größe
<Position>x
<Position>y
<Größe>x
<Größe>y
<Leitung>von
<Leitung>nach
Entity
Port
Entity
Port


Package
@<Package>Name
Datentypen
Konstanten
Datentyp *
Konstante *
Typ °
Subtyp °
@<Konstanten>Name
Datentyp
<Konstanten>Wert
@<Typ>Name
<Typ>Definition
@<Subtyp>Name
<Ober>Datentyp
Ausdruck

Beziehungen zwischen Speichern (ERD):



Prozeßspezifikationen:

Prozeß: **ProjektDB_verwalten** Datum: 01.05.98
Bearbeiter: Jan Horbach

```
Voraussetzung: ProjektDB ist vorhanden, Nutzer/Gruppe existiert,
               es treten keine Fehler während des Prozesses auf

begin
  Bestätigung_ProjektDB_Änderung := "successful";
  case ProjektDB_Änderung of
    "create" + Verzeichnis: erzeuge Verzeichnis;
                            if Verzeichnis.ACL enthält "write" then
                              schreibe Verzeichnis in ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
    "del"    + Verzeichnis: if Verzeichnis.ACL enthält "write" then
                              lösche Verzeichnis aus ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
    "del"    + Entity:      suche Verzeichnis in ProjektDB mit Entity
                            if Verzeichnis.ACL enthält "write" then
                              lösche Entity aus ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
    "del"    + Architecture:suche Verzeichnis in ProjektDB mit Architecture
                            if Verzeichnis.ACL enthält "write" then
                              lösche Architecture aus ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
    "del"    + Package:     suche Verzeichnis in ProjektDB mit Package
                            if Verzeichnis.ACL enthält "write" then
                              lösche Package aus ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
    "edit"   + Verzeichnis: if Verzeichnis.ACL enthält "read" then
                              lies Verzeichnis aus ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
                            if Verzeichnis.ACL enthält "write" then
                              ändere Verzeichnis;  * Name usw. *
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
                            if Verzeichnis.ACL enthält "changeACL" then
                              ändere Verzeichnis.ACL;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
                            if Verzeichnis.ACL enthält "write" then
                              schreibe Verzeichnis in ProjektDB;
                            else
                              Bestätigung_ProjektDB_Änderung := "error";
                            end if;
  end case;
end
```

Prozeß: **NutzerDB_verwalten** Datum: 01.05.98
Bearbeiter: Jan Horbach

Voraussetzung: NutzerDB ist vorhanden, nur der Administrator kann diese Datenbank ändern, Administrator ist bereits in der NutzerDB vorhanden

```
begin
  case NutzerDB_Änderung of
    "add"  + Nutzer: erzeuge Nutzer;
                     schreibe Nutzer in NutzerDB;
    "del"  + Nutzer: lösche Nutzer aus NutzerDB;
    "edit" + Nutzer: lies Nutzer aus NutzerDB;
                     ändere Nutzer;  * Name, Passwort usw. *
                     schreibe Nutzer in NutzerDB;
    "add"  + Gruppe: erzeuge Gruppe;
                     schreibe Gruppe in NutzerDB;
    "del"  + Gruppe: lösche Gruppe aus NutzerDB;
    "edit" + Gruppe: lies Gruppe aus NutzerDB;
                     ändere Gruppe;  * Name, Nutzer usw. *
                     schreibe Gruppe in NutzerDB;
  end case;
  Bestätigung_NutzerDB_Änderung := "successful";
end
```

Prozeß: **Login_prüfen** Datum: 01.05.98
Bearbeiter: Jan Horbach

Voraussetzung: Netzverbindung funktioniert ohne Fehler

```
begin
  if Nutzer_Login.<Nutzer>Name enthalten in NutzerDB then
    schreibe Nutzer_Login.<Nutzer>Name in Nutzer;
  else
    Login_Resultat := "login incorrect";
    return;
  end if;
  if Nutzer_Login.Passwort = Nutzer.Passwort then
    schreibe Nutzer_Login.Passwort in Nutzer;
  else
    Login_Resultat := "login incorrect";
    return;
  end if;
  if Verbindung zu Nutzer_Login.Server erfolgreich then
    schreibe Nutzer_Login.Server in Nutzer;
  else
    Login_Resultat := "unknown server";
    return;
  end if;
  Login_Resultat := "successful";
end
```

Prozeß: **Entwurf_drucken** Datum: 02.05.98
Bearbeiter: Jan Horbach

Voraussetzung: Entwurf (Entity + Architectures, evtl. Packages) ist vorhanden und Zugriff ist gestattet, Drucker ist angeschlossen und funktioniert

```
begin
  lies Entity;
  lies Entity.Architectures aus ProjektDB;
  if Architecture neuer als entsprechende Entity.Architecture then
    setze entsprechende Entity.Architecture auf Architecture;
  end if;
  if Entity verwendet Packages then
    lies Entity.Packages aus ProjektDB;
  end if;
  if existiert Entity.Architecture.Struktur then
    drucke Entity.Architecture als Grafik;
  else
    drucke Entity;
    drucke Entity.Architecture;
    if Entity verwendet Packages then
      drucke Entity.Packages;
    end if;
  end if;
  Druck_Resultat := "successful";
end
```

Prozeß: **Entwurf_exportieren** Datum: 02.05.98
Bearbeiter: Jan Horbach

Voraussetzung: Entwurf (Entity + Architectures, evtl. Packages) ist vorhanden und Zugriff ist gestattet, Zugriff auf lokalen Speicher ist möglich

```
begin
  lies Entity;
  lies Entity.Architectures aus ProjektDB;
  if Architecture neuer als entsprechende Entity.Architecture then
    setze entsprechende Entity.Architecture auf Architecture;
  end if;
  if Entity verwendet Packages then
    lies Entity.Packages aus ProjektDB;
  end if;
  if existiert Entity.Architecture.Struktur then
    wandle Entity.Architecture.Struktur in Text um;
  end if;
  schreibe Entity als Text in Lokaler_Speicher;
  schreibe Entity.Architecture als Text in Lokaler_Speicher;
  if Entity verwendet Packages then
    schreibe Entity.Packages als Text in Lokaler_Speicher;
  end if;
  Export_Resultat := "successful";
end
```

Prozeß: **Entity_bearbeiten** Datum: 01.06.98
Bearbeiter: Jan Horbach

Voraussetzung: Entity ist in ProjektDB vorhanden (bei "load") oder kann erstellt werden ("save"), Zugriff auf ProjektDB ist gestattet, Netzverbindung klappt ohne Probleme

```
begin
  case Entity_Bearbeitung of
    "new"          : erzeuge leere, uninitialisierte Entity;
                     schreibe Entity in Entity;
    "load" + Entity: lies Entity aus ProjektDB;
                     schreibe Entity in Entity;
    "save" + Entity: lies Entity aus Entity;
                     schreibe Entity in ProjektDB;
    "edit" + Entity: schreibe Entity in Entity;
  end case;
  Entity_Resultat := "successful";
end
```

Prozeß: **Architecture_bearbeiten** Datum: 30.04.98
Bearbeiter: Jan Horbach

Voraussetzung: Architecture ist in ProjektDB vorhanden (bei "load") oder kann erstellt werden ("save"), Zugriff auf ProjektDB ist gestattet, Netzverbindung klappt ohne Probleme

```
begin
  case Architecture_Bearbeitung of
    "new"                : erzeuge leere, uninitialisierte Architecture;
                           schreibe Architecture in Architecture;
    "load" + Architecture: lies Architecture aus ProjektDB;
                           schreibe Architecture in Architecture;
    "save" + Architecture: lies Architecture aus Architecture;
                           schreibe Architecture in ProjektDB;
    "edit" + Architecture: schreibe Architecture in Architecture;
  end case;
  Architecture_Resultat := "successful";
end
```

Prozeß: **Package_bearbeiten** Datum: 02.05.98
Bearbeiter: Jan Horbach

Voraussetzung: Package ist in ProjektDB vorhanden (bei "load") oder kann erstellt werden ("save"), Zugriff auf ProjektDB ist gestattet, Netzverbindung klappt ohne Probleme

```
begin
  case Package_Bearbeitung of
    "new"           : erzeuge Package;
                      schreibe Package in Package;
    "load" + Package: lies Package aus ProjektDB;
                      schreibe Package in Package;
    "save" + Package: lies Package aus Package;
                      schreibe Package in ProjektDB;
    "edit" + Package: schreibe Package in Package;
  end case;
  Package_Resultat := "successful";
end
```

1.3. Vorläufige Definition der Nutzerschnittstelle

Generell: Die Größenverhältnisse der Fenster und Dialoge in den Layoutskizzen sind in etwa maßstäblich. Sämtliche „normalen“ Fenster sind beliebig skalierbar, die Dialoge besitzen eine feste Größe. Alle Fenster können mit der Maus oder per Tastatur an eine beliebige Stelle des Bildschirms verschoben werden, was nur durch das zugrundeliegende Betriebssystem bzw. die Nutzeroberfläche oder den Fenstermanager eingeschränkt wird. Diese Änderungen bleiben solange bestehen, bis eine erneute Änderung erfolgt.

Die Fenster werden abhängig vom System auf dem Bildschirm plaziert und verdecken sich dabei eventuell. Dialoge befinden sich immer im Vordergrund, Menüs ebenfalls. Dadurch verdecken sie andere GUI-Elemente zeit- und teilweise. Alle Hauptfenster befinden sich direkt auf dem Desktop, es gibt also keine Unterfenster.

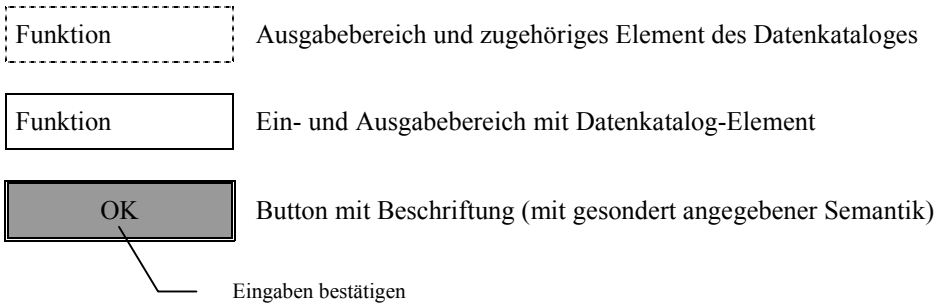
Farbgestaltung: Die Farbgestaltung hängt zum Teil von der benutzten Oberfläche ab. Auch die Benutzung von Farbschemas (z.B. unter Windows) beeinflußt die Darstellung. Deshalb können hier keine weiteren Aussagen getroffen werden.

Ein- und Ausgabegeräte

Als Eingabegeräte sind Tastatur und Maus vorgesehen. Für die Ausgabe wird ein Monitor (monochrom ist ausreichend, Farbe wird aber empfohlen) benötigt. Um einen Entwurf auszudrucken, ist zusätzlich ein Drucker (oder zumindest ein entsprechender Treiber, der die Ausgabe z.B. in eine Datei umleitet) notwendig.

Layoutentwurf

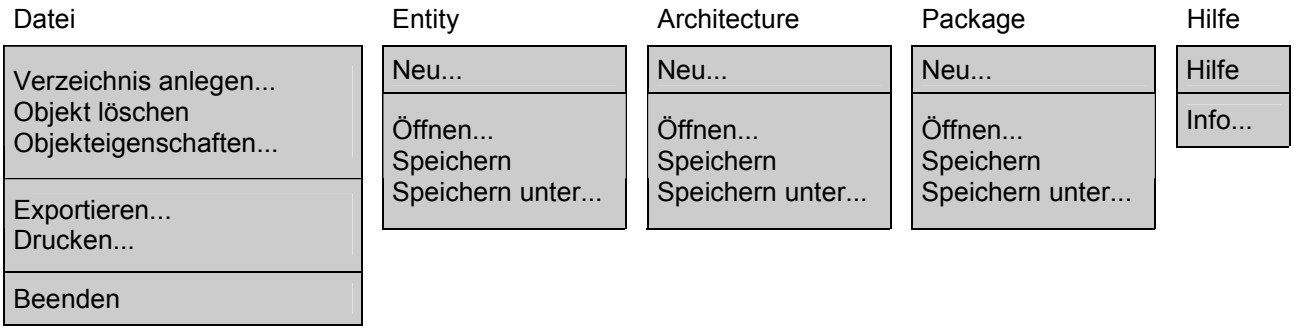
Legende:



Grundfenster:

Menüs:

Alle Untermenüs sind vom Typ pull-down.



Toolbar:

Im Toolbar befinden sich die wichtigsten Funktionen noch einmal in Form von Buttons, um die Arbeit mit VisualVHDL zu vereinfachen.

Statuszeile:

Hier werden aktuelle Statusmeldungen, Beschreibungen der Menüeinträge usw. ausgegeben.

Nicht gesondert aufgeführte Dialoge:

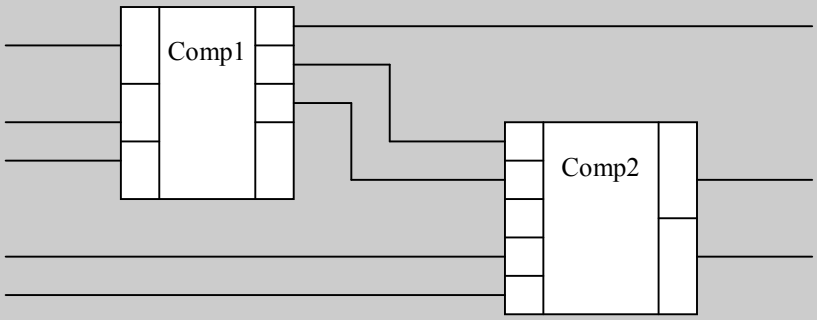
Der Info-Dialog enthält eine kurze Information zum Programm und über seine Autoren.

Alle anderen Dialoge (Exportieren, Drucken, Löschen) entsprechen den jeweiligen Standarddialogen des jeweiligen Systems und werden deshalb auch abhängig von der verwendeten Nutzeroberfläche dargestellt.

Bearbeiten-Fenster:

Das Bearbeiten-Fenster ist entweder als grafischer Editor zur Zusammenstellung von Strukturkomponenten oder als Texteditor zur Eingabe von Verhaltensbeschreibungen gedacht.

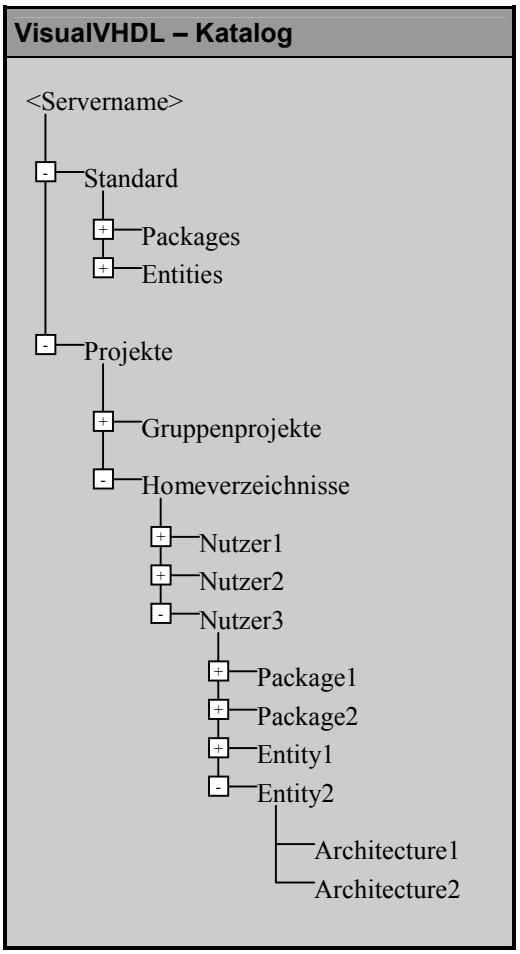
VisualVHDL – Architecture1 (Entity1)



VisualVHDL – behavior (mux4)

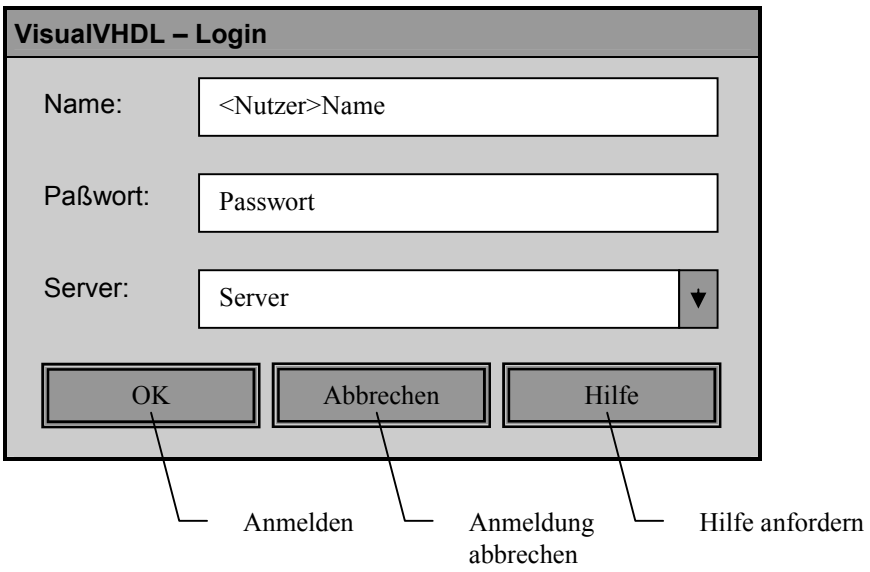
```
process (en,s,d)
  variable i : integer;
  begin
    case en is
      when '1' =>
        q <= '0'; n <= '1';
      when '0' =>
        i := 0;
        if s(0) = '1' then i := i + 1; end if;
        if s(1) = '1' then i := i + 2; end if;
        q <= d(i); n <= not d(i);
    end case;
  end process;
```

Katalogfenster:



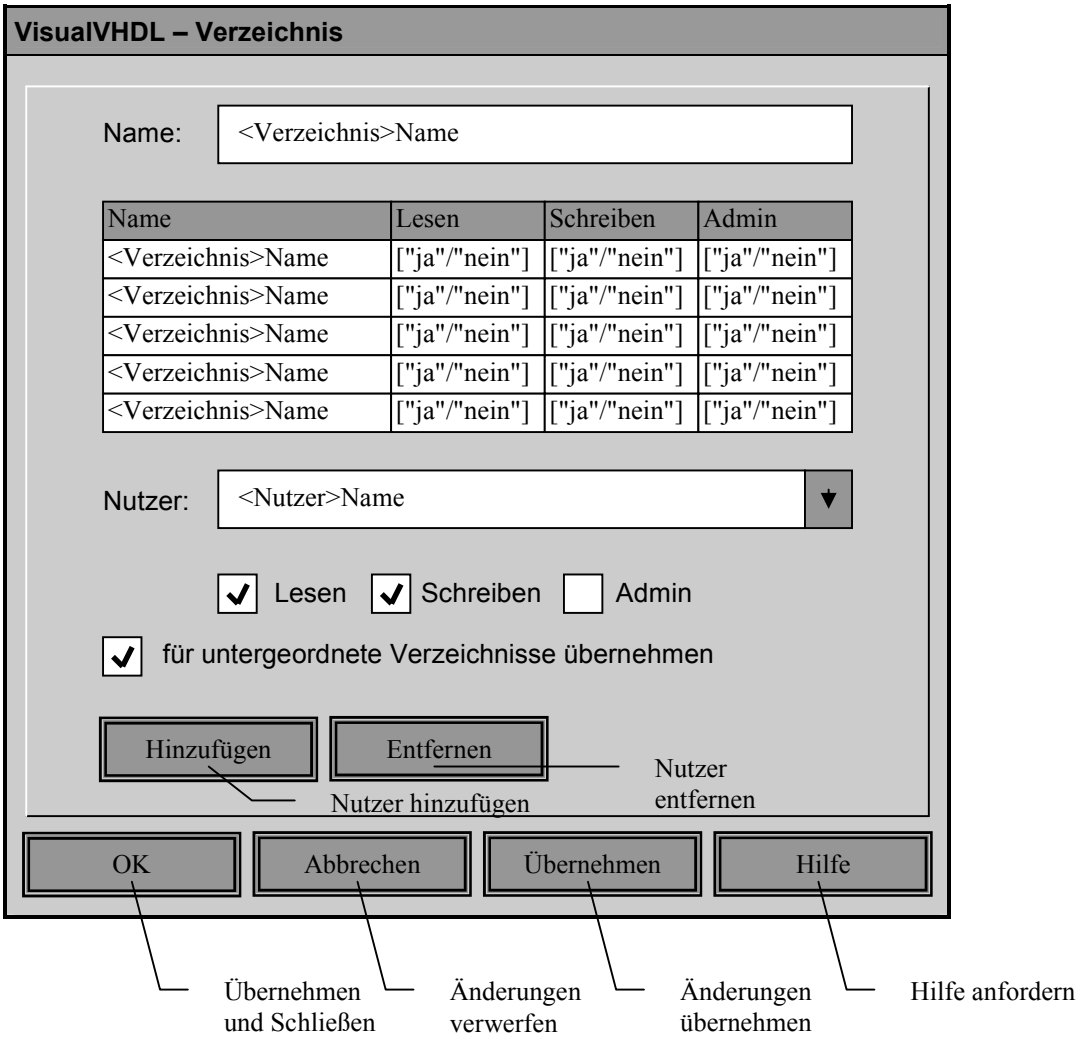
Login-Dialog:

Über diesen Dialog kann sich der Benutzer an das System anmelden. Dabei muß der Nutzername, ein Paßwort sowie der Server, auf dem der Datenkatalog (ProjektDB) liegt, angegeben werden.



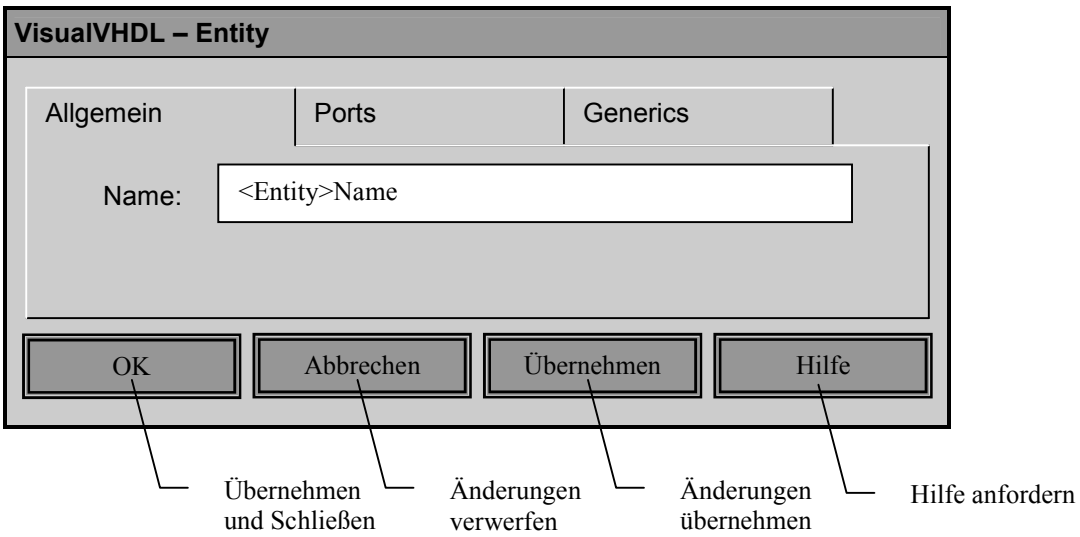
Verzeichnis-Dialog:

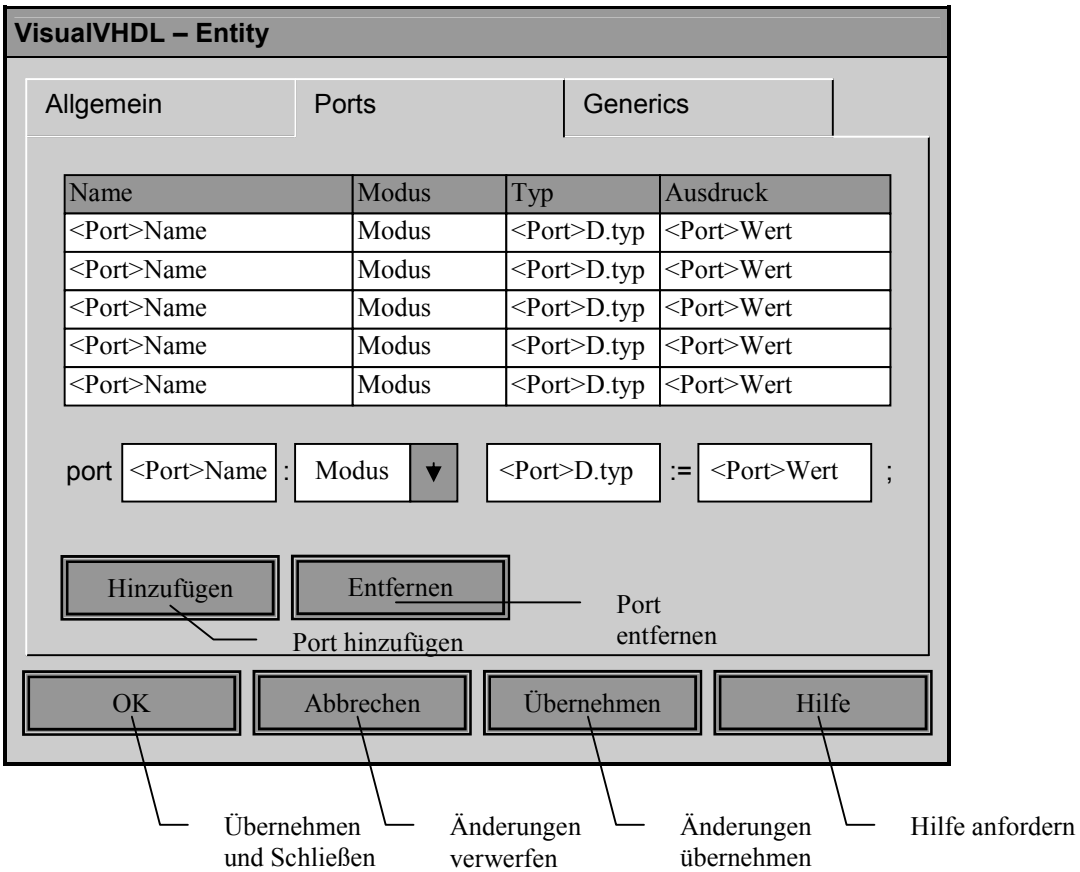
Im Verzeichnis-Dialog kann der Name des Verzeichnisses geändert werden, und die Vergabe von Nutzerrechten ist möglich.

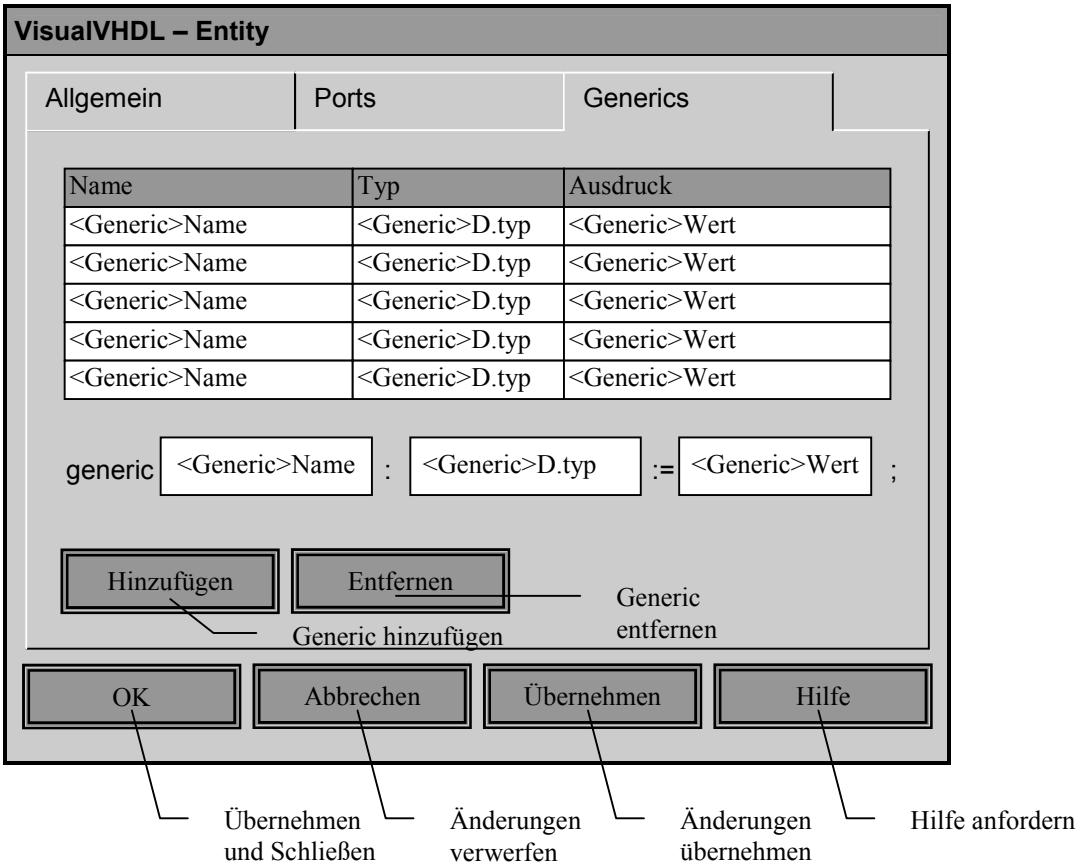


Entity-Dialog:

Im Entity-Dialog können der Name der Entity geändert sowie Ports und Generics festgelegt werden.

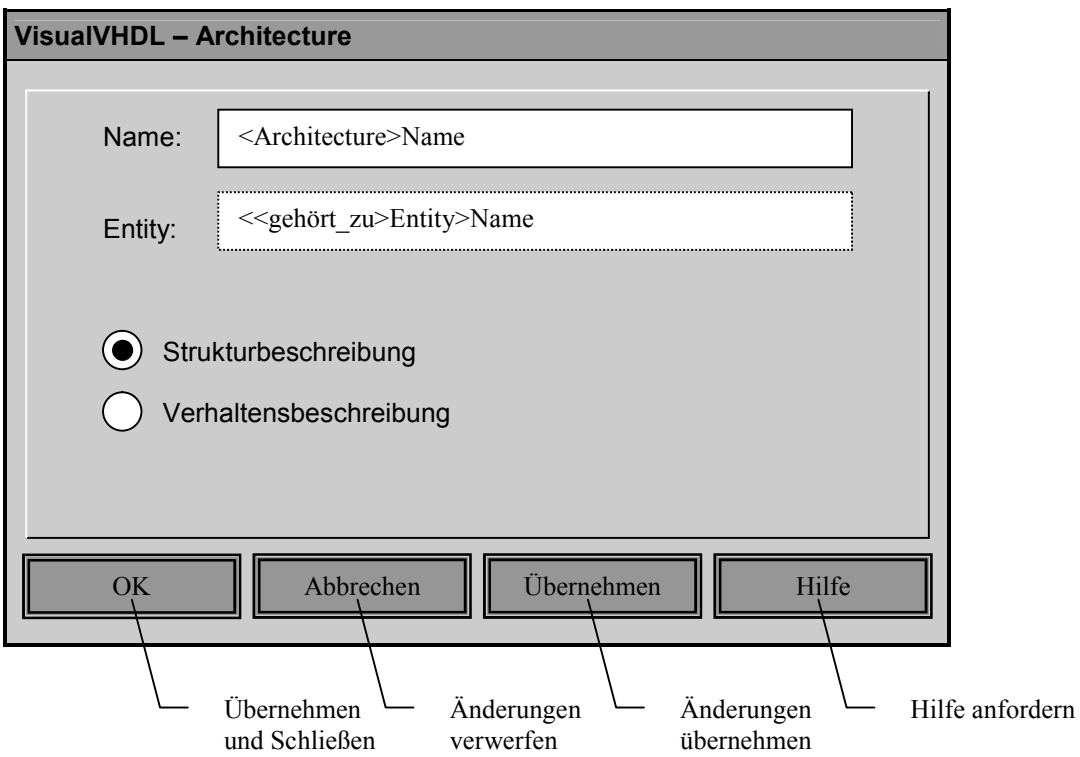






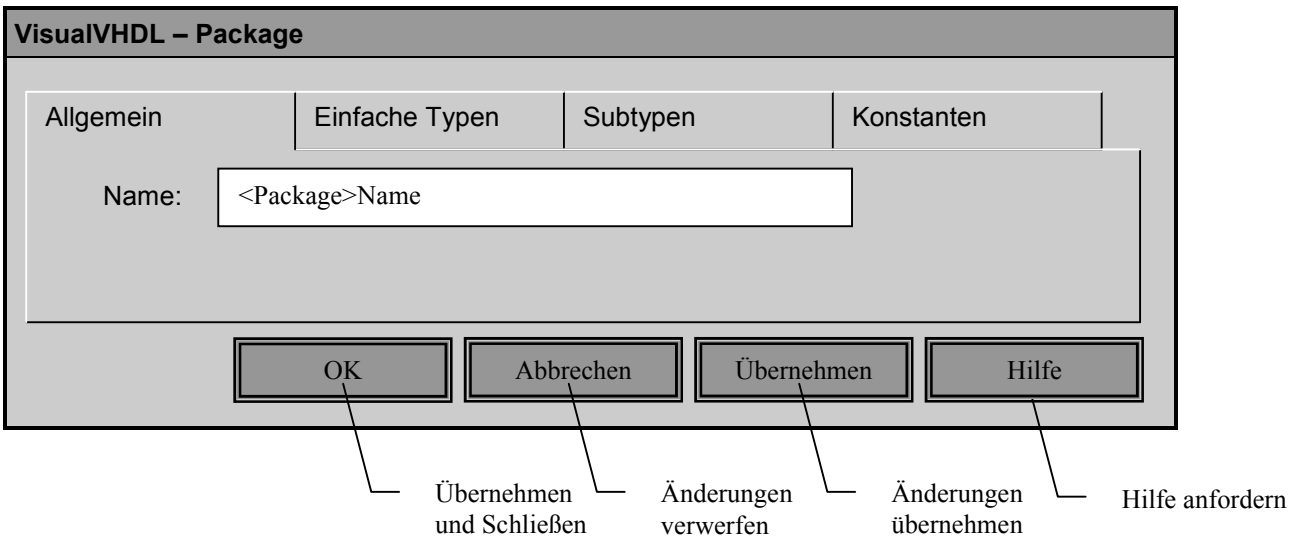
Architecture-Dialog:

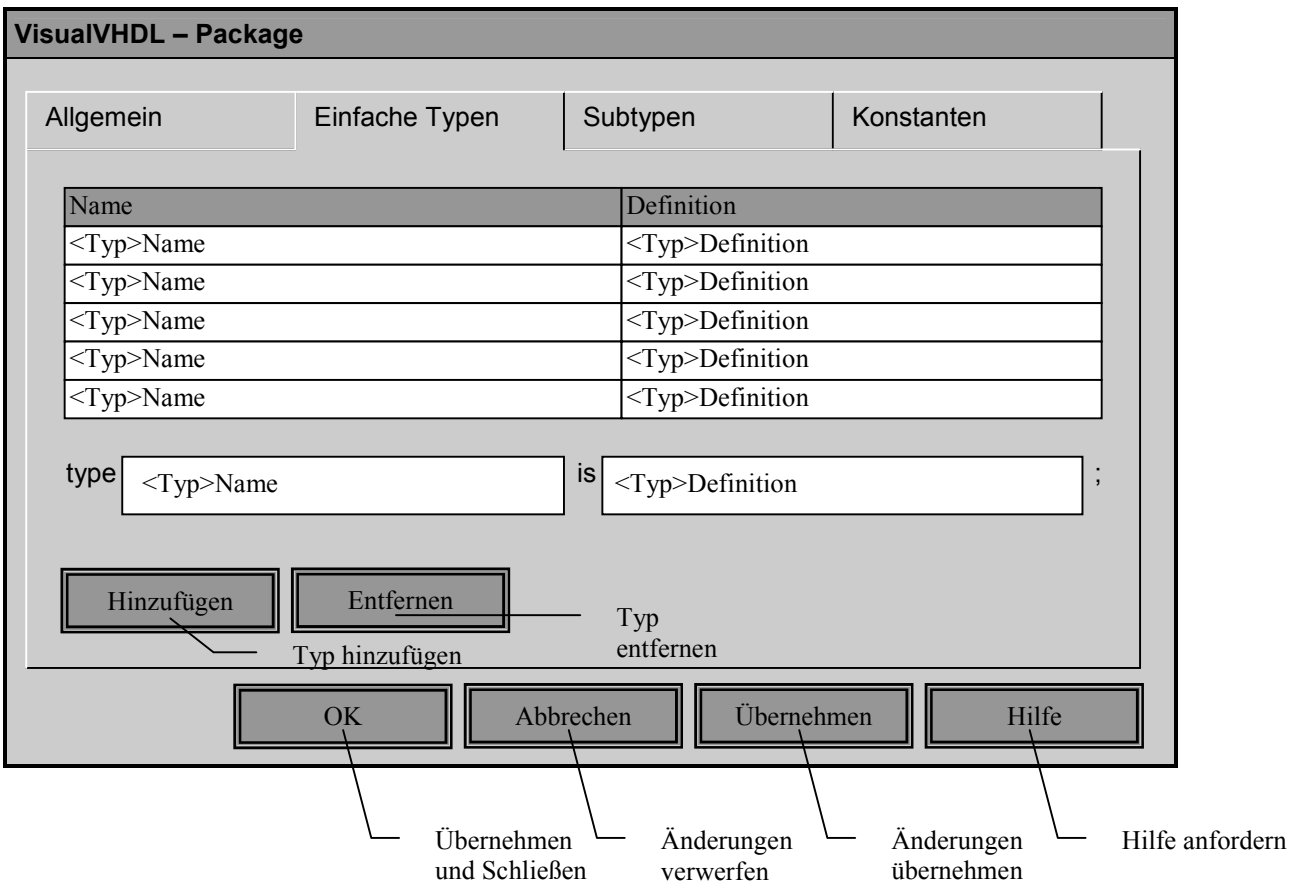
Im Architecture-Dialog kann der Name der Architecture geändert werden, und es erfolgt die Auswahl der Beschreibungsart.

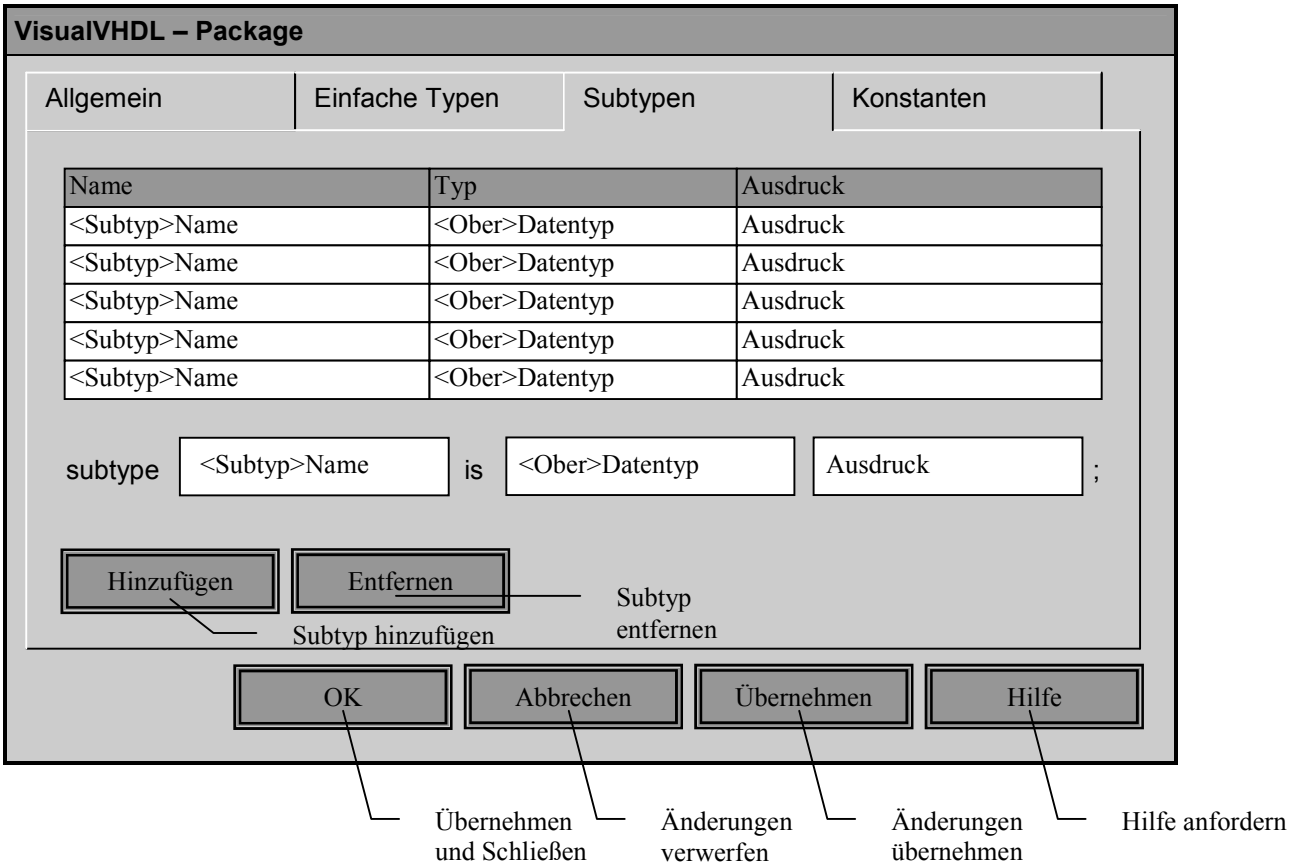


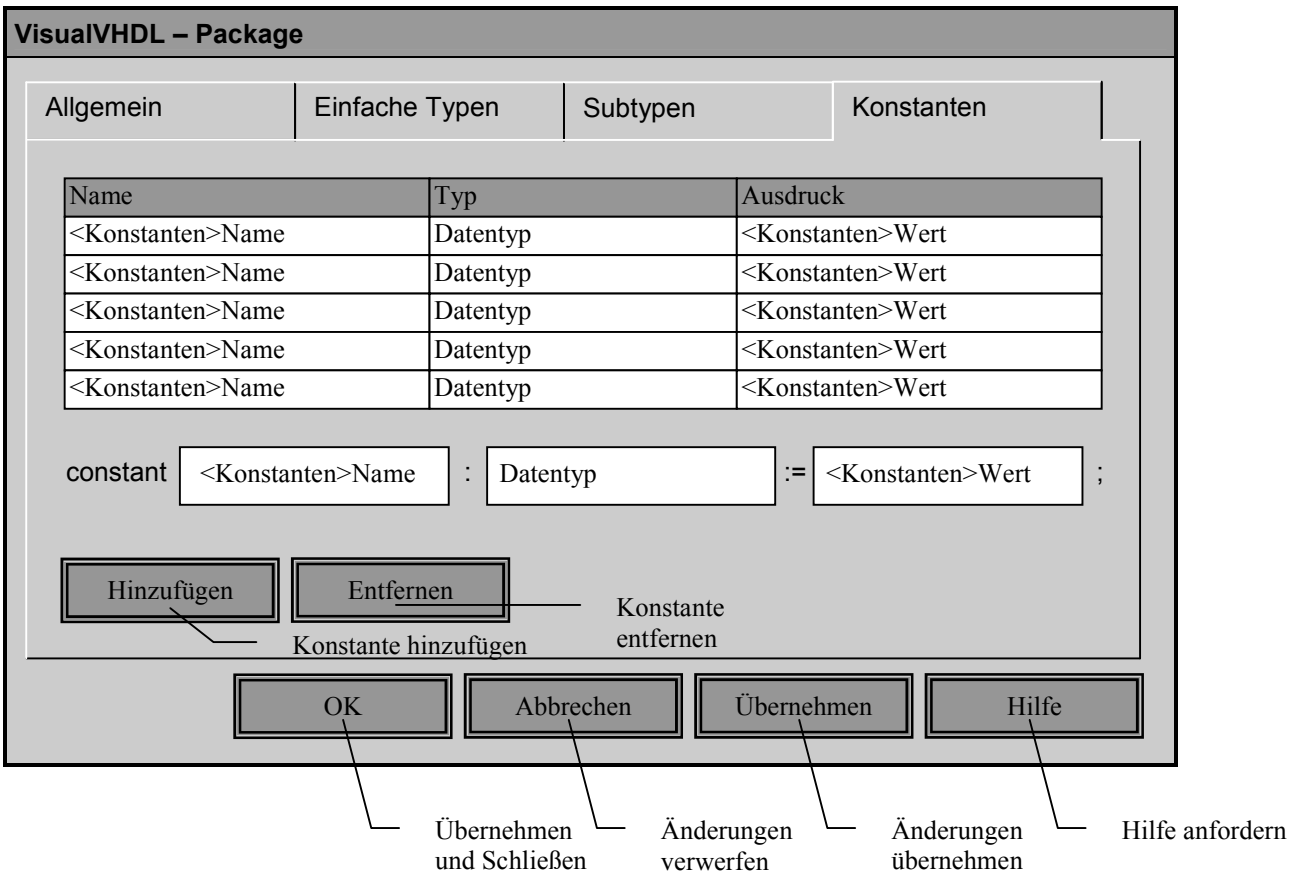
Package-Dialog:

Im Package-Dialog kann der Name geändert sowie Typen, Subtypen und Konstanten definiert werden.



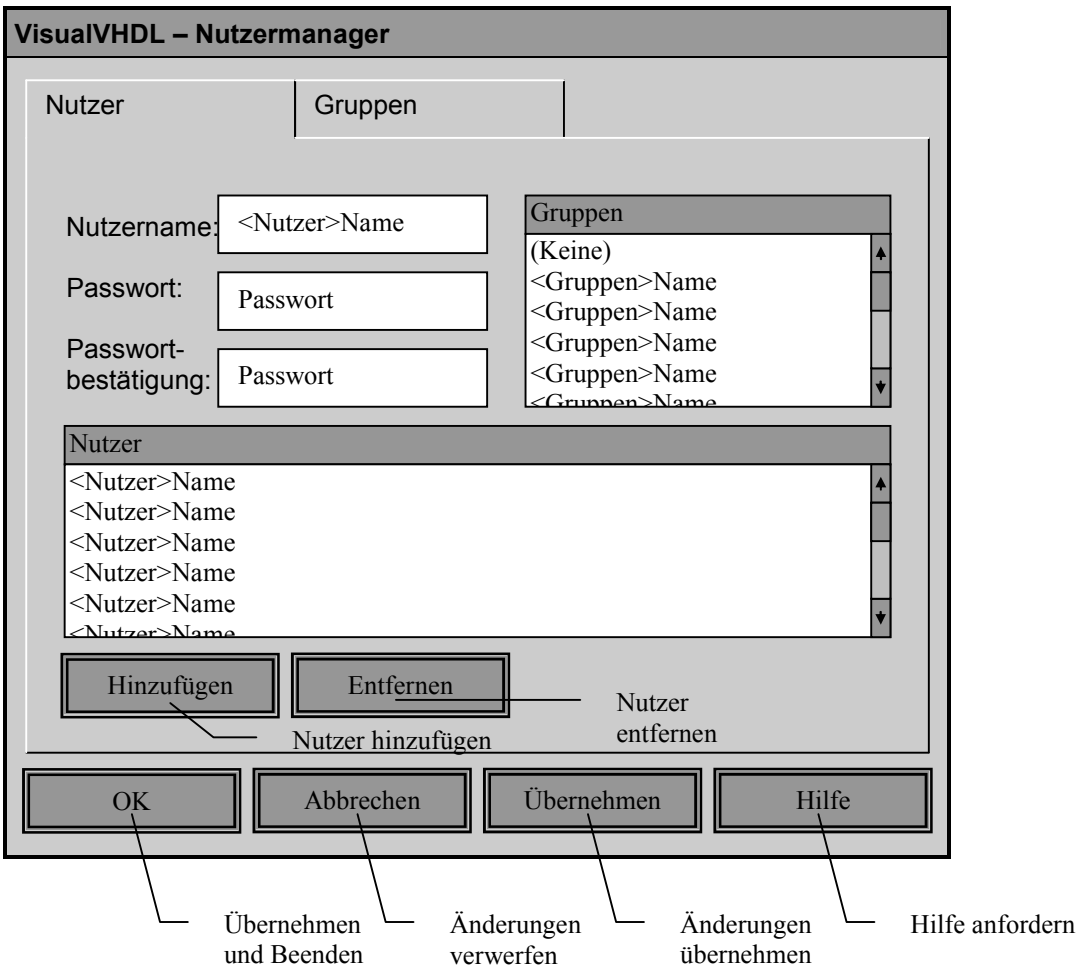


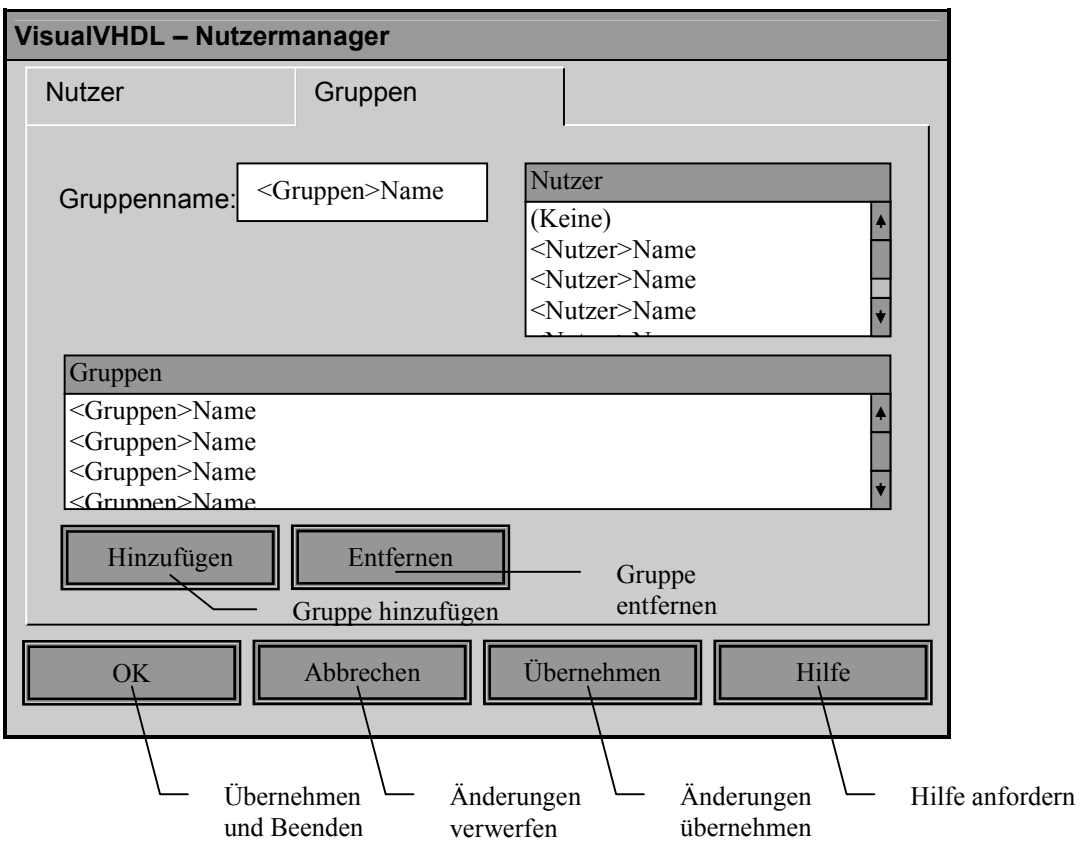

VisualVHDL – Package
Allgemein
Einfache Typen
Subtypen
Konstanten
Name
Typ
Ausdruck
<Subtyp>Name
<Ober>Datentyp
Ausdruck
subtype
is
;
Hinzufügen
Entfernen
Subtyp hinzufügen
Subtyp entfernen
OK
Abbrechen
Übernehmen
Hilfe
Übernehmen und Schließen
Änderungen verwerfen
Änderungen übernehmen
Hilfe anfordern


VisualVHDL – Package
Allgemein
Einfache Typen
Subtypen
Konstanten
Name
Typ
Ausdruck
<Konstanten>Name
Datentyp
<Konstanten>Wert
constant
:
:=
;
Hinzufügen
Entfernen
Konstante hinzufügen
Konstante entfernen
OK
Abbrechen
Übernehmen
Hilfe
Übernehmen und Schließen
Änderungen verwerfen
Änderungen übernehmen
Hilfe anfordern

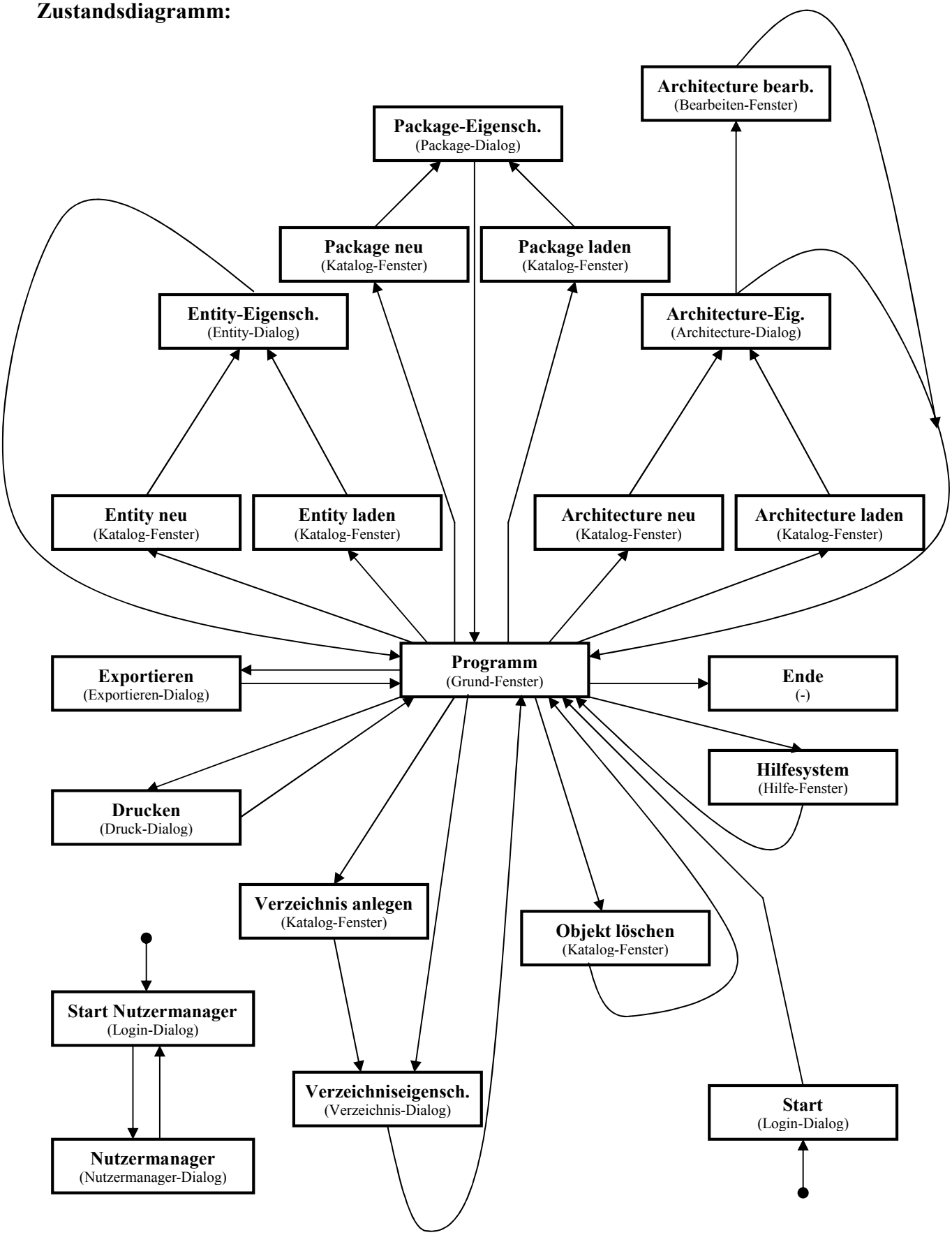
Nutzermanager:

Hier kann der Administrator Nutzer und Gruppen erstellen, löschen sowie Gruppen-Zugehörigkeiten ändern.





Zustandsdiagramm:



Aus Übersichtlichkeitsgründen sind die jeweiligen Speicherdialoge in diesem Zustandsdiagramm nicht mit aufgeführt. Aus allen Dialogboxen gelangt man mit "OK" oder mit "Abbrechen" zurück zum Hauptfenster.

2. Spezifikation der operationellen Anforderungen

2.1. Operationelle Anforderungen an die Daten und die Datenbasen

Für die Abschätzung des Speicherplatzes für Daten und Speicher wird von den Annahmen ausgegangen, daß für die Speicherung von gZahl 4 Byte, bei Zeichenketten 2 Byte pro Zeichen (Speicher für Runtime-Verwaltung nicht berücksichtigt!) und für (Objekt-)Referenzen 4 Byte benötigt werden.

Element	Struktur	Bytes
NutzerDB	**Nutzerdaten + Gruppen**	**s.u.**
Nutzerdaten	{Nutzer}	s.u.
Nutzer	@<Nutzer>Name + Passwort + <Mitglied_von>Gruppen + Server	20+40+4+200=264
Name	Zeichenkette10	20
Passwort	Zeichenkette20	40
Server	Zeichenkette100	200
Gruppen	{Gruppe}	s.u.
Gruppe	@<Gruppen>Name + Nutzerdaten	20+4=24
Name	Zeichenkette10	20
ProjektDB	**Verzeichnisse**	**s.u.**
Verzeichnisse	{Verzeichnis}	s.u.
Verzeichnis	@<Verzeichnis>Name + Verzeichnisse + Entities + Packages + ACL	s.u.
Name	Zeichenkette50	100
ACL	{Nutzerrechte} + {Gruppenrechte} * insgesamt mindestens einer *	s.u.
Nutzerrechte	Nutzer + Rechte	4+36=40
Gruppenrechte	Gruppe + Rechte	4+36=40
Rechte	(“read”) + (“write”) + (“changeACL”)	max. 36
Entities	{Entity}	s.u.
Entity	@<Entity>Name + Ports + Generics + Architectures + Packages	s.u.
Name	Zeichenkette50	100
Ports	{Port}	s.u.
Port	@<Port>Name + Modus + <Port>Datentyp + <Port>Wert	100+16+304+2000 =2420
Name	Zeichenkette50	100
Modus	[“in” / “out” / “inout” / “buffer” / “register”]	max. 16
Wert	Zeichenkette1000	2000
Generics	{Generic}	s.u.
Generic	@<Generic>Name + <Generic>Datentyp + <Generic>Wert	100+304+2000 =2404
Name	Zeichenkette50	100
Wert	Zeichenkette1000	2000
Architectures	{Architecture}	s.u.
Architecture	@<Architecture>Name + <gehört_zu>Entity + [Verhalten / Struktur]	s.u.
Name	Zeichenkette50	100
Verhalten	Zeichenkette * enthält einen VHDL-Quelltext *	max. 64KByte*2
Struktur	Components + Leitungen	s.u.
Components	{Component}	s.u.
Component	@Entity + Position + Größe * eine grafische Komponente *	4+8+8=20
Position	<Position>x + <Position>y	4+4=8
x	gZahl * x-Position auf dem Bildschirm *	4
y	gZahl * y-Position auf dem Bildschirm *	4
Größe	<Größe>x + <Größe>y	4+4=8
x	gZahl * x-Ausdehnung auf dem Bildschirm *	4
y	gZahl * y-Ausdehnung auf dem Bildschirm *	4
Leitungen	{Leitung}	s.u.
Leitung	<Leitung>von + <Leitung>nach	8+8=16
von	Entity + Port	4+4=8
nach	Entity + Port	4+4=8

Element	Struktur	Bytes
Packages	{Package}	s.u.
Package	@<Package>Name + Datentypen + Konstanten	s.u.
Name	Zeichenkette50	100
Datentypen	{Datentyp}	s.u.
Datentyp	[Typ / Subtyp]	max. 304
Typ	@<Typ>Name + <Typ>Definition	100+200=300
Name	Zeichenkette50	100
Definition	Zeichenkette100 * z.B. Aufzähltyp *	200
Subtyp	@<Subtyp>Name + <Ober>Datentyp + Ausdruck	100+4+200=304
Name	Zeichenkette50	100
Ausdruck	Zeichenkette100 * z.B. Bereichseingrenzung *	200
Konstanten	{Konstante}	s.u.
Konstante	@<Konstanten>Name + Datentyp + <Konstanten>Wert	100+304+200=604
Name	Zeichenkette50	100
Wert	Zeichenkette100	200

Element	Struktur	Bytes
Dynamische Daten		
Nutzer_Login	<Nutzer>Name + Passwort + Server	20+40+200=260
Name	Zeichenkette10	20
Passwort	Zeichenkette20	40
Server	Zeichenkette100	200
Login_Resultat	[“successful” / “login incorrect” / “unknown server”]	max. 30
Entity_Bearbeitung	[“new” / “load” + Entity / “save” + Entity]	max. 8+4=12
Entity_Resultat	[“successful” / “error”]	max. 20
Architecture_Bearbeitung	[“new” / “load” + Architecture / “save” + Architecture]	max. 8+4=12
Architecture_Resultat	[“successful” / “error”]	max. 20
Package_Bearbeitung	[“new” / “load” + Package / “save” + Package]	max. 8+4=12
Package_Resultat	[“successful” / “error”]	max. 20
Entwurf_Export	“export” + Architecture	12+4=16
Export_Resultat	[“successful” / “error”]	max. 20
Entwurf_Druck	“print” + Architecture	10+4=14
Druck_Resultat	[“successful” / “error”]	max. 20
NutzerDB_Änderung	[“add” / “del” / “edit”] + [Nutzer / Gruppe]	max. 8+4=12
Bestätigung_NutzerDB_ Änderung	[“successful” / “error”]	max. 20
ProjektDB_Änderung	[[“create” / “del” / “edit”] + Verzeichnis / “del” + [Verzeichnis / Entity / Architecture / Package]]	max. 8+4=12
Bestätigung_ProjektDB_ Änderung	[“successful” / “error”]	max. 20

Abschätzungen:

Element	Struktur	Minimum	Mittel	Maximum
NutzerDB	**Nutzerdaten + Gruppen**	**264+0=264**	**2640+1200 =3840**	**132000+2400 =134 400**
Nutzerdaten	{Nutzer}	1*264=264	100*264= 26400	500*264= 132 000
Gruppen	{Gruppe}	0*24=0	50*24=1200	100*24=2400
ProjektDB	**Verzeichnisse**	**152**	**ca. 103,6MB**	**ca. 21,9GB**
Verzeichnisse	{Verzeichnis}	1*152=152	150*2*361984 =108 595 200	600*5* 7853444 =23560332000
Verzeichnis	@<Verzeichnis>Name + Verzeichnisse + Entities + Packages + ACL	100+4+0+0+40 =152	100+4+346600 +9280+6000 =361 984	100+4+ 7768240+ 61100+24000 =7 853 444
ACL	{Nutzerrechte} + {Gruppenrechte} * insgesamt mindestens einer *	1*40=40	150*40=6000	600*40=24000
Entities (pro Verzeichnis)	{Entity}	0*408=0	5*69320 =346 600	10*776824 =7 768 240
Entity	@<Entity>Name + Ports + Generics + Architectures + Packages	100+0+0+304+ 4=408	100+24200+ 4808+40208+4 =69320	100+96800+ 24040+655880 +4=776 824
Ports	{Port}	0*2420=0	10*2420 =24200	40*2420 =96800
Generics	{Generic}	0*2404=0	2*2404=4808	10*2404 =24040
Architectures (pro Entity)	{Architecture}	1*304=304	2*20104 =40 208	5*131176 =655 880
Architecture	@<Architecture>Name + <gehört_zu>Entity + [Verhalten / Struktur]	100+4+200 =304	100+4+20000 =20104	100+4+131072 =131 176
Verhalten	Zeichenkette * enthält einen VHDL-Quelltext *	100*2=200	10000*2 =20000	65536*2 =131 072
Struktur	Components + Leitungen	40+32=72	100+1000= 1100	800+32000 =32800
Components	{Component}	2*20=40	5*20=100	40*20=800
Leitungen	{Leitung}	2*16=32	5*10*20=1000	40*40*20 =32000
Packages (pro Verzeichnis)	{Package}	0*1008=0	2*4640=9280	5*12220 =61100
Package	@<Package>Name + Datentypen + Konstanten	100+304+604 =1008	100+1520+ 3020=4640	100+6080+ 6040=12220
Datentypen	{Datentyp}	1*304=304	5*304=1520	20*304=6080
Konstanten	{Konstante}	1*604=604	5*604=3020	10*604=6040

Gesamtsystem: Die voraussichtliche Nutzungsdauer beträgt voraussichtlich 10 Jahre (maximal 50 Jahre). Die Speicherabschätzungen beziehen sich dabei auf die (maximale) Nutzungszeit.

Nutzer: Es wird von einer durchschnittlichen Nutzeranzahl von 100 Nutzern und 50 Gruppen (maximal 500 Nutzer und 100 Gruppen) ausgegangen. Die Maximalabschätzung geht davon aus, daß 500 Nutzer und 100 Gruppen jeweils Entwürfe mit maximaler Ausnutzung des Systems beschreiben. Im praktischen Einsatz dürfte das kaum vorkommen, so daß die 21,9GB für die ProjektDB wohl nicht erreicht werden.

Genauigkeit: Es gibt keine besonderen Genauigkeitsanforderungen, da VisualVHDL nur mit Ganzzahlen arbeitet.

2.2. Operationelle Anforderungen an die Datenströme

Innerhalb des Systems treten keine allzu umfangreichen Datenströme auf. Nur bei der Arbeit mit dem Server (Laden, Speichern) und beim Exportieren können erhebliche Zeitverzögerungen auftreten, da pro Nutzer Datenmengen von bis zu einem Megabyte auf einmal verschickt werden können. Über die genaue Geschwindigkeit läßt sich keine Aussage treffen, da dies vom verwendeten Server und der Netzanbindung abhängt.

2.3. Operationelle Anforderungen an die Prozesse

Innerhalb von VisualVHDL treten aufgrund der geringen Datenströme auch nur geringe Verzögerungen auf. Lediglich bei der Serverarbeit sind wieder längere Antwortzeiten zu erwarten. Wo dies möglich ist (Speichern, Exportieren) sind daher entsprechend aufwendige Funktionen als nebenläufige Threads zu realisieren, um den Nutzer sowenig wie möglich zu behindern. Falls der Server einmal ausgefallen ist, sollte es einen Timeout geben, damit der Client nicht ewig wartet. Auch der Server muß Mechanismen zur Verfügung haben, um einen eventuell abgestürzten oder anderweitig verlorengegangenen Clienten automatisch nach einer gewissen Zeit zu disconnecten.

3. Spezifikation wichtiger Qualitätsanforderungen

3.1. Benutzbarkeit

In VisualVHDL existieren zwei Haupt-Benutzerklassen, Nutzer und Administratoren. An den Nutzer werden keine besonderen Anforderungen in Hinsicht auf Arbeit mit VHDL vorausgesetzt, es ist jedoch empfehlenswert, sich schon vorher (z.B. im Rahmen des Studiums) mit VHDL beschäftigt zu haben. Es ist daher auf eine übersichtliche und einfache Oberfläche zu achten, die intuitiv verständlich ist. Der Administrator sollte jedoch zumindest etwas Erfahrung im Umgang mit Nutzer- und Gruppeneinrichtung mitbringen, obwohl der Nutzermanager so gehalten ist, daß ihn auch ein Laie versteht.

3.2. Zuverlässigkeit

Um Bedienfehler durch den Nutzer weitestgehend auszuschließen, ist auf eine einfache und klar verständliche Oberflächengestaltung zu achten. Sollten dennoch Eingabefehler und dergleichen auftreten, muß das Programm dies dem Nutzer melden und eine Korrekturmöglichkeit anbieten.
Besonderes Augenmerk ist auf die Client-Server-Kommunikation zu legen: Wie schon in Punkt 2.3 beschrieben, müssen geeignete Mechanismen existieren, um den Ausfall eines der beiden Rechner zu kompensieren, ohne daß der andere Rechner ebenfalls ausfällt.

3.3. Integrität

Aufgrund der Einsatzbedingungen des Systems (Entwürfe und Nutzerdaten werden über Netzwerk zum und vom Server übertragen), werden besondere Anforderungen an die Sicherheit gestellt. Paßworte müssen generell verschlüsselt übertragen werden. Als Verschlüsselungsalgorithmus ist z.B. DES zu verwenden.

3.4. Flexibilität

VisualVHDL ist ein Prototyp, der später noch umfassend erweitert werden kann, z.B. Multiserverunterstützung, Einbau eines Parsers zur Analyse von textuell beschriebenen Strukturbeschreibung und Umwandlung in eine grafische Beschreibung, Anbindung von Synopsys zur Simulation der entworfenen Bausteine usw. Daher ist darauf zu achten, daß das System so aufgebaut wird, daß eine Erweiterung später leicht möglich ist. Durch den objektorientierten Ansatz bei der Programmierung ist das hinreichend gegeben.

3.5. Portabilität

Durch Verwendung der Programmiersprache Java ist das System auf allen Plattformen lauffähig, auf denen eine Java 1.1 kompatible Java Virtual Machine existiert. Das sind zur Zeit u.a. die Systeme Windows 95/NT, Unix, Linux und MacOS. Dadurch und durch die Vermeidung von plattformspezifischen (native) Elementen ist eine große Portabilität gesichert.

4. Basismaschine und Entwicklungsumgebung

4.1. Basismaschine

4.1.1. Hardware

Für die minimale Hardwareausstattung der Basismaschine werden folgende Annahmen getroffen:

- Basisrechner: PC mindestens 486er CPU, 66 MHz, 16 MB RAM; andere Systeme analoge Leistung
- Monitor: mindestens 14“, monochrom ist ausreichend
- Tastatur, Maus
- Festplatte mit mindestens 10 MB freiem Speicher
- möglichst Drucker
- möglichst Netzanschluß

Das ist nur die Minimalvariante, mit der VisualVHDL läuft. Eine leistungsfähigere CPU, mehr Speicher, ein größerer Monitor (Farbe) sind jedoch zu empfehlen.

Der Server muß über Netzanschluß, eine möglichst große Festplatte (mindestens 4 GB) und viel Arbeitsspeicher (mindestens 64 MB) verfügen sowie eine möglichst hohe Taktfrequenz (mindestens 200 MHz) besitzen.

4.1.2. Betriebssystem

Als Betriebssystem sind sämtliche Systeme geeignet, für die eine JDK 1.1 kompatible Java Virtual Machine existiert, z.B. Windows 95, Windows NT, Unix, Linux und MacOS. Lange Dateinamen müssen unterstützt werden. Auf dem Server gilt das entsprechend.

4.1.3. Sonstige Basissoftware

Auf der Basismaschine müssen eine JDK 1.1 kompatible Java Virtual Machine (zumindest das Java Runtime Environment JRE) und die Klassen von Swing 1.0 installiert sein. Auf dem Server entsprechend. Zusätzlich ist bei Netzbetrieb Netzwerksoftware erforderlich.

4.2. Entwicklungsumgebung

Als Entwicklungsumgebung ist das JDK 1.1.6 mit Swing 1.0.1 vorgesehen.

5. Anwenderdokumentation

5.1. Produktzweck

Die Entwicklungsumgebung VisualVHDL unterstützt den VHDL-Programmierer (besonders den Studenten) bei der Generierung von VHDL-Quellcode aus einer grafischen Strukturbeschreibung. Dazu bietet das System einen grafischen Editor, in dem die Komponenten zu Strukturarchitekturen zusammengesetzt werden. Im Editor werden die Komponenten und Signalleitungen interaktiv miteinander verbunden. Dabei kann der Entwickler auf schon vorhandene Entwürfe zurückgreifen, die alle in einer multiuserfähigen Datenbank abgelegt sind, die sich auf einem Server befindet. Von ihr werden Entities, Architectures und Packages verwaltet. Um die Sicherheit der Daten zu gewährleisten, werden allen Datenobjekten spezielle Zugriffsrechte auf Nutzer- und Nutzergruppenebene zugeteilt.

5.2. Basismaschine und Ressourcenanforderungen

Für die minimale Hardwareausstattung der Basismaschine werden folgende Annahmen getroffen:

- Basisrechner: PC mindestens 486er CPU, 66 MHz, 16 MB RAM; andere Systeme analoge Leistung
- Monitor: mindestens 14“, monochrom ist ausreichend
- Tastatur, Maus
- Festplatte mit mindestens 10 MB freiem Speicher
- möglichst Drucker
- möglichst Netzanschluß

Das ist nur die Minimalvariante, mit der VisualVHDL läuft. Eine leistungsfähigere CPU, mehr Speicher, ein größerer Monitor (Farbe) sind jedoch zu empfehlen.

Das System kann auf jedem Rechner mit einer Java 1.1 kompatiblen Java Virtual Machine und Swing 1.0 betrieben werden. Netzanbindung ist empfohlen, der Server kann aber auch lokal betrieben werden.

5.3. Nutzerklassen

Die Nutzer von VisualVHDL sind hauptsächlich in zwei Gruppen aufgeteilt: Die Administratoren, die Rechte festlegen können, und die „normalen“ Nutzer, die sich der Datenbank bedienen können, um ihre eigenen Entwürfe zu entwickeln.

5.4. Bedienungsanleitung

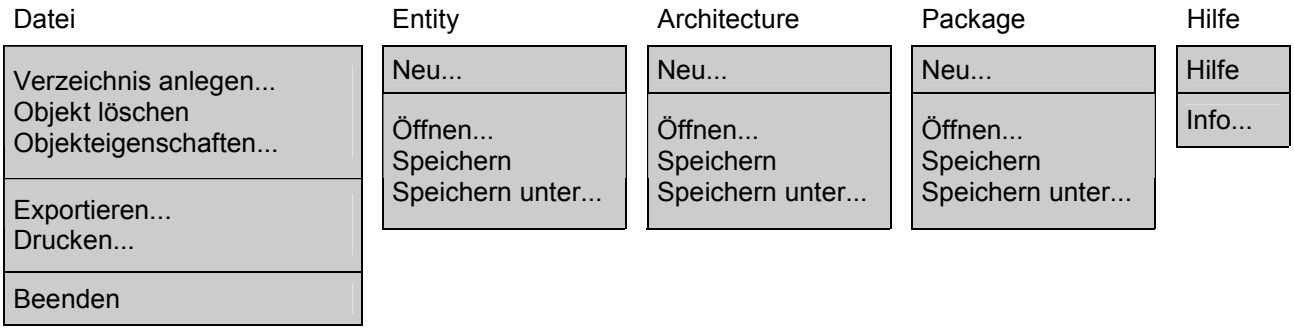
VisualVHDL wird zusammen mit dem Java Runtime Environment (JRE) auf einer CD-ROM ausgeliefert. Für die verschiedenen Plattformen existieren unterschiedliche Installationsprogramme, die je nach System aufgerufen werden müssen, um VisualVHDL zu installieren. Nach der Installation ist das Programm über die Benutzeroberfläche startbar oder muß von einer Konsole z.B. mit "java VisualVHDL" gestartet werden. Genauere Angaben dazu sind in der Datei readme.txt zu finden.

Nach erfolgreicher Installation kann das Programm wie oben beschrieben gestartet werden. Es erscheint als erstes der Login-Dialog, in dem der Nutzer sich anmelden muß. Danach erscheinen das Grundfenster und das Katalogfenster. Im Hauptfenster befindet sich eine Menüzeile, von der aus alle Operationen ausführbar sind. Zusätzlich sind einige Funktionen noch über den Toolbar erreichbar. Die Bedienung der Oberfläche erfolgt wie auf dem jeweiligen System üblich (vorzugsweise per Maus).

Grundfenster:

Menüs:

Alle Untermenüs sind vom Typ pull-down.



Toolbar:

Im Toolbar befinden sich die wichtigsten Funktionen noch einmal in Form von Buttons, um die Arbeit mit VisualVHDL zu vereinfachen.

Statuszeile:

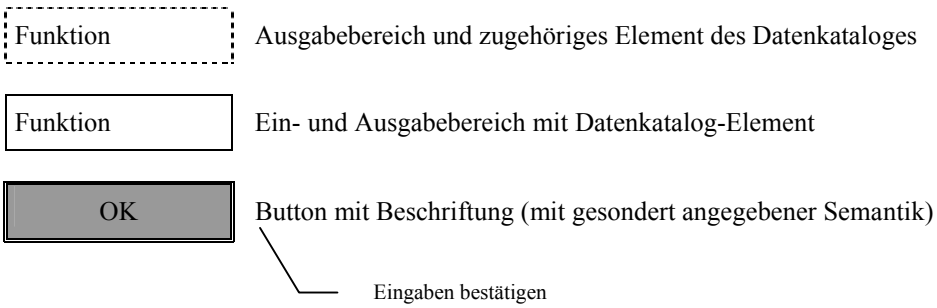
Hier werden aktuelle Statusmeldungen, Beschreibungen der Menüeinträge usw. ausgegeben.

Nicht gesondert aufgeführte Dialoge:

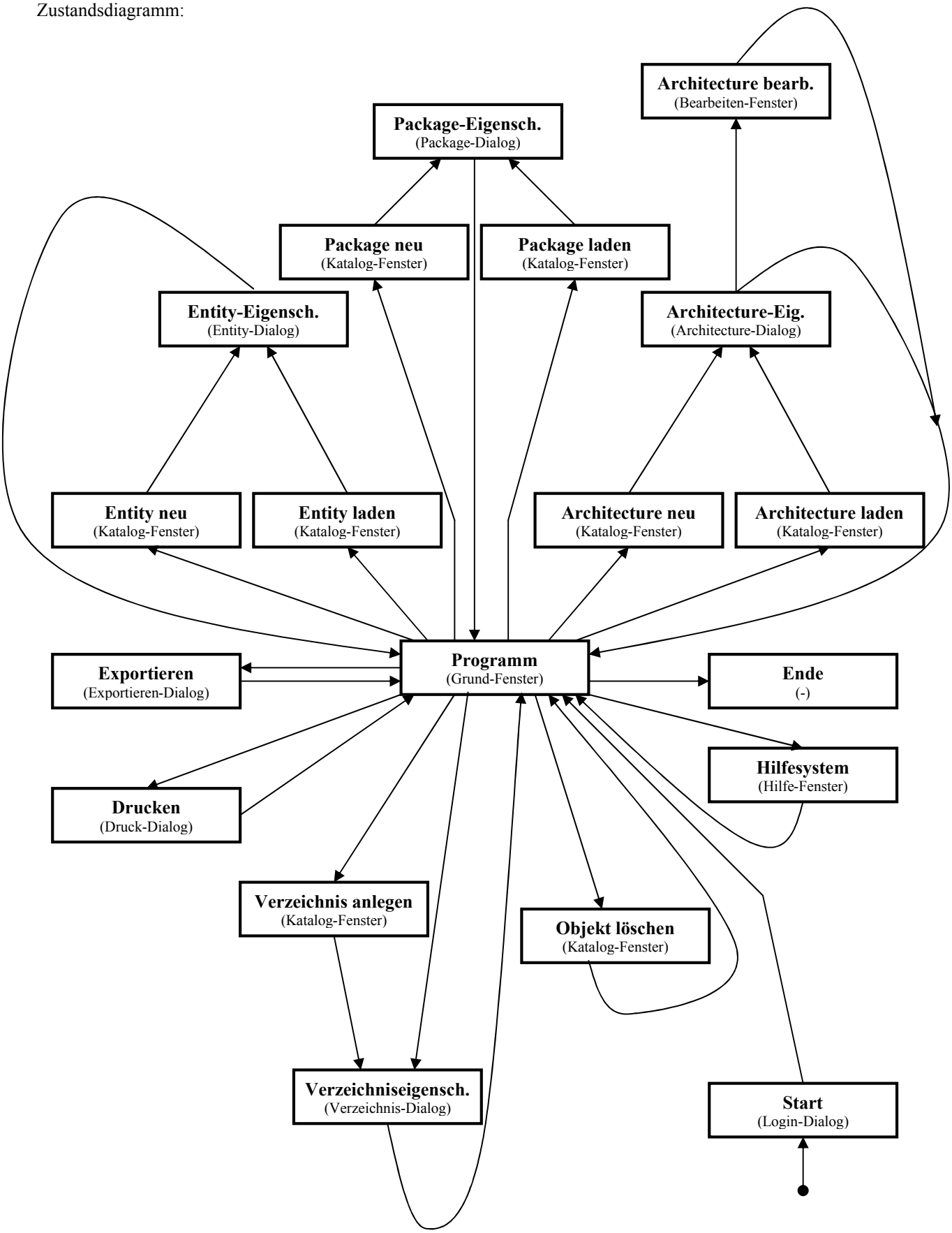
Der Info-Dialog enthält eine kurze Information zum Programm und über seine Autoren.

Alle anderen Dialoge (Exportieren, Drucken, Löschen) entsprechen den jeweiligen Standarddialogen des jeweiligen Systems und werden deshalb auch abhängig von der verwendeten Nutzeroberfläche dargestellt.

Legende zur Darstellung:



Welche Aktionen in welchem Zustand von VisualVHDL ausgeführt werden können, zeigt das folgende Zustandsdiagramm:

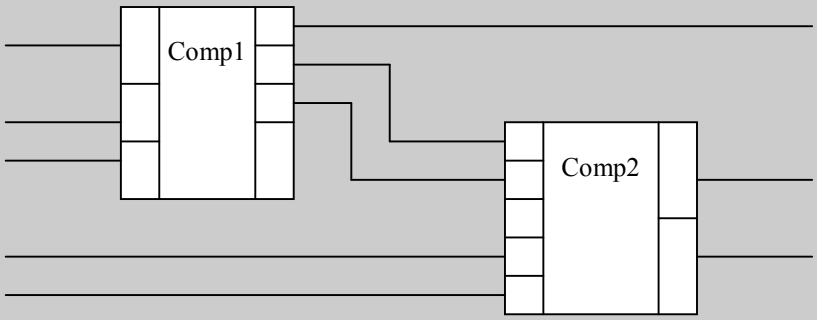


Aus Übersichtlichkeitsgründen sind die jeweiligen Speicherdialoge in diesem Zustandsdiagramm nicht mit aufgeführt. Aus allen Dialogboxen gelangt man mit "OK" oder mit "Abbrechen" zurück zum Hauptfenster.

Bearbeiten-Fenster:

Das Bearbeiten-Fenster ist entweder als grafischer Editor zur Zusammenstellung von Strukturkomponenten oder als Texteditor zur Eingabe von Verhaltensbeschreibungen gedacht.

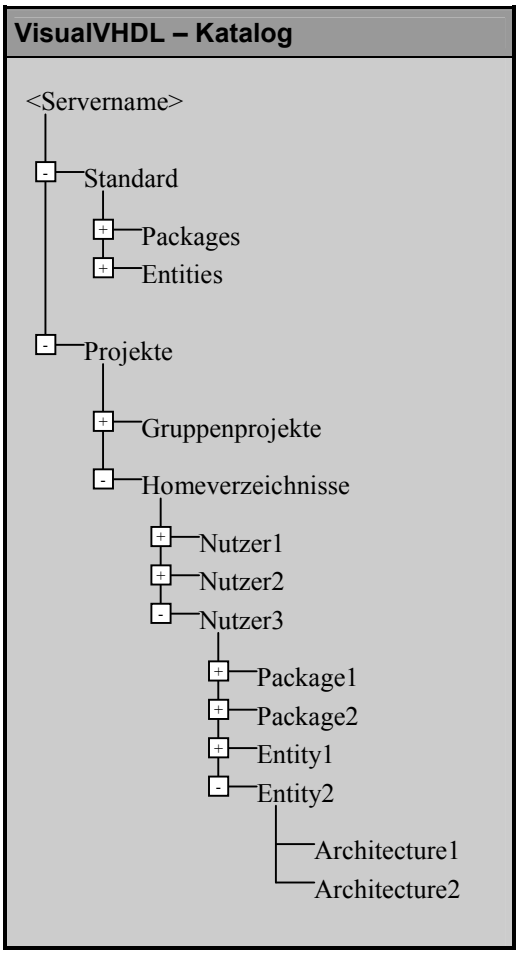
VisualVHDL – Architecture1 (Entity1)



VisualVHDL – behavior (mux4)

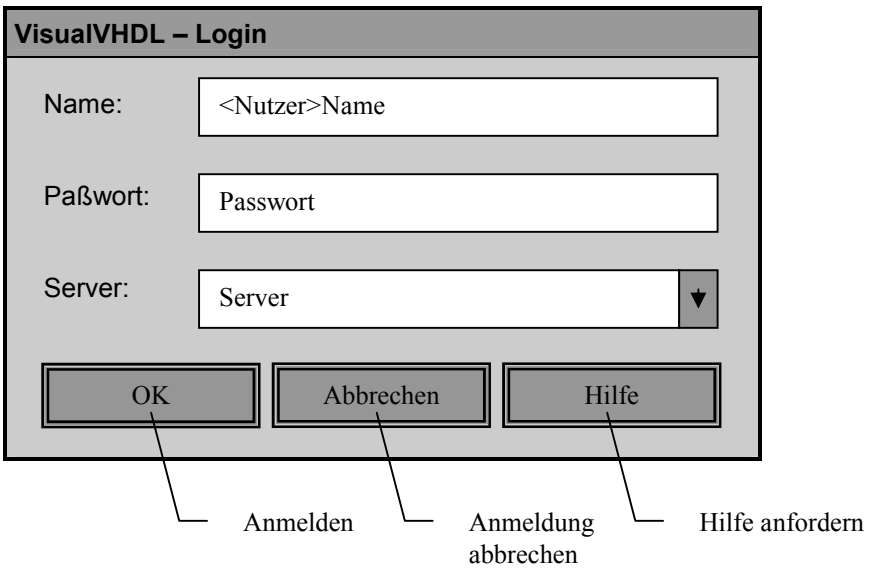
```
process (en,s,d)
  variable i : integer;
  begin
    case en is
      when '1' =>
        q <= '0'; n <= '1';
      when '0' =>
        i := 0;
        if s(0) = '1' then i := i + 1; end if;
        if s(1) = '1' then i := i + 2; end if;
        q <= d(i); n <= not d(i);
    end case;
  end process;
```

Katalogfenster:



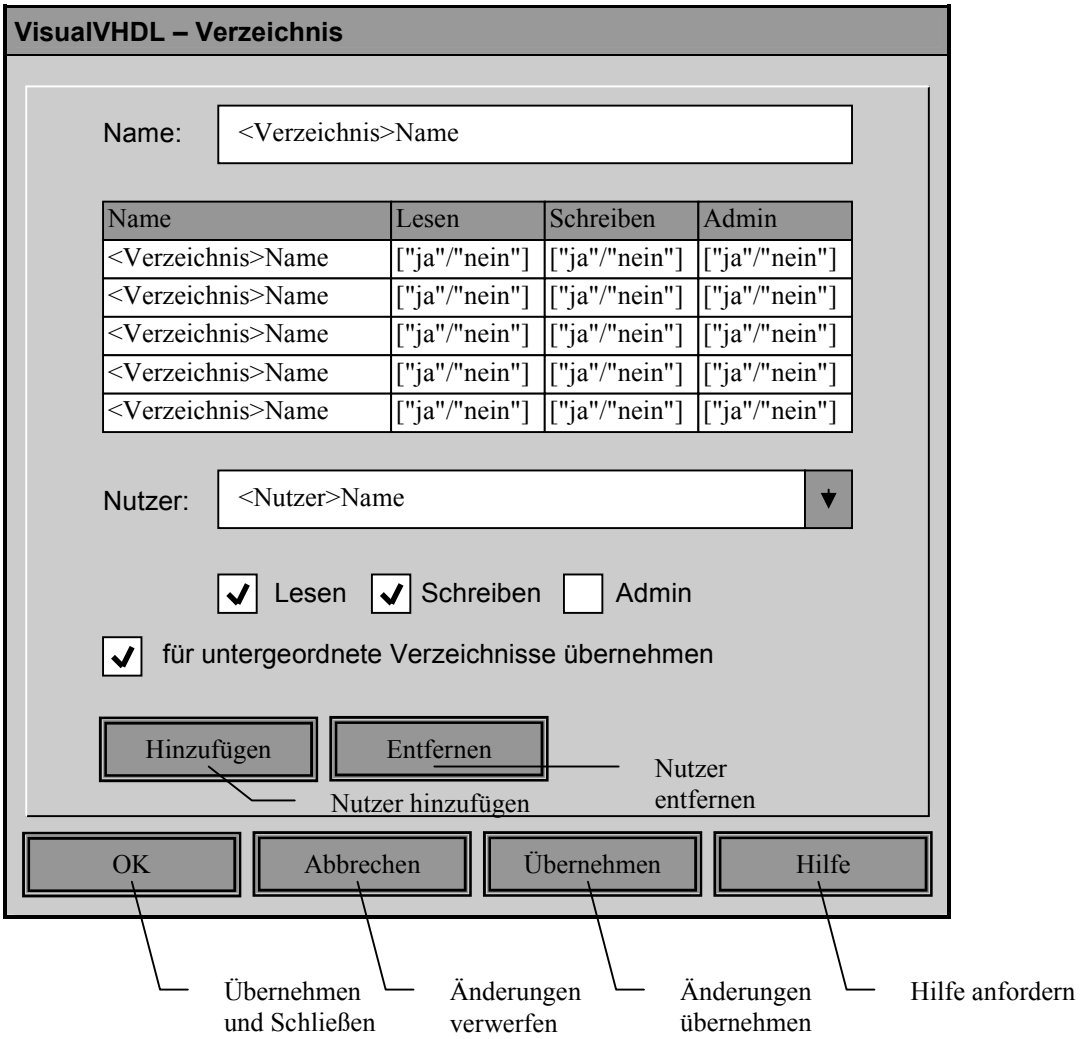
Login-Dialog:

Über diesen Dialog kann sich der Benutzer an das System anmelden. Dabei muß der Nutzername, ein Paßwort sowie der Server, auf dem der Datenkatalog (ProjektDB) liegt, angegeben werden.



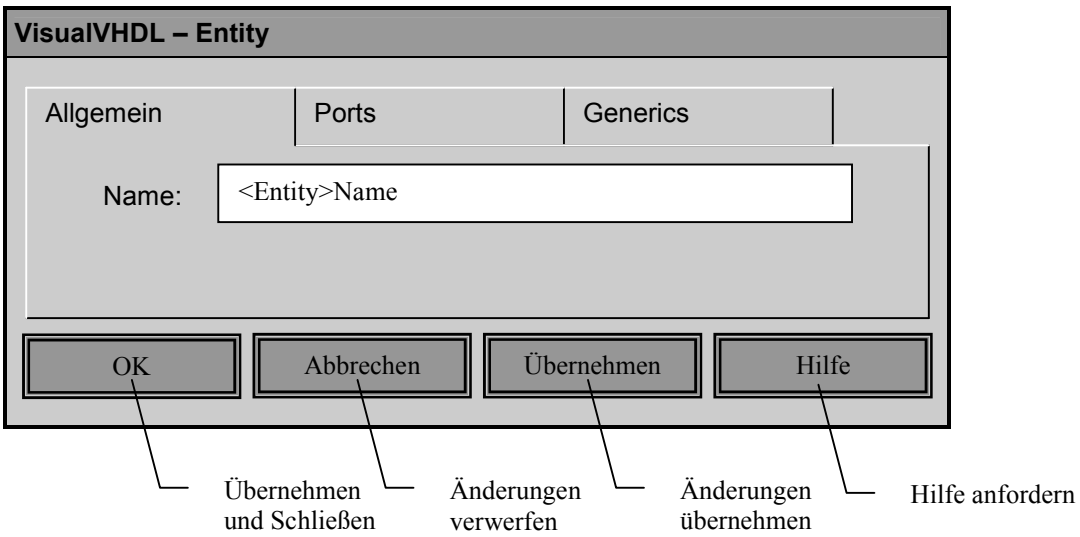
Verzeichnis-Dialog:

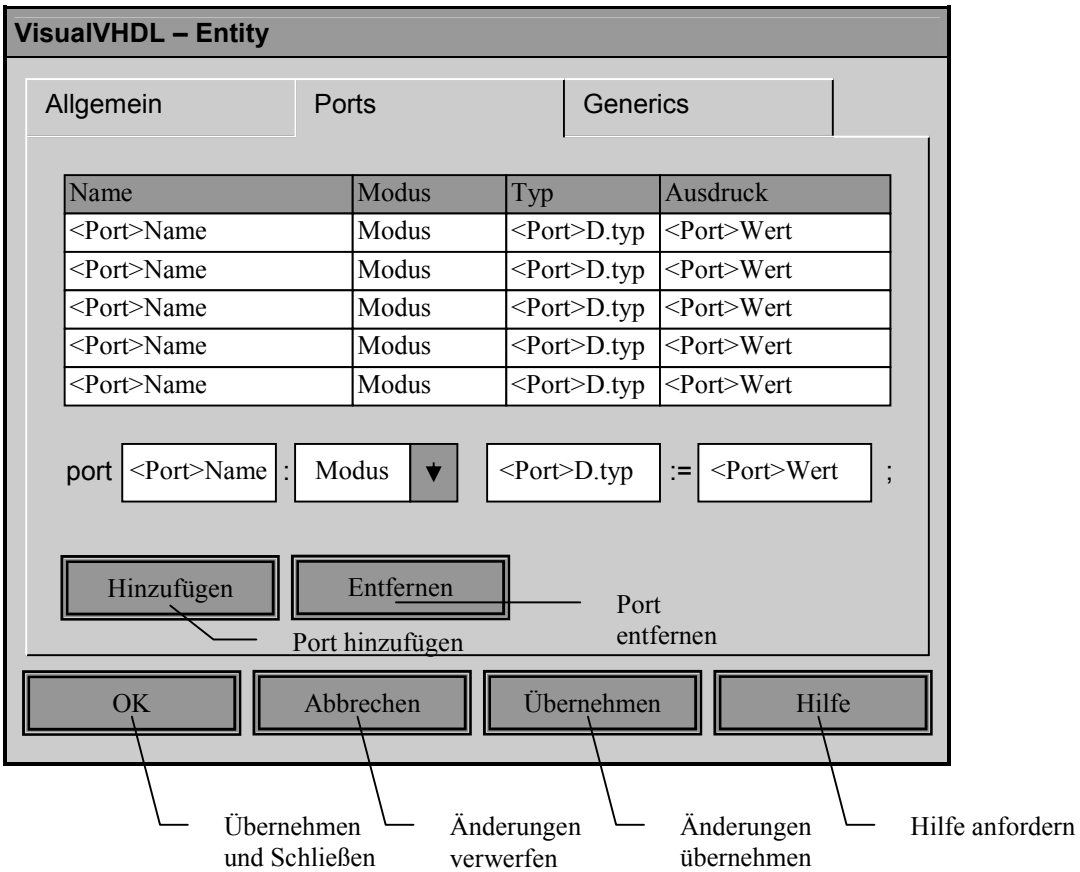
Im Verzeichnis-Dialog kann der Name des Verzeichnisses geändert werden, und die Vergabe von Nutzerrechten ist möglich.

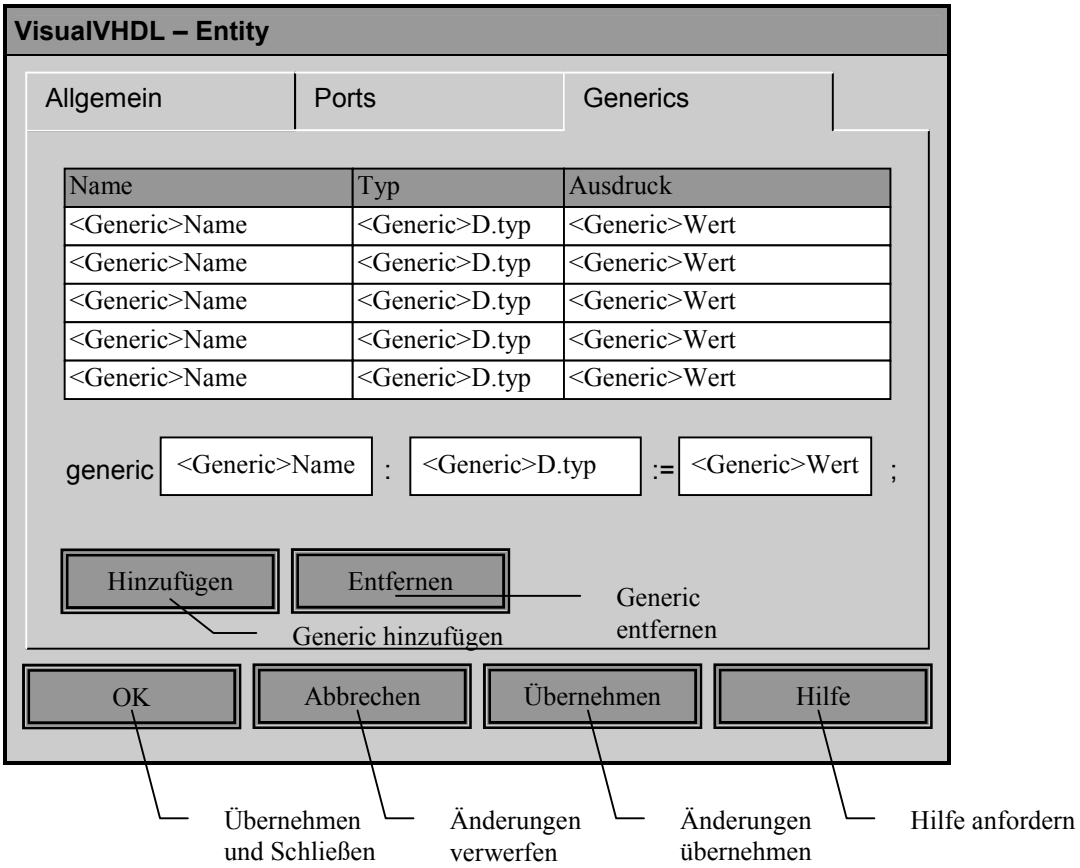


Entity-Dialog:

Im Entity-Dialog können der Name der Entity geändert sowie Ports und Generics festgelegt werden.

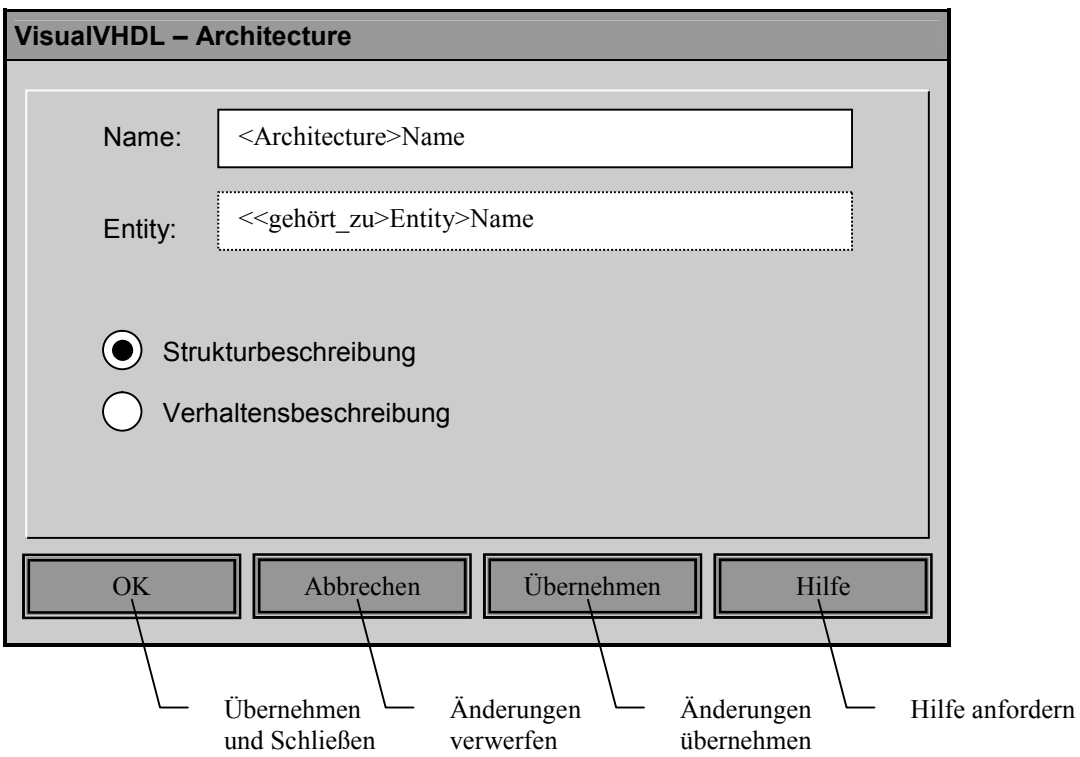






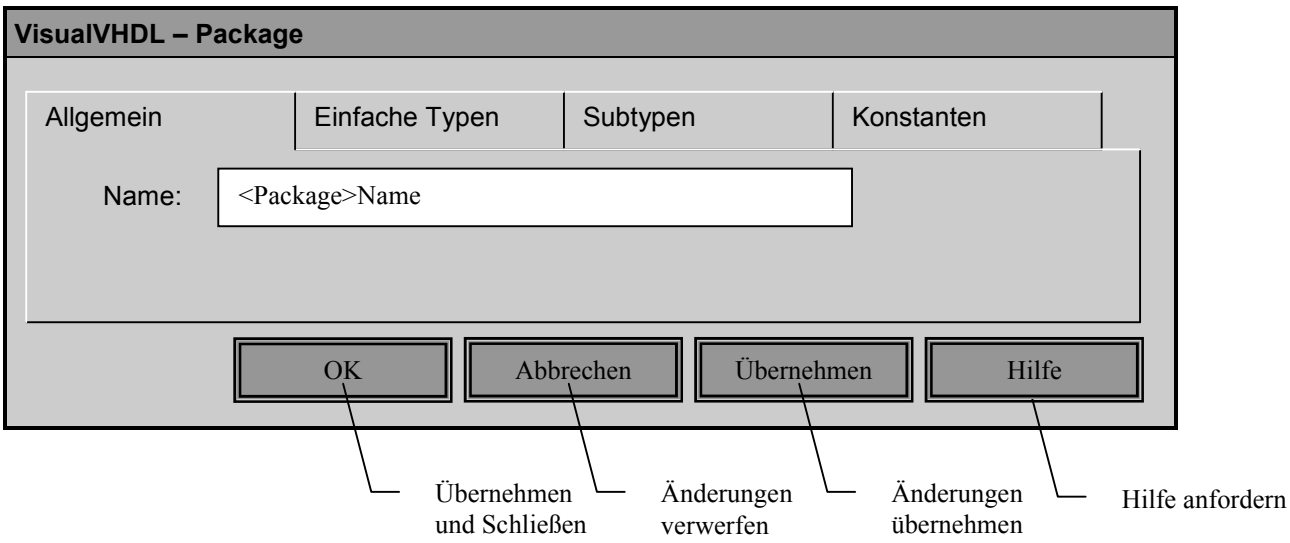
Architecture-Dialog:

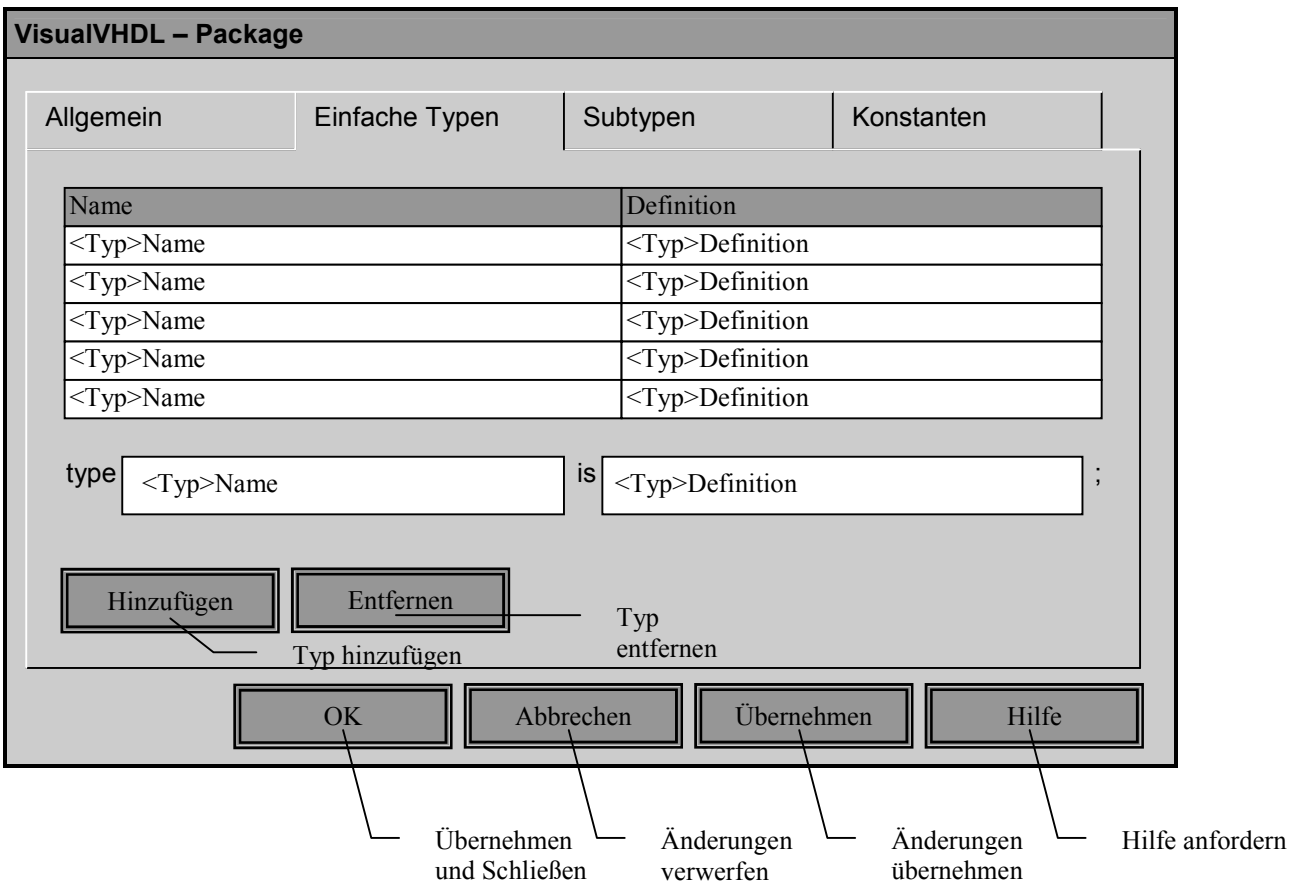
Im Architecture-Dialog kann der Name der Architecture geändert werden, und es erfolgt die Auswahl der Beschreibungsart.

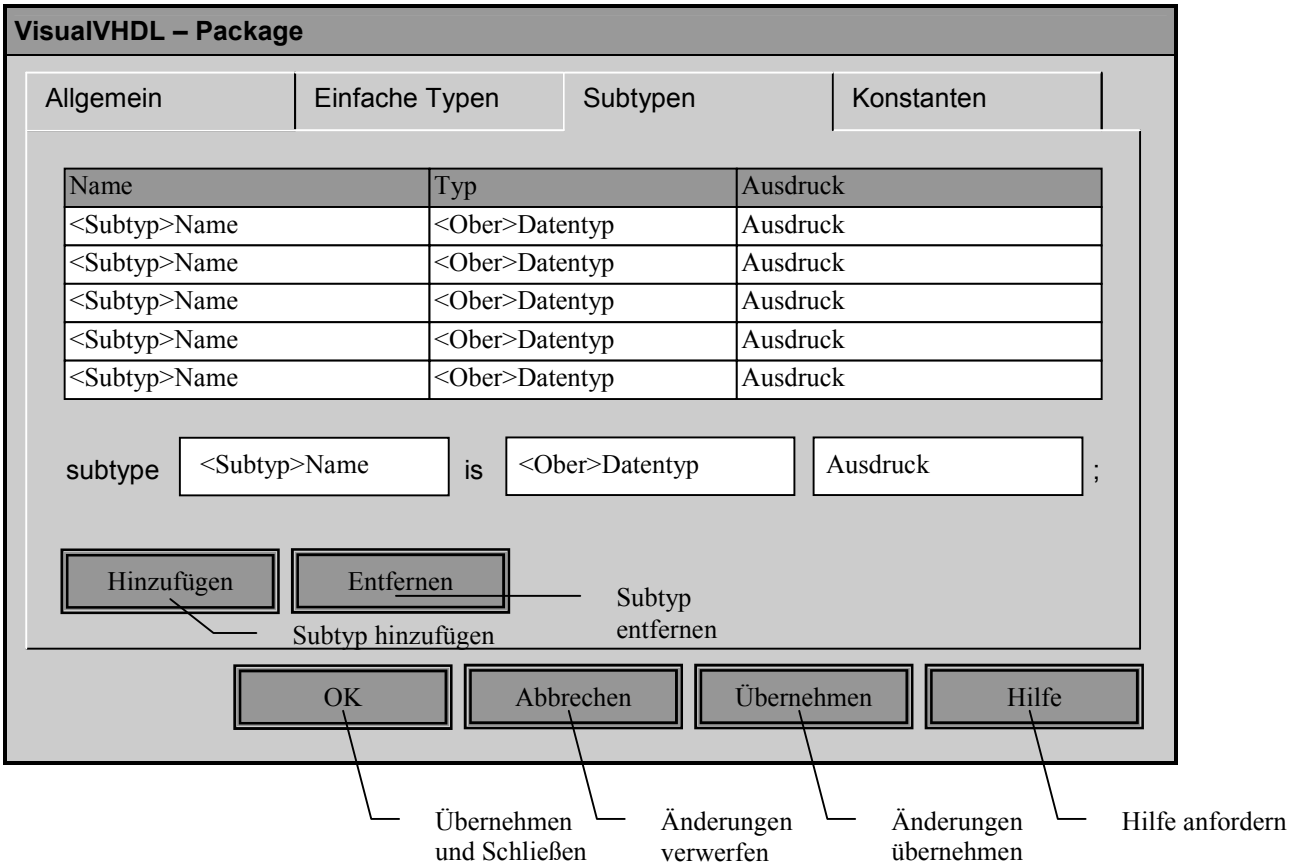


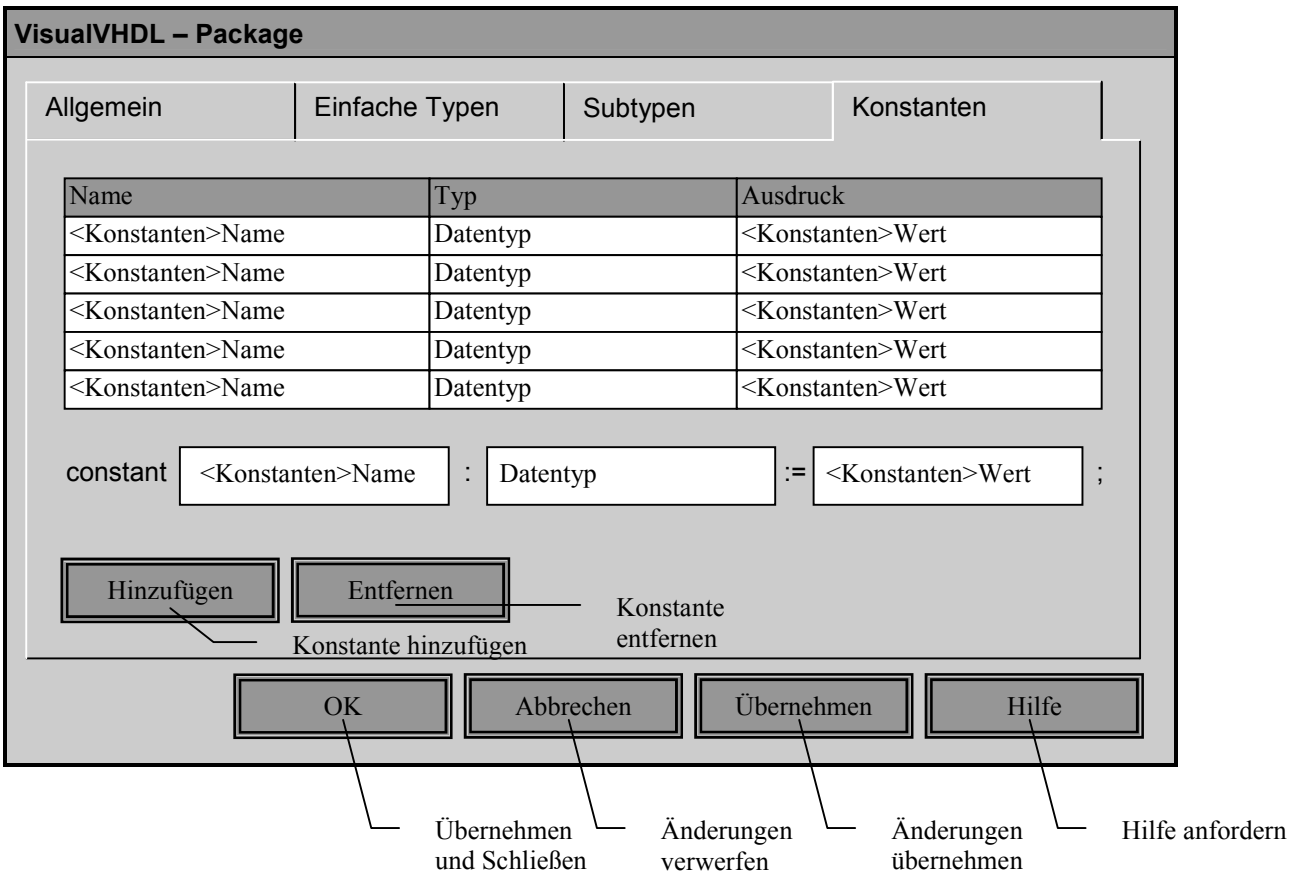
Package-Dialog:

Im Package-Dialog kann der Name geändert sowie Typen, Subtypen und Konstanten definiert werden.



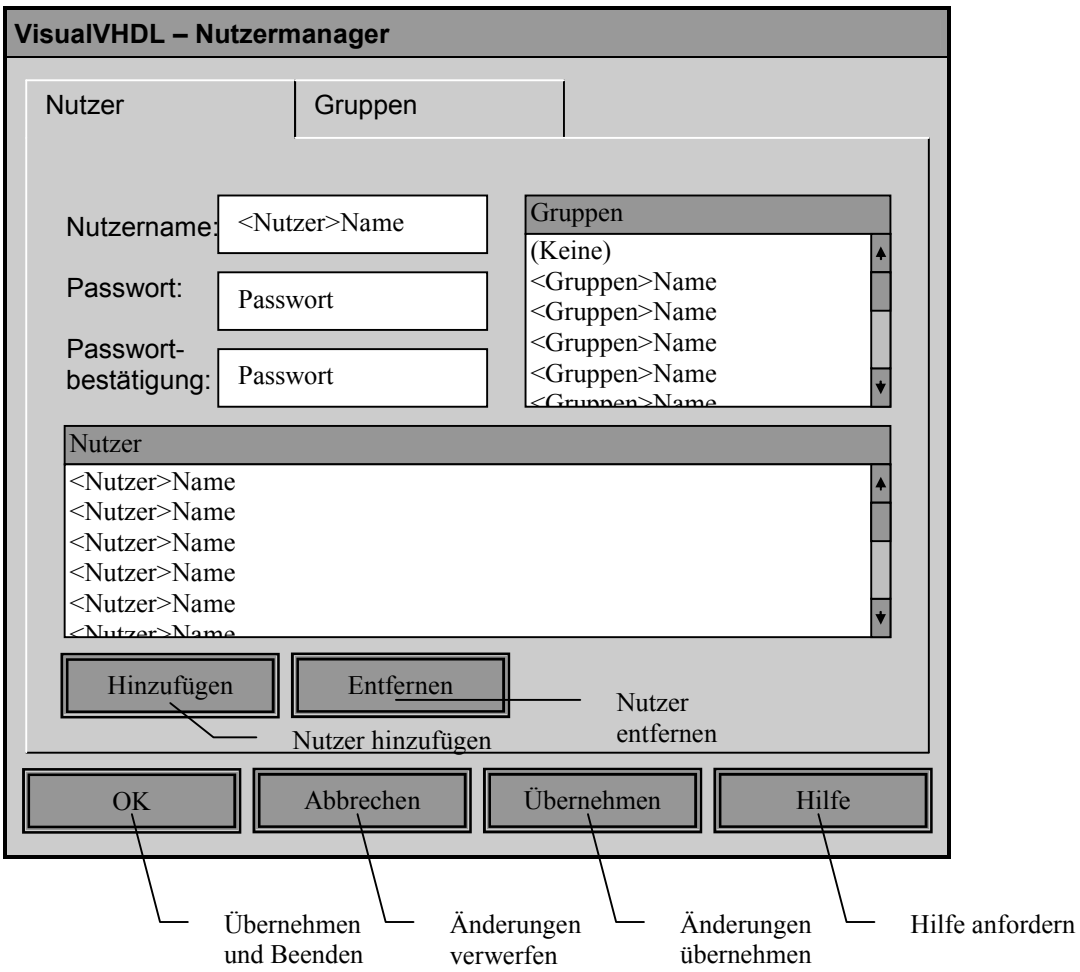


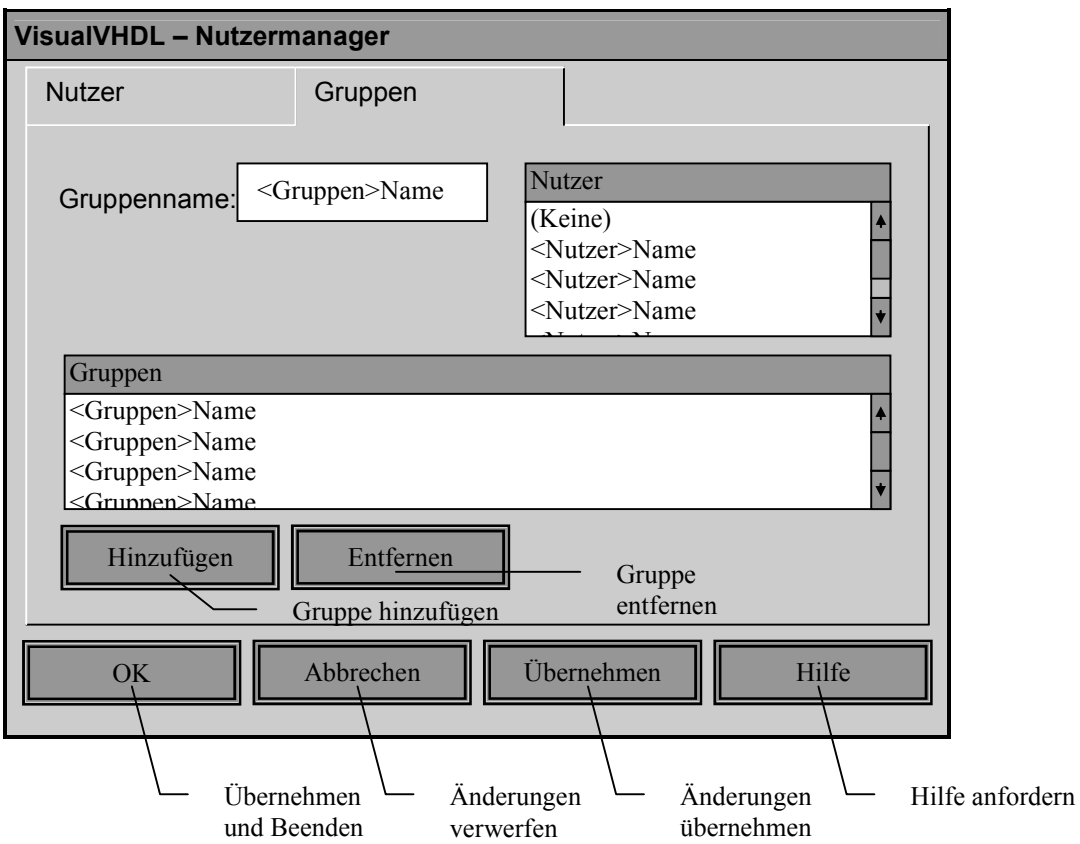

VisualVHDL – Package
Allgemein
Einfache Typen
Subtypen
Konstanten
Name
Typ
Ausdruck
<Subtyp>Name
<Ober>Datentyp
Ausdruck
subtype
is
;
Hinzufügen
Entfernen
Subtyp hinzufügen
Subtyp entfernen
OK
Abbrechen
Übernehmen
Hilfe
Übernehmen und Schließen
Änderungen verwerfen
Änderungen übernehmen
Hilfe anfordern


VisualVHDL – Package
Allgemein
Einfache Typen
Subtypen
Konstanten
Name
Typ
Ausdruck
<Konstanten>Name
Datentyp
<Konstanten>Wert
constant
:
:=
;
Hinzufügen
Entfernen
Konstante hinzufügen
Konstante entfernen
OK
Abbrechen
Übernehmen
Hilfe
Übernehmen und Schließen
Änderungen verwerfen
Änderungen übernehmen
Hilfe anfordern

Nutzermanager:

Hier kann der Administrator Nutzer und Gruppen erstellen, löschen sowie Gruppen-Zugehörigkeiten ändern.





5.5. Hilfesystem

Das Hilfesystem von VisualVHDL wurde auf der Basis von HTML entwickelt. Sämtliche Hilfethemen sind in einem HTML-Viewer, der im Programm enthalten ist, anzeigbar. Durch die Verwendung von HTML können mittels Hyperlinks ähnliche Themen schnell und einfach erreicht werden.

Die Hilfe ist natürlich kontextsensitiv, d.h. zu jedem Zustand des Systems existiert ein entsprechendes Hilfethema, so daß überall die wirklich relevanten Hilfestellungen zu einer bestimmten Aktion abrufbar sind.

Ergänzt wird dieses Hilfesystem durch eine VHDL-Kurzreferenz, wo sowohl Neueinsteiger die Möglichkeit haben, sich Grundkenntnisse in Sachen VHDL anzueignen, als auch erfahrene Benutzer z.B. die genaue Befehlssyntax nachschlagen können.