UTOPP-Dokumentation

Harald Borsutzky
Burghard Grüter
Wolfram Urich
Carsten Keßler
Christoph Pohlmann
Dominik Wilmsen
Eugen Scheinke

Greenhorn limited

Inhalt:

Kurze Produktbeschreibung

Wir wollen ein UML-Modellierungstool entwickeln, dass dem User den möglichst komfortablen Entwurf von Klassen- und Use-Case-Diagrammen ermöglichen soll.
Dabei soll das Programm über Eingabemasken die erforderlichen Angaben vom Benutzer abfragen und aus diesen Angaben dann das jeweilige Diagramm als grafisches Objekt erzeugen.

Pflichtenheft

- **Mußkriterien** Utopp soll aus Angaben des Users Klassen- und Use-Case-Diagramme erzeugen. Alle erstellten Diagramm sowie die dazugehörigen Daten sollen abgespeichert werden können.
- **Wunschkriterien** Die Benutzeroberfläche sollte möglichst User-freundlich sein. Außerdem wäre eine Speicherung im XML-Format schön, um die Daten auch außerhalb des Programms anzeigen zu können.
- **Abgrenzungskriterien** Das Programm soll keinen automatische Sourcecode-Erzeugung enthalten. Es soll keine Multi-User-Anwendung werden sondern auf einen Arbeitsplatz beschränkt bleiben.

Produkteinsatz:

- **Anwendungsbereiche** Softwarmodellierung und –entwicklung mit Hilfe der UML.
- **Zielgruppen** Mehr oder weniger erfahrene Programmierer, die mit der UML ihre Softwaremodelle entwerfen und ein Werkzeug zum schnellen „zusammenklicken“ der UML-Komponenten zur Verfügung haben wollen.

Produktumgebung:

- **Software** Java Runtime Environment
- **Hardware** PC

Produktinformationen :

Use-Cases :

0, Editieren

Akteure : Use-Case 1 bzw. 2

Ablauf : - Klick auf 'editieren'
- Eingabemaske erscheint (aktuelle Daten werden abgefragt)
- Felder : Stereotyp
Paket
Klassenname
Attribute (Datentypen, Sichtbarkeit, Initiatialwert)
Operationen (Sichtbarkeit, Datentypen)

- User füllt Felder aus klickt auf 'ok' in der Eingabemaske

Nachbedingungen : Daten werden zurückgeliefert

1, Anlegen einer Klasse

Akteure : Benutzer

Ablauf : - Benutzer klickt auf 'neue Klasse'
- Use-Case 0 wird gestartet
- Falls Daten korrekt wird neues Objekt angelegt

Nachbedingung : Neues Objekt wird angelegt

Vorbedingung : Objekt noch nicht vorhanden

2, Klasse editieren

Akteure : Benutzer

Ablauf : - Benutzer klickt auf 'editiere Klasse'
- Use-Case 0 erhält vorhandene Daten
- Benutzer ändert Daten
- Use-Case 0 gibt Daten zurück
- Falls Daten korrekt wird Objekt geändert und gespeichert

Nachbedingung : Objekt ist geändert

Vorbedingung : Klasse vorhanden

3, Klasse löschen

Akteure: Benutzer

Ablauf: - Benutzer klickt auf „Klasse löschen“

- - Sicherheitsabfrage erscheint: ‚wirklich löschen?’
- - Wenn ‚Ja’, Klasse löschen und alle Assoziationen, mit denen die Klasse zusammenhängt, löschen (Aufruf von Use-Case 7); sonst Abbruch des Use-Cases

Nachbedingung: Klasse ist gelöscht

4, Assoziationsdaten erfassen

Akteur: Use-Cases 5 und 6

Ablauf: Eingabemaske erscheint, zur Auswahl: Assoziation zwischen 2 oder mehr als 2 Klassen

Eingabemaske für Assoziation zwischen 2 Klassen:

- Klasse 1
- Rolle Klasse 1
- Multiplizität Klasse 1
- Klasse 2
- Rolle Klasse 2
- Multiplizität Klasse 2
- Assoziationstyp: zur Auswahl stehen:
  - Vererbung
  - Gerichtete Assoziation
  - Abgeleitete Assoziation
  - Qualifizierte Assoziation
  - Geordnete Assoziation
  - Verfeinerung
  - Abhängigkeit
  - Komposition
  - Aggregation
  - Mehrgliedrige Assoziation

Beschriftung der Assoziation

Eingabemaske für Assoziationen zwischen mehr als 2 Klassen:

- Klassenliste wird angezeigt
- Button ‚hinzufügen', um Klassen aus der Liste in die Assoziation aufzunehmen

- Für jede hinzugefügte Klasse werden Rolle und Multiplizität erfragt

Nachbedingung: Daten werden zurückgeliefert

5, Anlegen einer Assoziation

Akteur: Benutzer

Ablauf:

- Benutzer klickt auf „neue Assoziation“
- Use-case 4 wird aufgerufen
- Falls die Daten korrekt sind, wird ein neues Objekt angelegt

Vorbedingungen: Es ist noch keine Assoziation zwischen den beteiligten Klassen vorhanden

Nachbedingungen: Assoziation wurde angelegt

6, Assoziation editieren

Akteur: Benutzer

Ablauf:

- Benutzer klickt auf „Assoziation editieren“
- Use-Case 4 wird aufgerufen, dabei wird die Änderung der beteiligten Klassen deaktiviert (hierzu muss die Assoziation gelöscht werden und eine neue angelegt werden)

Nachbedingungen: Assoziation wurde mit Änderungen gespeichert

7, Assoziation löschen

Akteur: Benutzer oder Use-Case 3

Ablauf:

- Benutzer klickt auf „Assoziation löschen“
- Sicherheitsabfrage erscheint: ‚Wirklich löschen?’
- Wenn ‚Ja’ wird die Assoziation gelöscht, sonst Abbruch des Use-Cases

Nachbedingung: Assoziation gelöscht

8, Use-Case Daten erfassen

Akteur: Use Case 9 bzw. 10

Ablauf: Eingabemaske erscheint mit folgenden Eingabefeldern:

- Name
- Akteur
- Ablauf
- Vorbedingungen
- Nachbedingungen
- Invarianten
- Fehler
- Dialogbeispiel
- Nicht-funktionale Anwendungsanforderungen
- Variationen
- Ansprechpartner
- Regeln
- Services

- [Wunsch: Beziehungen zu anderen Use-Cases definieren]

Nach Ausfüllen der gewünschten Felder (hier müssen nicht alle Felder ausgefüllt werden, nur der Name muss eingegeben werden) schließt der Benutzer die Eingabe mit Klick auf „Okay“ ab.

Nachbedingungen: Daten wurden zurückgegeben

9, Use-Case anlegen

Akteur: Benutzer

Ablauf:

- Benutzer klickt auf „neuer Use-Case“
- Use-Case 8 wird aufgerufen
- Neuer Use-Case wird mit den übergebenen Daten angelegt und gespeichert

Nachbedingung: Use-Case ist gespeichert

10, Use-Case ändern

Akteur: Benutzer

Ablauf:

- Benutzer klickt auf „Use-Case ändern“
- Use-Case 8 wird aufgerufen, dabei werden die aktuellen Angaben zum Use-Case angezeigt
- Benutzer ändert die Angaben seinen Wünschen gemäss
- Use-Case wird mit den geänderten, von Use-Case 8 zurückgegebenen Daten gespeichert

Vorbedingung: Zu bearbeitender Use-Case exisitiert bereits

Nachbedingungen: geänderter Use-Case wird abgespeichert

11, Use-Case löschen

Akteur: Benutzer

Ablauf:

- Benutzer klickt „Use-Case löschen“
- Sicherheitsabfrage erscheint: ‚wirklich löschen?’
- Bei ‚Ja’ wird der Use-Case gelöscht, sonst Abbruch

Vorbedingung: Use-Case existiert bereits

Nachbedingung: Use-Case ist gelöscht

12, Pflichtenheftdaten editieren

Akteur: Benutzer, Use-Case 13

Ablauf: Eingabemaske erscheint, in der entweder die aktuellen Daten oder (beim Anlegen eines neuen Projekts) Default-Werte angezeigt werden:

- Zielbestimmung
- Produkteinsatz
- Produktumgebung
- Produktleistungen
- Qualitätsziele
- Testfälle / -szenarien
- Use-Cases (nur namentlich anzugeben)
- Entwicklungsumgebung
- Ergänzungen
- [Wunsch: GUI-Entwurf]

Dabei werden für alle hier aufgeführten Punkte (und deren Unterpunkte nach dem Pflichtenheft-Schema wie auf der Website zum ProPra – hier der Einfachheit halber weggelassen) Textfelder zur Eingabe aufgeführt. Ein neues Pflichtenheft wird automatisch angelegt, sobald der Benutzer ein neues Projekt anlegt, daher haben wir für diese Aktion keinen eigenen Use-Case modelliert.

Nachbedingung: geändertes Pflichtenheft ist gespeichert

13, neues Projekt anlegen

Akteur: Benutzer

Ablauf:

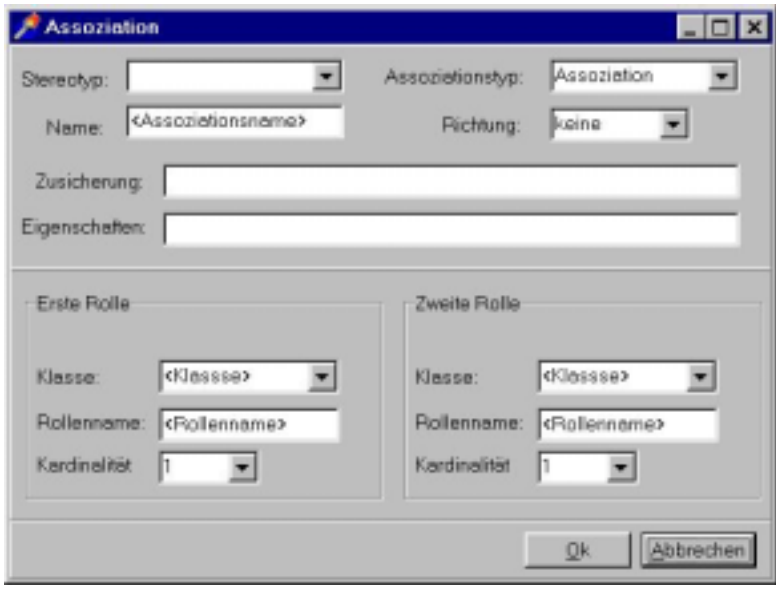
- Benutzer klickt auf „neues Projekt starten“
- Eventuell noch geöffnetes Projekt wird gespeichert und geschlossen (es kann also nur 1 Projekt zur gleichen Zeit geöffnet sein)
- Der Name für das neue Projekt wird erfragt
- Hat der Benutzer die Namenseingabe beendet, wird geprüft, ob es schon ein Projekt mit gleichem Namen gibt.
- Falls ja, wird der Benutzer zur Änderung aufgefordert
- Sonst: Pflichtenheft wird angelegt und Use-Case 12 wird aufgerufen
- Manager für Klassenliste, Use-Case Liste und Assoziationsliste werden angelegt

Nachbedingung: Projekt ist angelegt

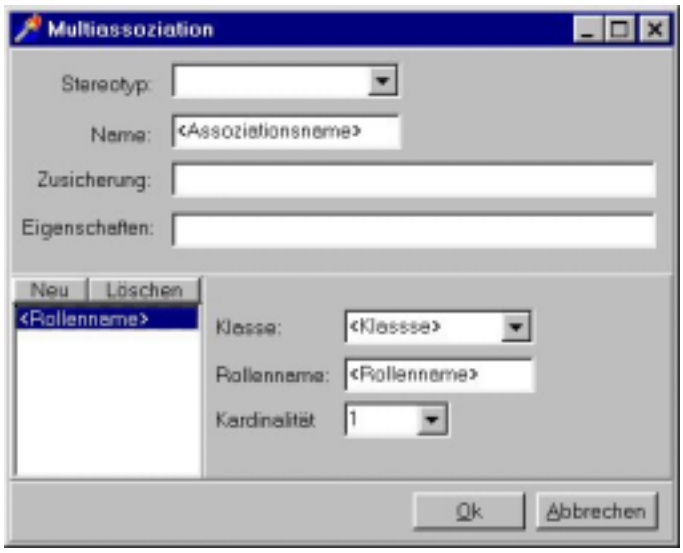
Entwürfe für die Eingabedialoge

Diese Dialoge haben wir vor dem implementieren entworfen und dann versucht, sie möglichst gut in Java nachzubauen.

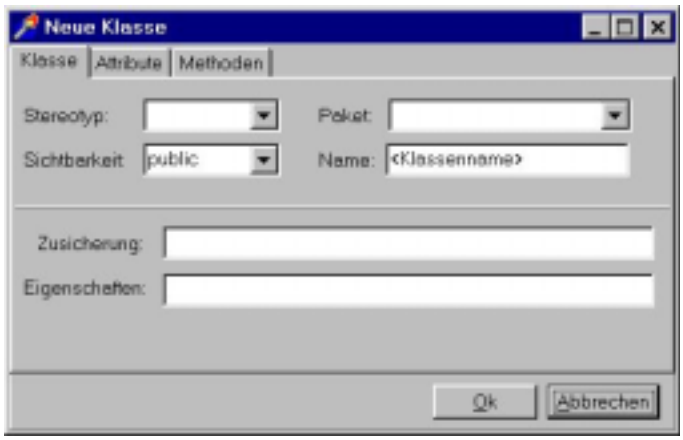
Eingabedialog für Assoziationen:



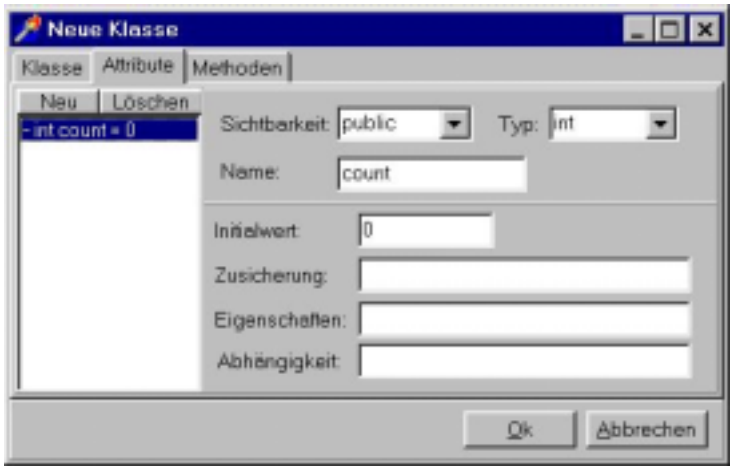
Eingabedialog für Assoziationen mit mehr als 2 Beteiligten:



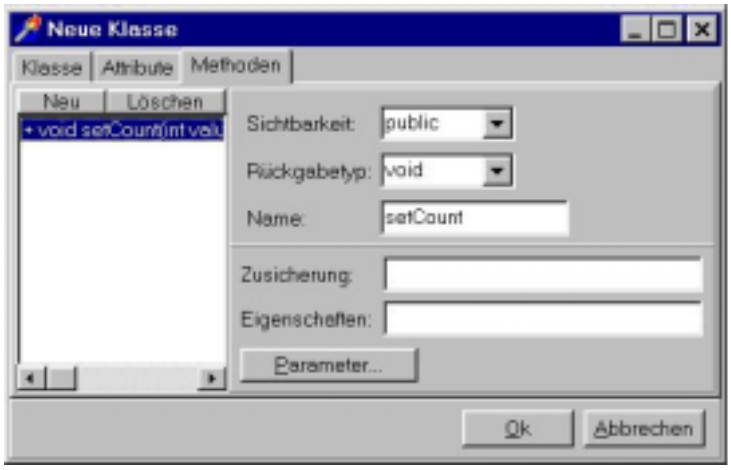
Eingabedialog zum anlegen einer neuen Klasse:



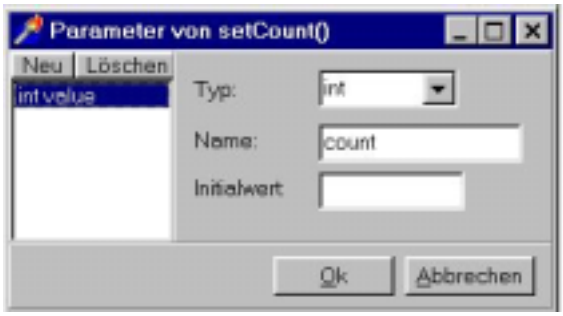
Eingabedialog zum anlegen der zur Klasse gehörenden Attribute:



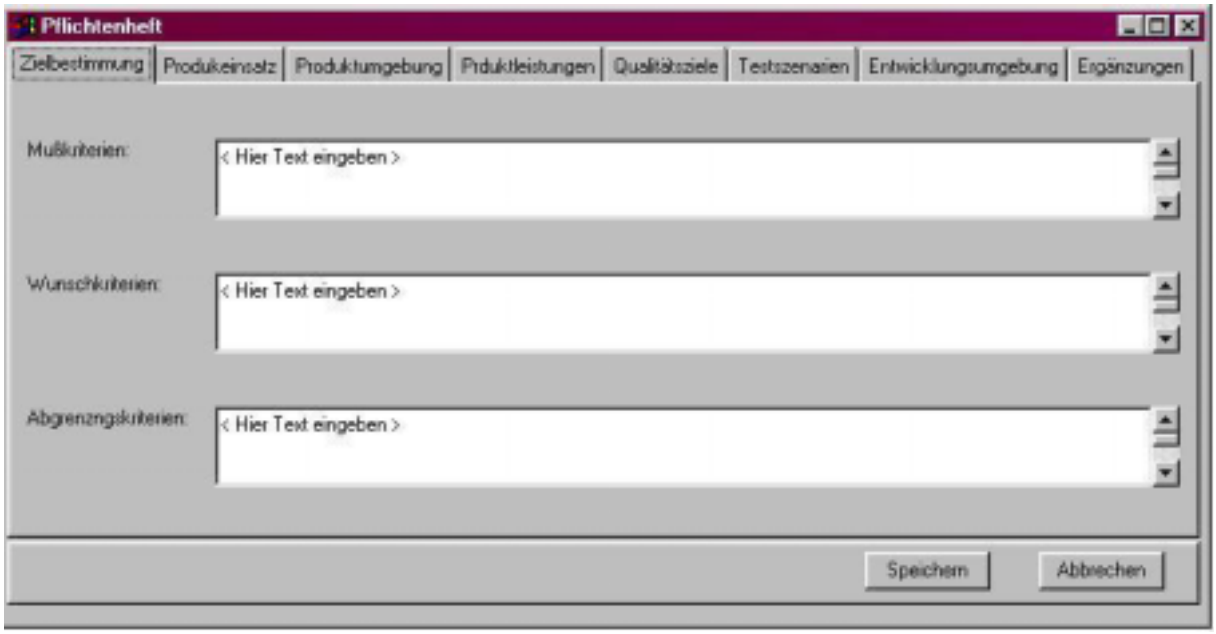
Eingabedialog zum anlegen der zur Klasse gehörenden Methoden:



Eingabedialog zum anlegen der zur Methode gehörenden Parameter:



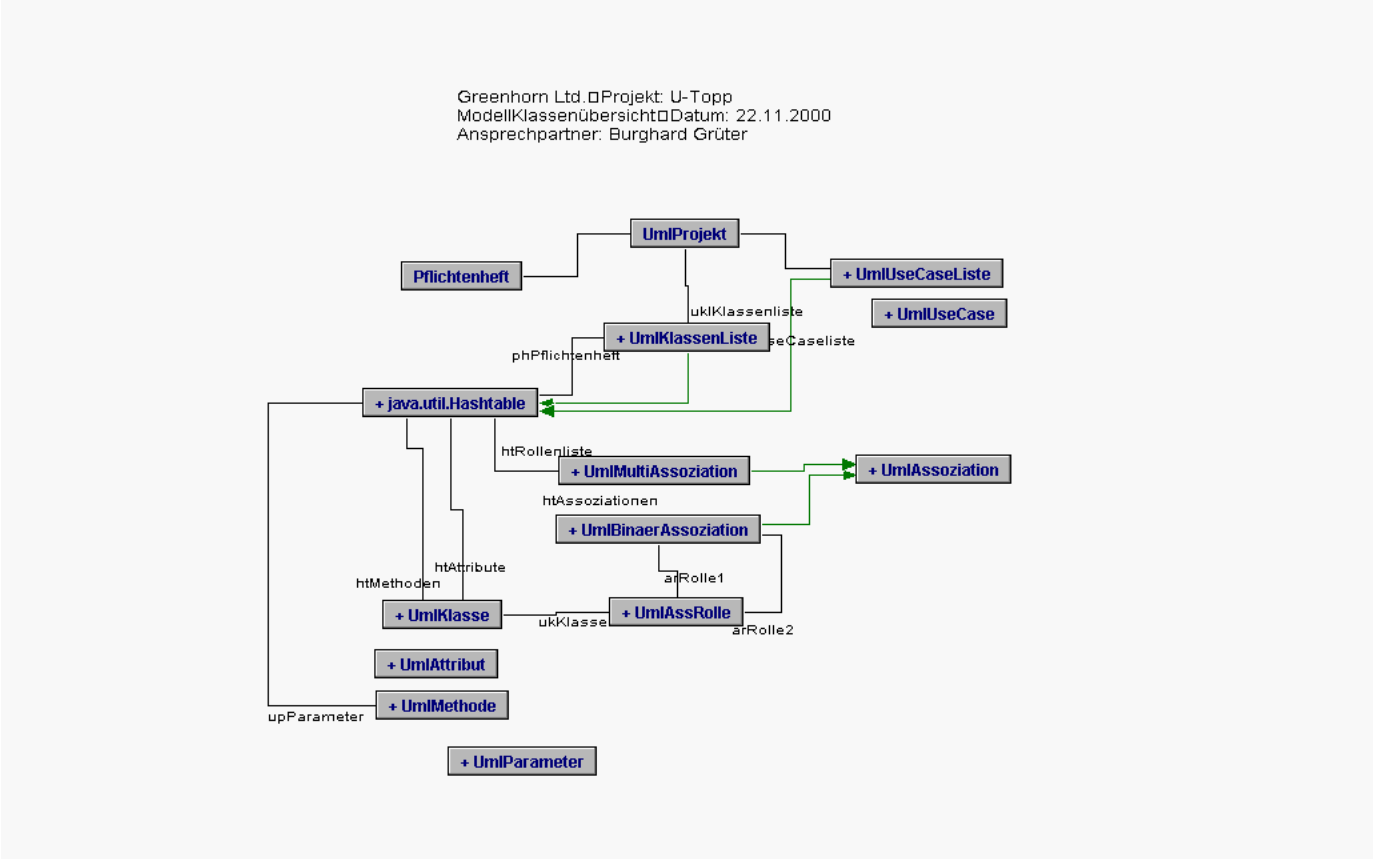
Eingabedialog für das Pflichtenheft:

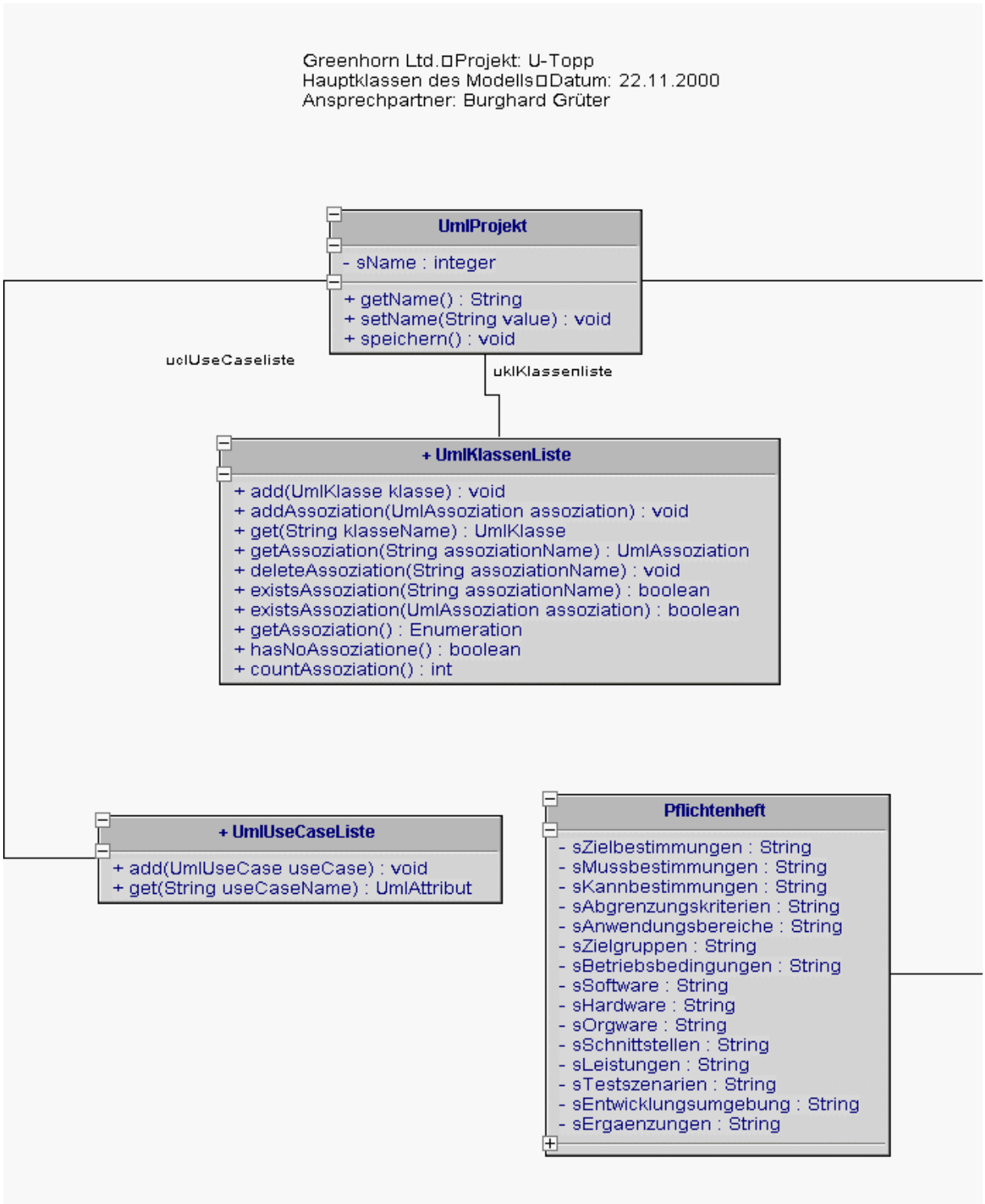


Eingabedialog für das Pflichtenheft (2):

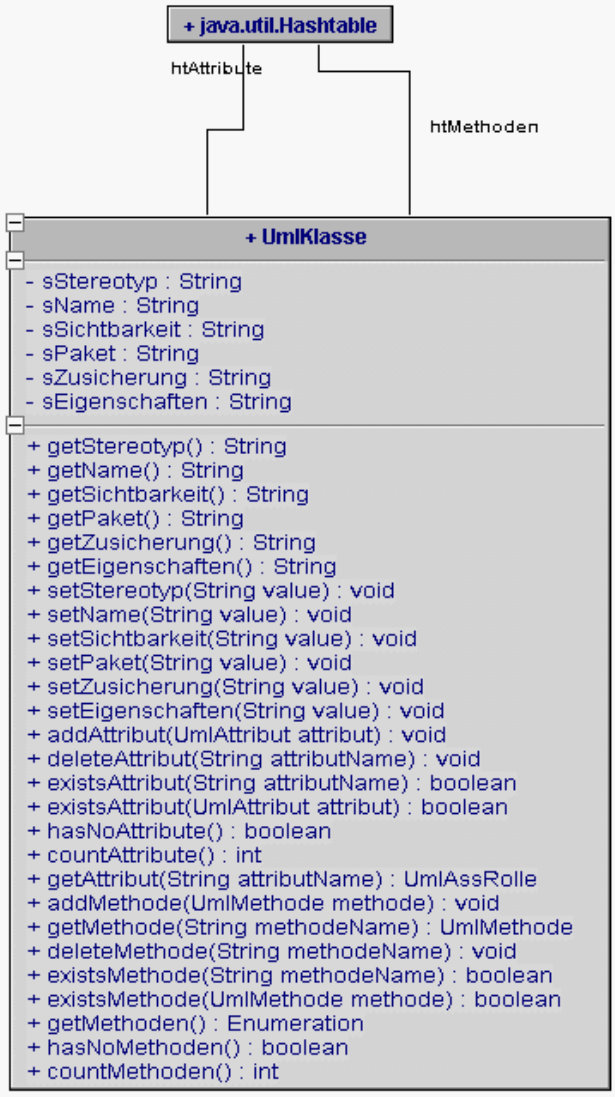
Klassendiagramme

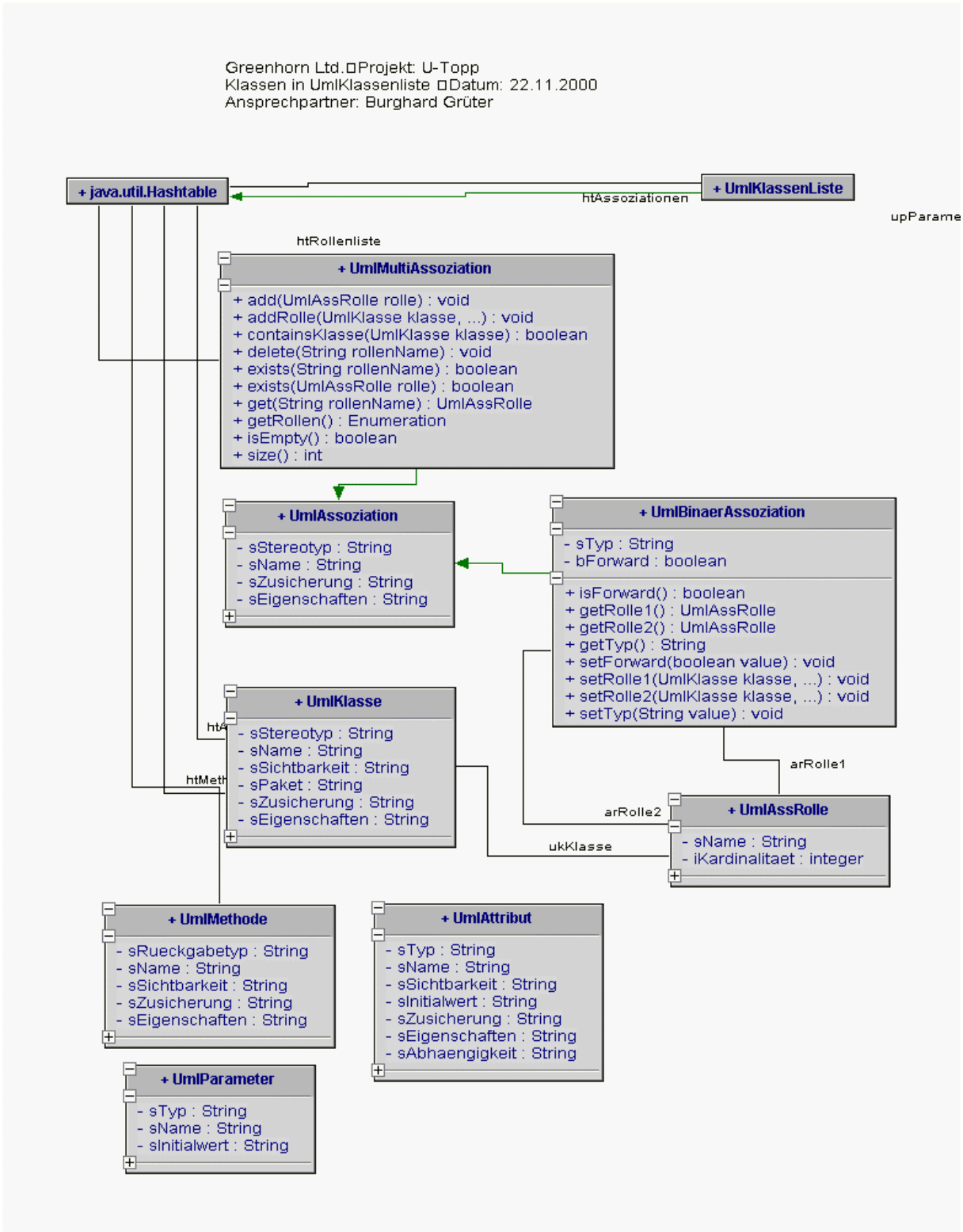
Zunächst die Klassendiagramme, wie wir sie vor dem Implementieren entworfen hatten.



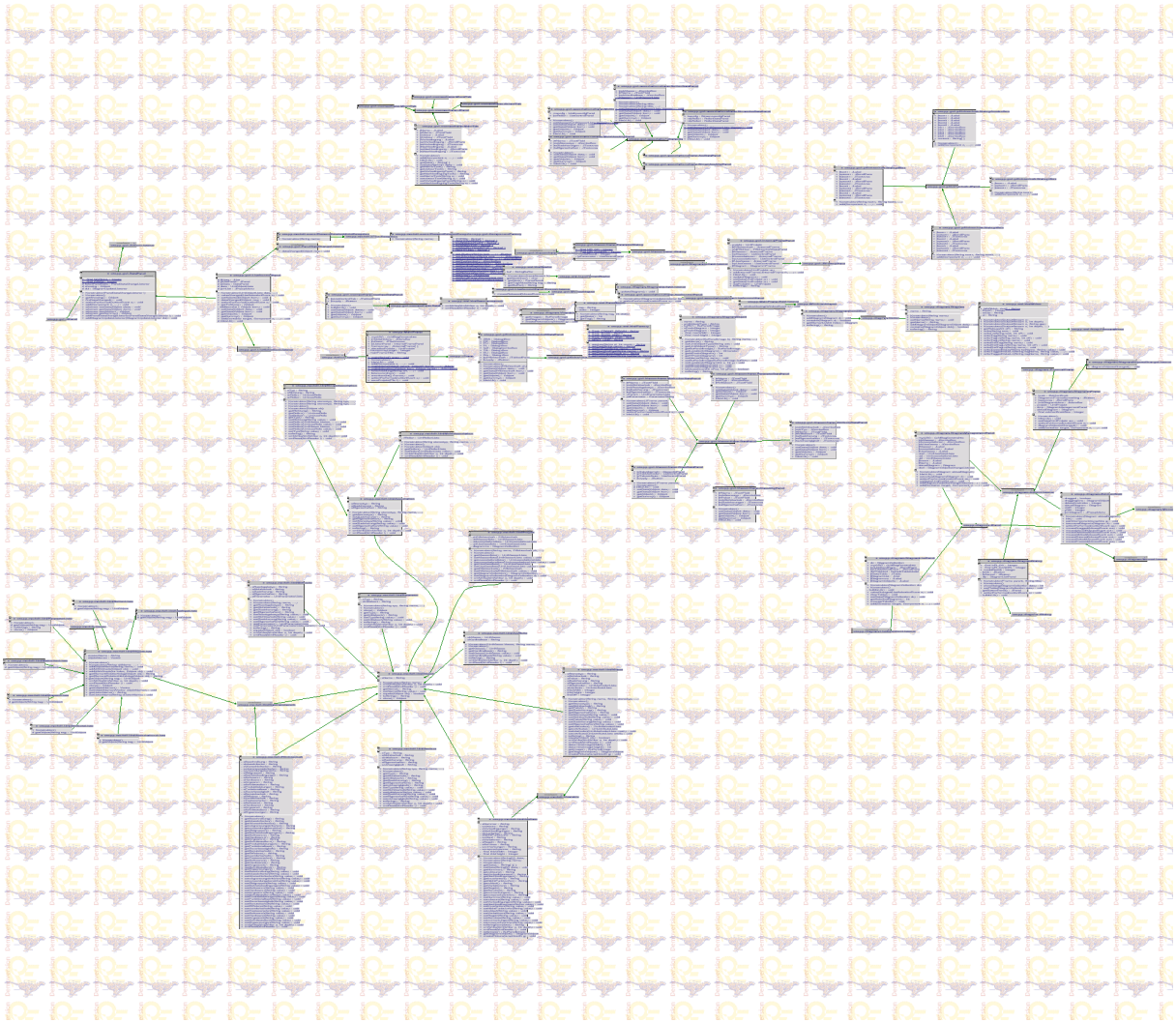

Greenhorn Ltd.□Projekt: U-Topp
Hauptklassen des Modells□Datum: 22.11.2000
Ansprechpartner: Burghard Grüter
UmlProjekt
- sName : integer
+ getName() : String
+ setName(String value) : void
+ speichern() : void
uclUseCaseliste
uklKlassenliste
+ UmlKlassenListe
+ add(UmlKlasse klasse) : void
+ addAssoziation(UmlAssoziation assoziation) : void
+ get(String klasseName) : UmlKlasse
+ getAssoziation(String assoziationName) : UmlAssoziation
+ deleteAssoziation(String assoziationName) : void
+ existsAssoziation(String assoziationName) : boolean
+ existsAssoziation(UmlAssoziation assoziation) : boolean
+ getAssoziation() : Enumeration
+ hasNoAssoziatione() : boolean
+ countAssoziation() : int
+ UmlUseCaseListe
+ add(UmlUseCase useCase) : void
+ get(String useCaseName) : UmlAttribut
Pflichtenheft
- sZielbestimmungen : String
- sMussbestimmungen : String
- sKannbestimmungen : String
- sAbgrenzungskriterien : String
- sAnwendungsbereiche : String
- sZielgruppen : String
- sBetriebsbedingungen : String
- sSoftware : String
- sHardware : String
- sOrgware : String
- sSchnittstellen : String
- sLeistungen : String
- sTestszenarien : String
- sEntwicklungsumgebung : String
- sErgaenzungen : String

Greenhorn Ltd.□Projekt: U-Topp
Klasse UmlKlasse □Datum: 22.11.2000
Ansprechpartner: Burghard Grüter

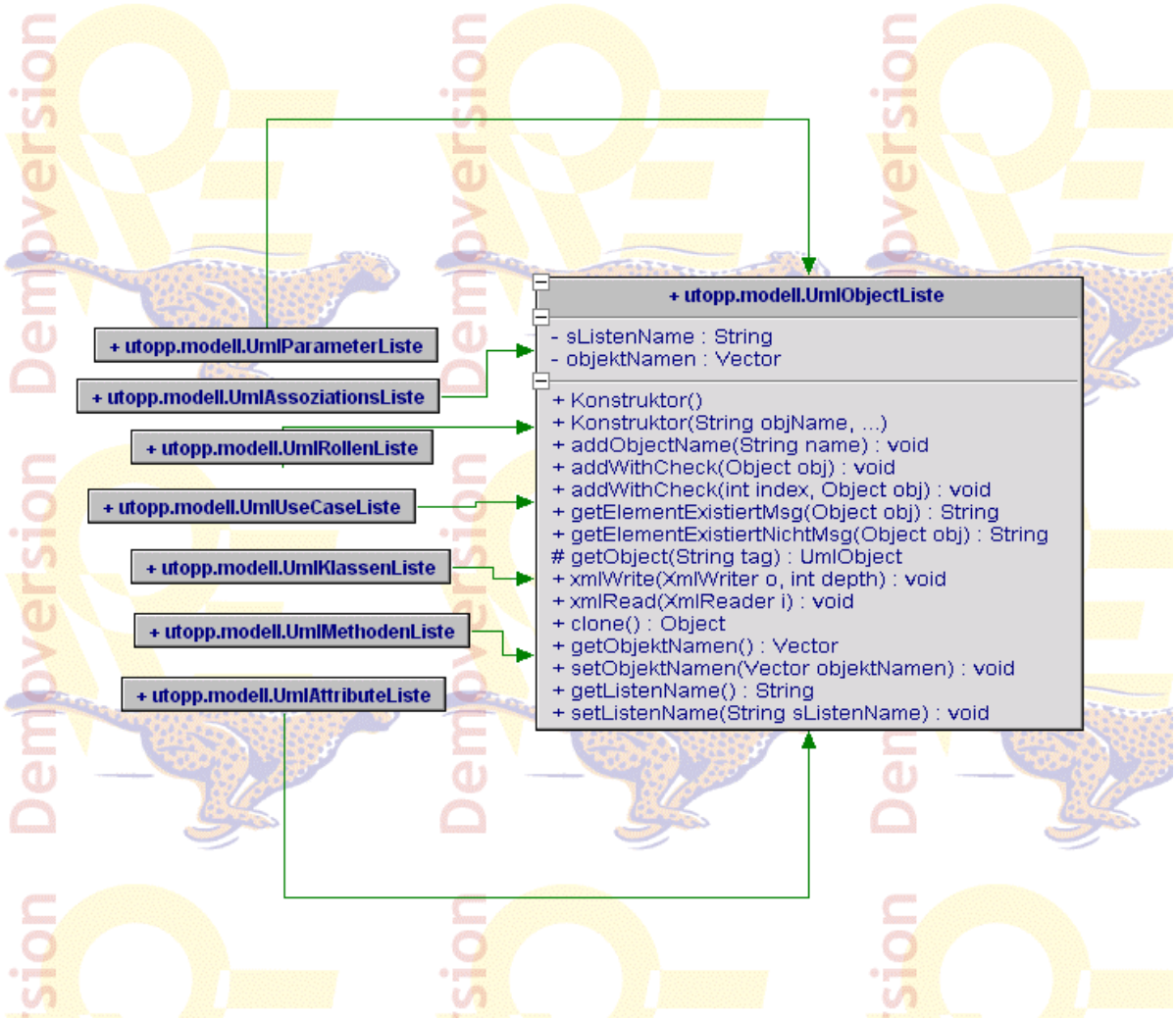




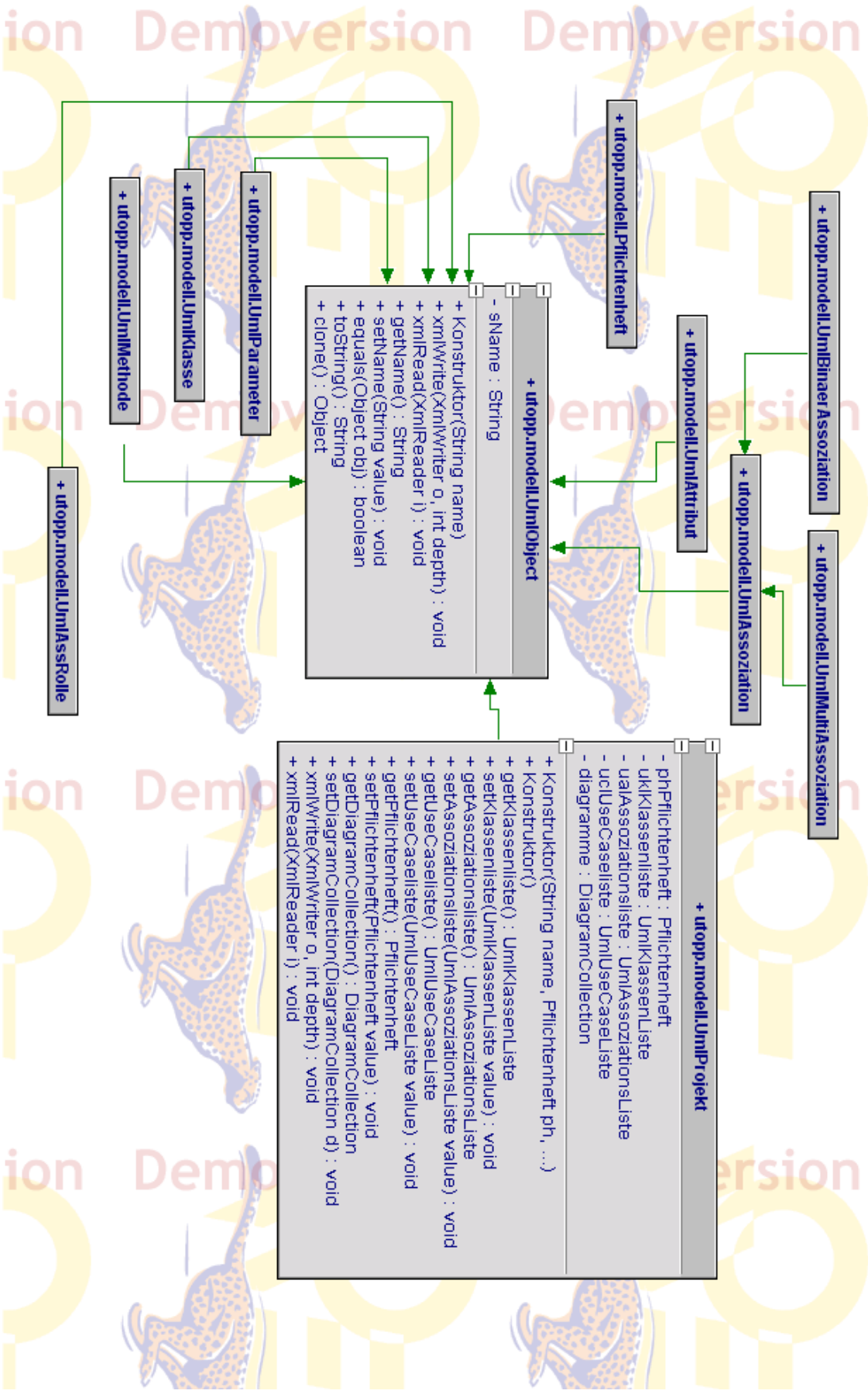
Da wir jedoch beim implemtieren festgestellt haben, daß wir mit diesem Modell nicht sehr weit kommen werden, haben wir es bei Bedarf (also ziemlich häufig) erweitert , so daß schließlich das unsere letzte Version des Programms auf dem folgenden Modell basiert:



Da in dieser Gesamtübersicht nicht viel zu erkennen ist, sind nachfolgend nochmal die zentralen Ausschnitte unseres Modells einzeln dargestellt:

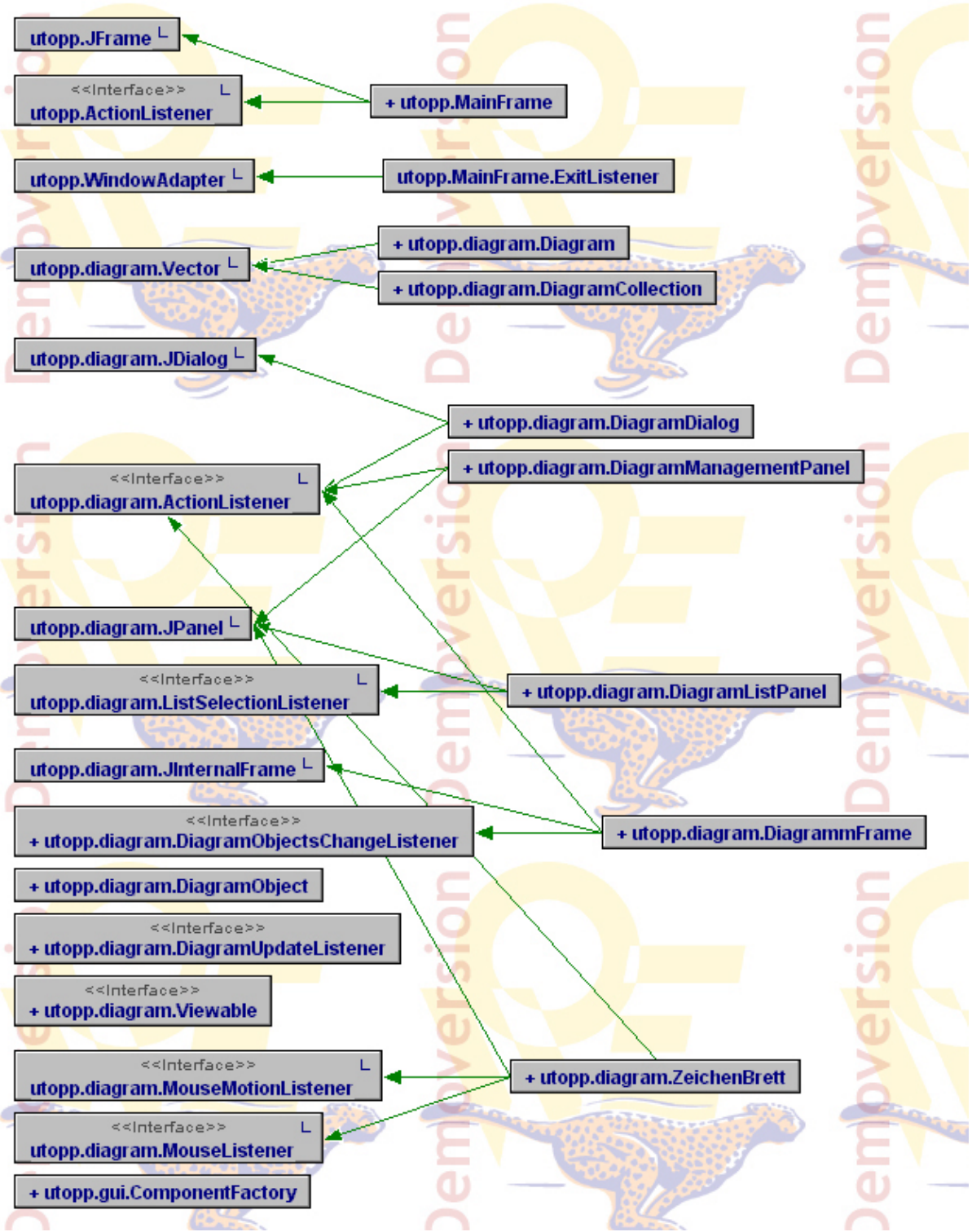


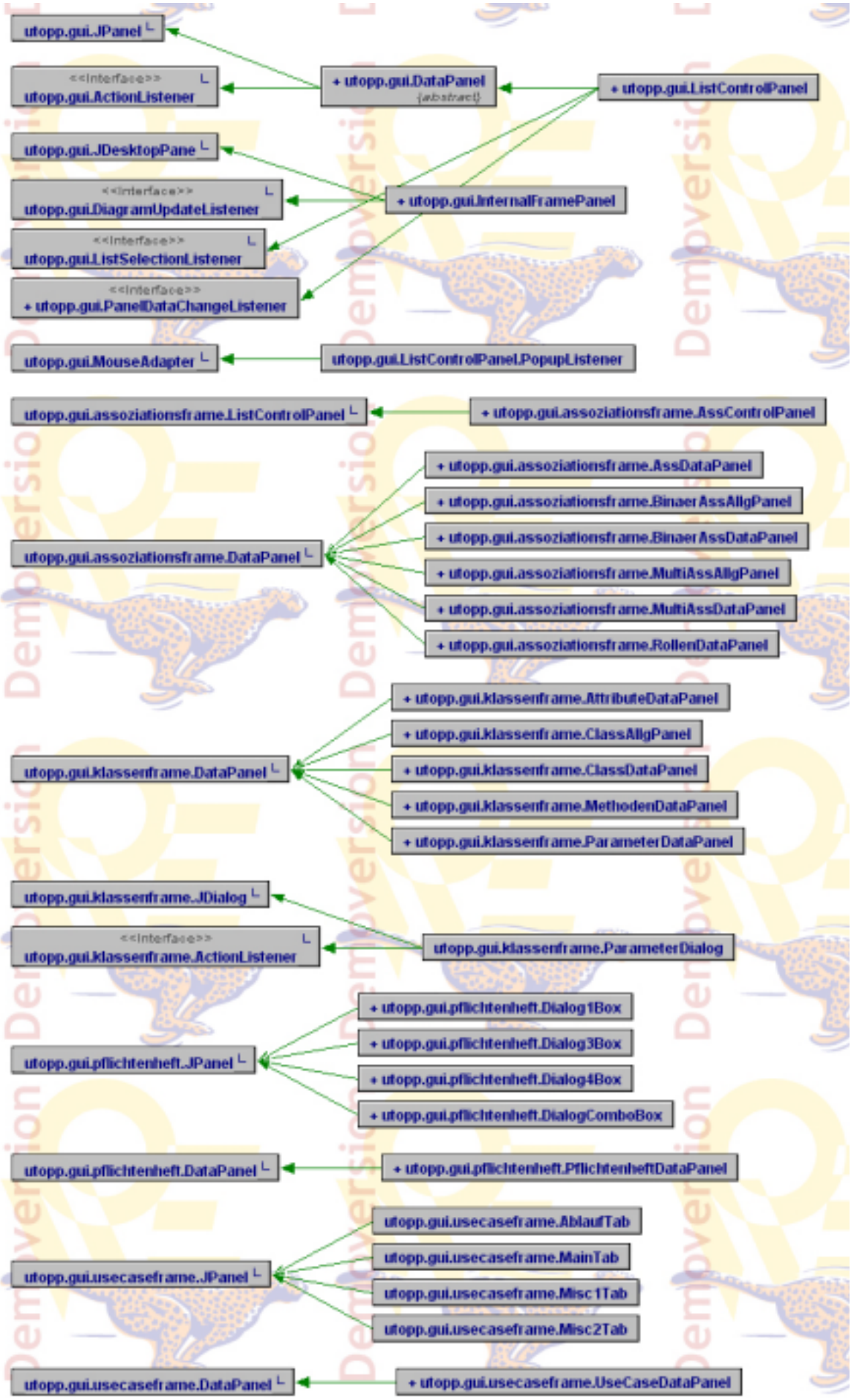
Sämtliche Objekte werden in unserem Projekt in Listen gespeichert, die alle von der UmlObjectListe erben.

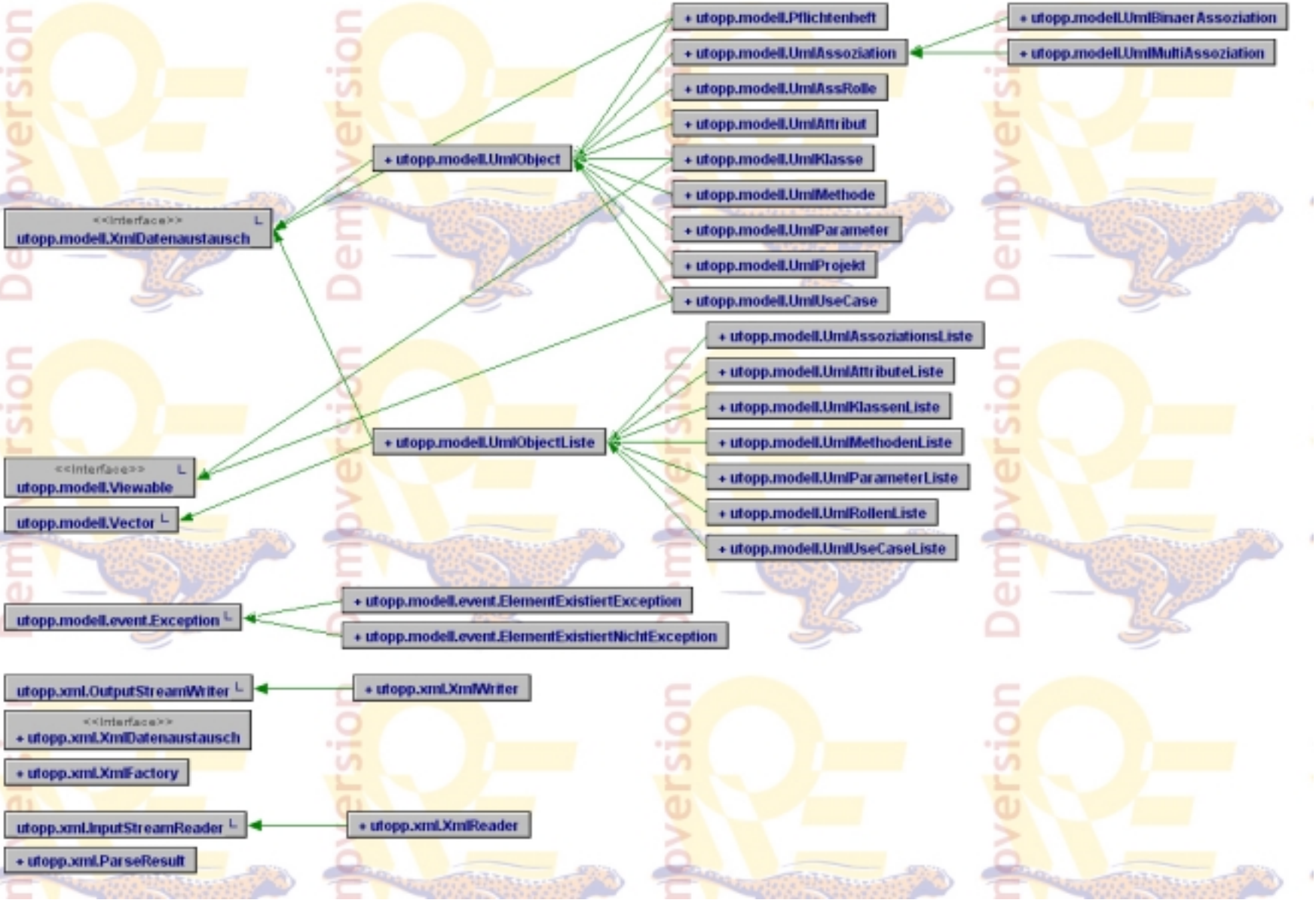


Alle Elemente der Uml sowie unser UmlProject sind von einer Klasse „UmlObject“ abgeleitet.

Verbungsdiagramme




utopp.gui.JPanel
<<Interface>>
utopp.gui.ActionListener
+ utopp.gui.DataPanel
{abstract}
+ utopp.gui.ListControlPanel
utopp.gui.JDesktopPane
<<Interface>>
utopp.gui.DiagramUpdateListener
+ utopp.gui.InternalFramePanel
<<Interface>>
utopp.gui.ListSelectionListener
<<Interface>>
+ utopp.gui.PanelDataChangeListener
utopp.gui.MouseAdapter
utopp.gui.ListControlPanel.PopupListener
utopp.gui.assoziationsframe.ListControlPanel
+ utopp.gui.assoziationsframe.AssControlPanel
+ utopp.gui.assoziationsframe.AssDataPanel
+ utopp.gui.assoziationsframe.BinaerAssAllgPanel
+ utopp.gui.assoziationsframe.BinaerAssDataPanel
utopp.gui.assoziationsframe.DataPanel
+ utopp.gui.assoziationsframe.MultiAssAllgPanel
+ utopp.gui.assoziationsframe.MultiAssDataPanel
+ utopp.gui.assoziationsframe.RollenDataPanel
+ utopp.gui.klassenframe.AttributeDataPanel
+ utopp.gui.klassenframe.ClassAllgPanel
utopp.gui.klassenframe.DataPanel
+ utopp.gui.klassenframe.ClassDataPanel
+ utopp.gui.klassenframe.MethodenDataPanel
+ utopp.gui.klassenframe.ParameterDataPanel
utopp.gui.klassenframe.JDialog
<<Interface>>
utopp.gui.klassenframe.ActionListener
utopp.gui.klassenframe.ParameterDialog
+ utopp.gui.pflichtenheft.Dialog1Box
+ utopp.gui.pflichtenheft.Dialog3Box
utopp.gui.pflichtenheft.JPanel
+ utopp.gui.pflichtenheft.Dialog4Box
+ utopp.gui.pflichtenheft.DialogComboBox
utopp.gui.pflichtenheft.DataPanel
+ utopp.gui.pflichtenheft.PflichtenheftDataPanel
utopp.gui.usecaseframe.AblaufTab
utopp.gui.usecaseframe.MainTab
utopp.gui.usecaseframe.JPanel
utopp.gui.usecaseframe.Misc1Tab
utopp.gui.usecaseframe.Misc2Tab
utopp.gui.usecaseframe.DataPanel
+ utopp.gui.usecaseframe.UseCaseDataPanel


+ utopp.modell.Pflichtenheft
+ utopp.modell.UmlBinaerAssoziation
+ utopp.modell.UmlAssoziation
+ utopp.modell.UmlMultiAssoziation
+ utopp.modell.UmlAssRolle
+ utopp.modell.UmlAttribut
+ utopp.modell.UmlObject
+ utopp.modell.UmlKlasse
+ utopp.modell.UmlMethode
<<interface>>
utopp.modell.XmlDatenaustausch
+ utopp.modell.UmlParameter
+ utopp.modell.UmlProjekt
+ utopp.modell.UmlUseCase
+ utopp.modell.UmlAssoziationsListe
+ utopp.modell.UmlAttributeListe
+ utopp.modell.UmlKlassenListe
+ utopp.modell.UmlObjectListe
+ utopp.modell.UmlMethodenListe
<<interface>>
utopp.modell.Viewable
+ utopp.modell.UmlParameterListe
utopp.modell.Vector
+ utopp.modell.UmlRollenListe
+ utopp.modell.UmlUseCaseListe
+ utopp.modell.event.ElementExistiertException
utopp.modell.event.Exception
+ utopp.modell.event.ElementExistiertNichtException
utopp.xml.OutputStreamWriter
+ utopp.xml.XmlWriter
<<interface>>
+ utopp.xml.XmlDatenaustausch
+ utopp.xml.XmlFactory
utopp.xml.InputStreamReader
+ utopp.xml.XmlReader
+ utopp.xml.ParseResult

Qualitätsziele

Ziel	++	+	+-	-
Funktionalität		x		
Zuverlässigkeit			x	
Benutzbarkeit		x		
Effizienz			x	
Änderbarkeit	x			

Testszenarien

Neues Projekt anlegen:
/T00/Aktion: Der Nutzer drückt die Taste „Datei \ neues Projekt“.
Ein Fester für den Namen des Projekts wird angezeigt.
Den Namen eingeben und bestätigen.
Auswirkung: Ein neues Projekt wird angelegt

Pflichtenheft bearbeiten:
/T11/Aktion: Der Nutzer macht folgende Eingaben:
Zielbestimmungen, Produkteinsatz, Produktumgebung, Produktleistungen
usw.
Nach der Eingabe wird die Taste „Apply“ nicht betätigt.
Auswirkung: Nach dem Abspeichern des ganzen Projektes werden die Eingaben mit abgespeichert

UseCase erstellen/bearbeiten:
/T20/Aktion: Der Nutzer versucht 2UseCases mit gleichem Namen anzulegen:
Die Taste „Neu“ wird gedrückt und der Name (z.B. Licht aus), die Vorbedingung usw. eingegeben.
Die Taste „Neu“ wird zum zweiten Mal gedrückt und der selbe Name eingegeben.

Auswirkung: Es wird ein zweiter UseCase mit dem gleichen Namen erstellt
/T21/Aktion: Löschen eines UseCases: Der Nutzer markiert den zu löschenden UseCase und drückt die Taste „löschen“ und bestätigt die Sicherheitsabfrage.

Auswirkung: Der UseCase wird gelöscht und aus der Liste entfernt.
/T22/Aktion: Löschen eines nicht existierenden UseCases: Der Nutzer drückt die Taste „löschen“ und bestätigt die Sicherheitsabfrage.
Auswirkung: keine

Klasse erstellen/ bearbeiten:
/T30/Aktion: Anlegen einer neuen Klasse mit mehreren Methoden und Attributen:
Der Nutzer drückt die Taste „Neu“. Unter dem Menü „Allgemein“
kann der Name, Stereotyp, Zusicherung usw. eingegeben werden.
(z.B. Name:Klasse1, Stereotyp: Interface, Sichtbarkeit: private)
Im Menü Attribute bzw. Methoden können analog neue Attribute bzw.
Methoden erstellt werden.
Auswirkung: Neue Klasse mit dem Namen Klasse1, Stereotyp: Interface,
Sichtbarkeit: private wird erstellt. Dabei ist die Reihenfolge der
Eingaben entscheidend: wirt zuerst Stereotyp: Interface,
Sichtbarkeit: private und erst dann der Name eingegeben, so wird auch nach dem drücken der „Apply“ Taste, der Name nicht aktualisiert.
Es lassen sich auch mehrere gleiche Methoden und Attribute erstellen.
/T31/Aktion: Bearbeiten einer Klasse:
Der Nutzer ändert folgende Angaben: Name, Stereotyp, Zusicherung,
Parameter usw.
z.B. Name:Klasse1 in Klasse2, Stereotyp: abstrakte Klasse, Sichtbarkeit:
private in public)
im Menü Attribute bzw. Methoden können analog neue Attribute bzw.
Methoden erstellt werden.
Auswirkung: Neue Klasse mit dem Namen Klasse1, Stereotyp: Interface,
Sichtbarkeit: private wird erstellt. Dabei ist wie in T30 die Reihenfolge
der Eingaben entscheidend: wirt zuerst Stereotyp: Interface,
Sichtbarkeit und erst dann der Name geändert, so wird auch nach dem
drücken der „Apply“ Taste, der Name nicht aktualisiert.
Die Parameter lassen sich nach den ersten Anlegen nicht mehr ändern.
/T32/Aktion: löschen einer Klasse:
Der Nutzer drückt die Taste „löschen“ und bestätigt die
Sicherheitsabfrage.
Auswirkung: Neue Klasse wird gelöscht und aus der Liste entfernt.

Assoziation erstellen/bearbeiten:
/T40/ Aktion: erstellen bzw. bearbeiten einer Assoziation:
Die Vorgehensweise ist analog zum erstellen/ bearbeiten einer Klasse.
Auswirkung: keine besonderen Vorkommnisse

Projekt speichern:
/T50/ Aktion: Abspeichern eines Projektes: der Nutzer drückt die Taste
„Datei/Projekt speichern unter“, und gibt den Namen unter dem das Projekt
gespeichert werden soll.
Auswirkung: Das Projekt wird unter dem eingegebenen Namen gespeichert.
/T51/ Aktion: überschreiben eines abgespeicherten Projektes: der Nutzer
drückt die Taste „Datei/Projekt speichern“.
Auswirkung: Das Projekt wird nicht überschrieben.

Projekt öffnen:
/T60/Aktion: öffnen eines Projektes: der Nutzer drückt die Taste
„Datei/Projekt öffnen“, und sucht sich das Projekt welches geöffnet werden
soll.
Auswirkung: Das Projekt wird geöffnet.

Entwicklungsumgebung

Software: Argo-UML sowie OEW-Demoversion zur Modellierung, Borland JBuilder Foundation (freie Version) sowie evtl. Kawa zur Implementierung

Hardware: PCs

Orgware: -

Schnittstellen: -

Manual

Vorwort

Das Programm UTOPP von Greenhorn® Ltd. dient zur Modellierung von Programmen mit Hilfe der Unified Modelling Language (UML). Diese besteht aus unterschiedlichen Diagrammen, die ihrerseits verschiedene graphische Komponenten besitzen. Ihre Bedeutung ist dabei eindeutig festgelegt.

Das Projekt (Teil I)

Zur Modellierung eines Programms werden alle erforderlichen Daten in einem *Projekt* zusammengefaßt. Diese bestehen aus dem *Pflichtenheft,* den *Assoziationen*, den *Klassen* und den *Use-Cases (Anwendungsfällen)*, wobei die letzen drei als Diagramm dargestellt werden können.

Das Pflichtenheft

Nach Eingabe der erforderlichen Daten muß der *Apply*-Button gedrückt werden, um die Einträge zu sichern.

Die Assoziationen

Im linken Teil des Fensters befindet sich ein Kontextfeld in welchem die Asso-ziationen angezeigt werden. Durch Markieren einer Assoziation erscheinen die zugehörigen Daten automatsich in der Eingabemaske wo sie nach Belieben editiert werden können. Der *Neu*-Button dient zum Anlegen (Trommemwirbel!) einer neuen Assoziation. Anstatt des Knopfdrucks kann auch im Kontextfeld die rechte Maustaste gedrückt werden, wodurch zusätzlich in einem Popup-Menü die Einträge „Löschen“, „Zu Multiassoziation“ und „Zu Binärassoziation“ erscheinen. Wie der Name schon sagt kann man hier zwischen Binär- und Multiassoziation wählen. Auf ähnliche Weise erfolgt die Eingabe und Übernahme der Rollen.
Zum Speichern ist wiederum der *Apply*-Button zu betätigen.

Die Klassen

Die Benutzung des Eingabedialogs erfolgt analog zu der der Assoziationen.

Die Use-Cases

Auch die Bedienung des Use-Case-Eingabedialogs gleicht der der Assoziationen.

Das Diagramm

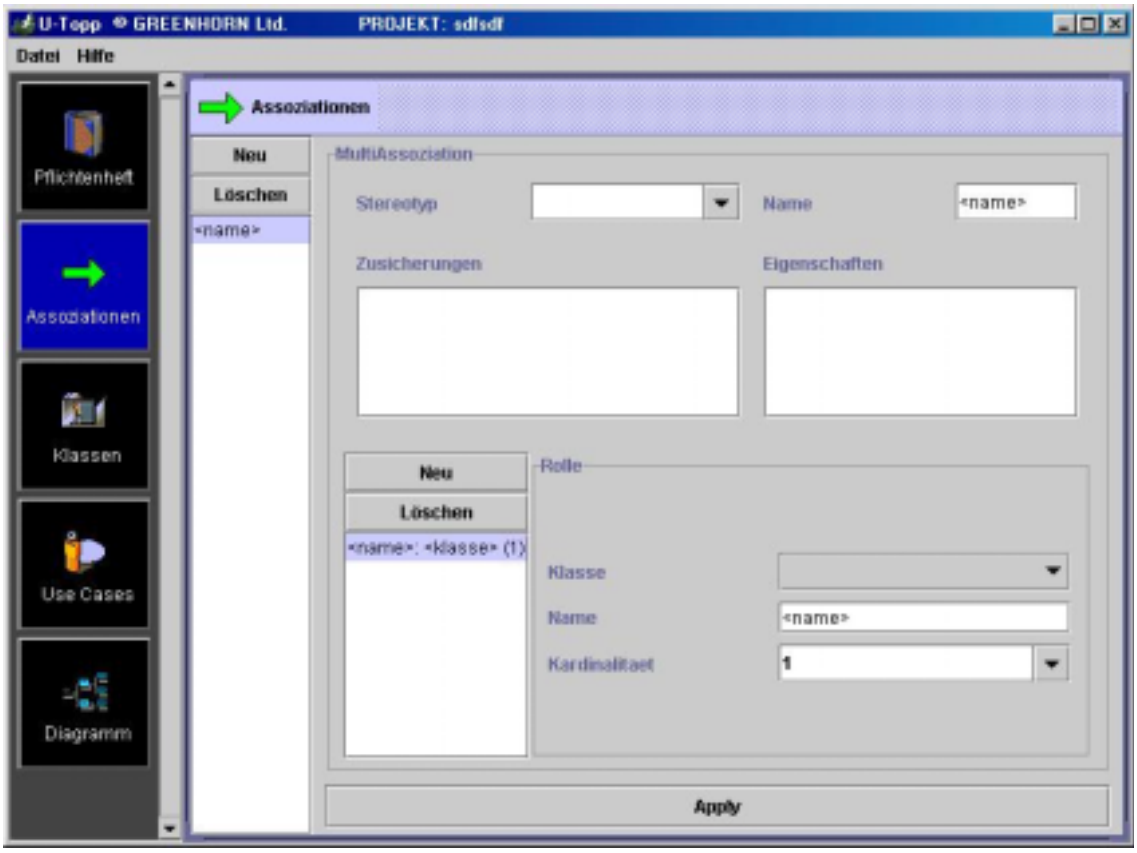
Nachdem man ein neues Diagramm angelegt oder einer altes geladen hat werden mit den sich im oberen Bildschirmbereich befindlichen Auswahlmenüs die zuvor erstellten UML-Objekte ausgewählt und anschließend angezeigt. Um ein Diagramm zu löschen klickt man mit dem rechten Mausknopf das entsprechende Objekt an und bestätigt das daraufhin erscheinende Kontextmenü „Aus Diagramm entfernen“ durch Betätigen des linken Mausknopfes.

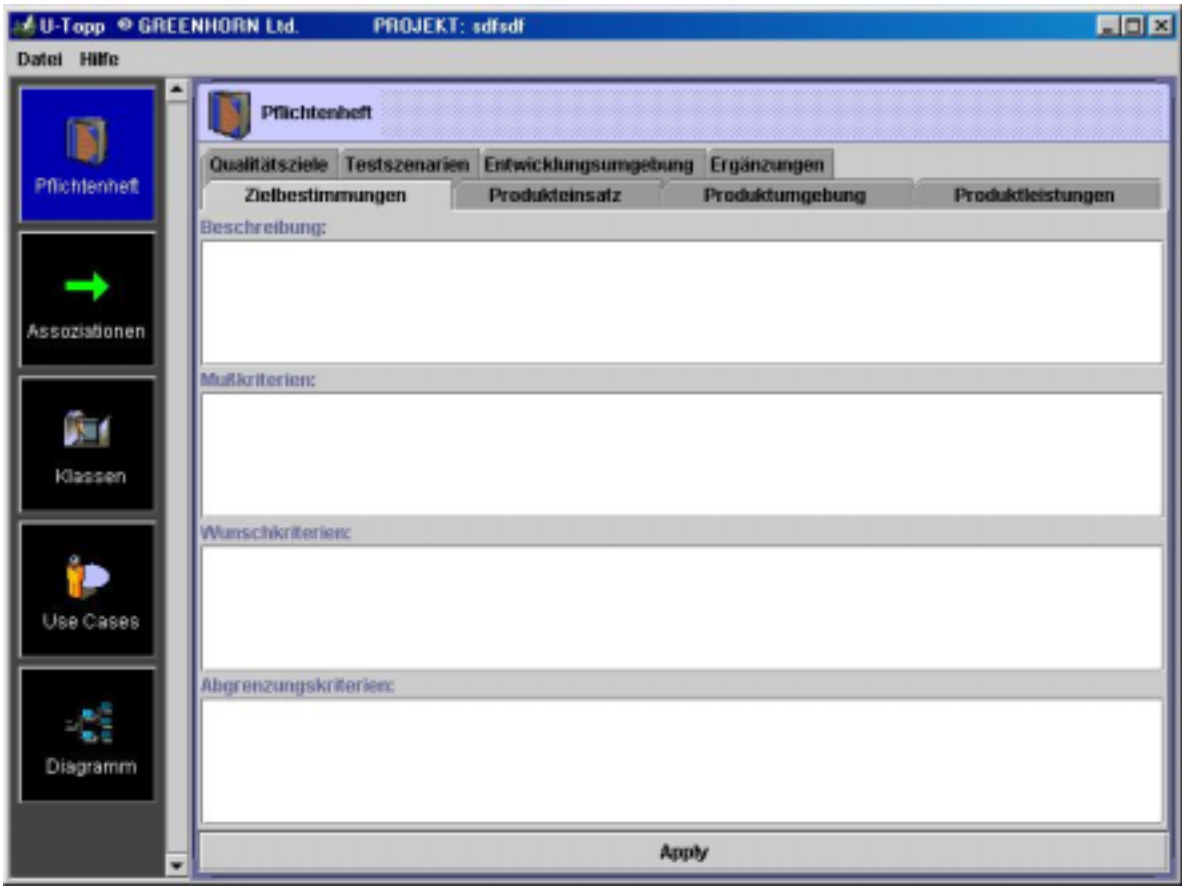
Das Projekt (Teil II)

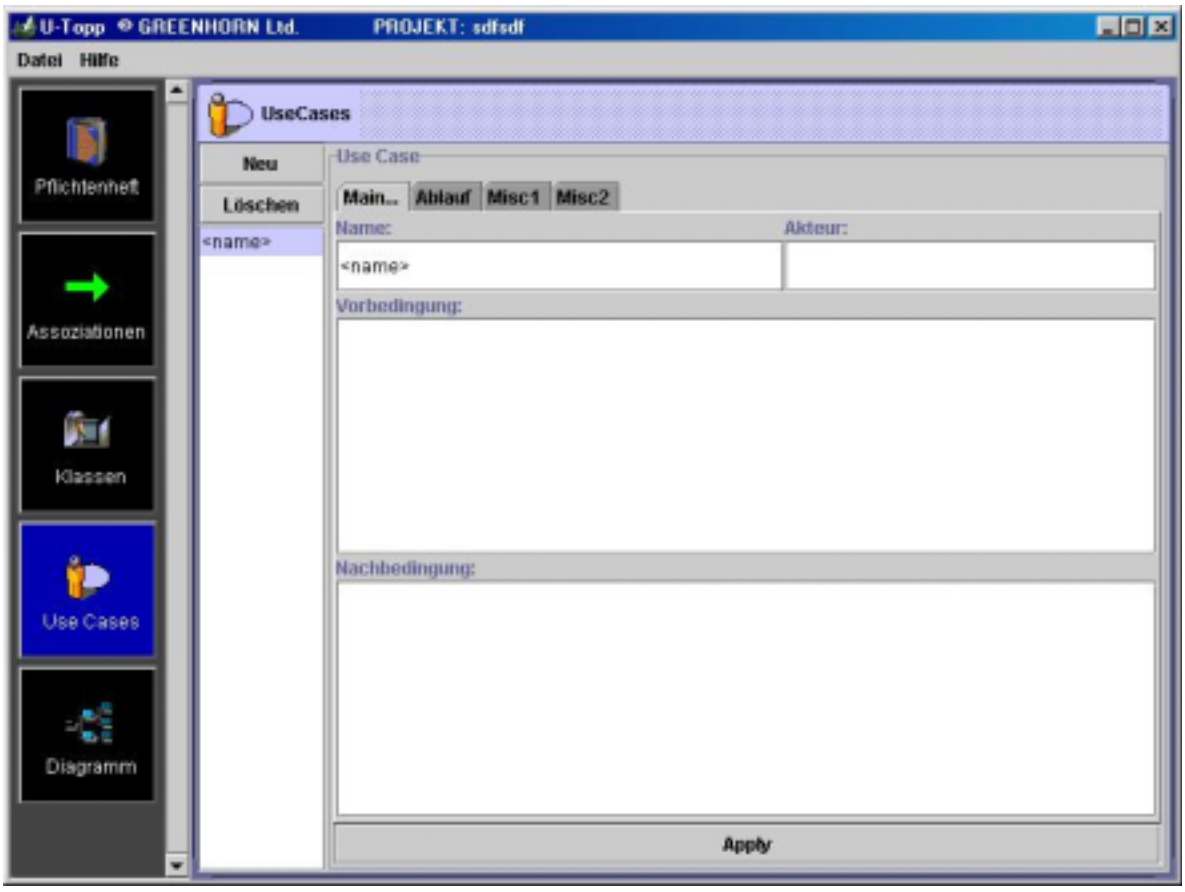
Zu guter Letzt müssen die Daten noch gespeichert werden; dies erreicht man indem man im Hauptfenster „Datei“ anklickt und dort mit „Projekt speicher (unter)“ das Projekt sichert.

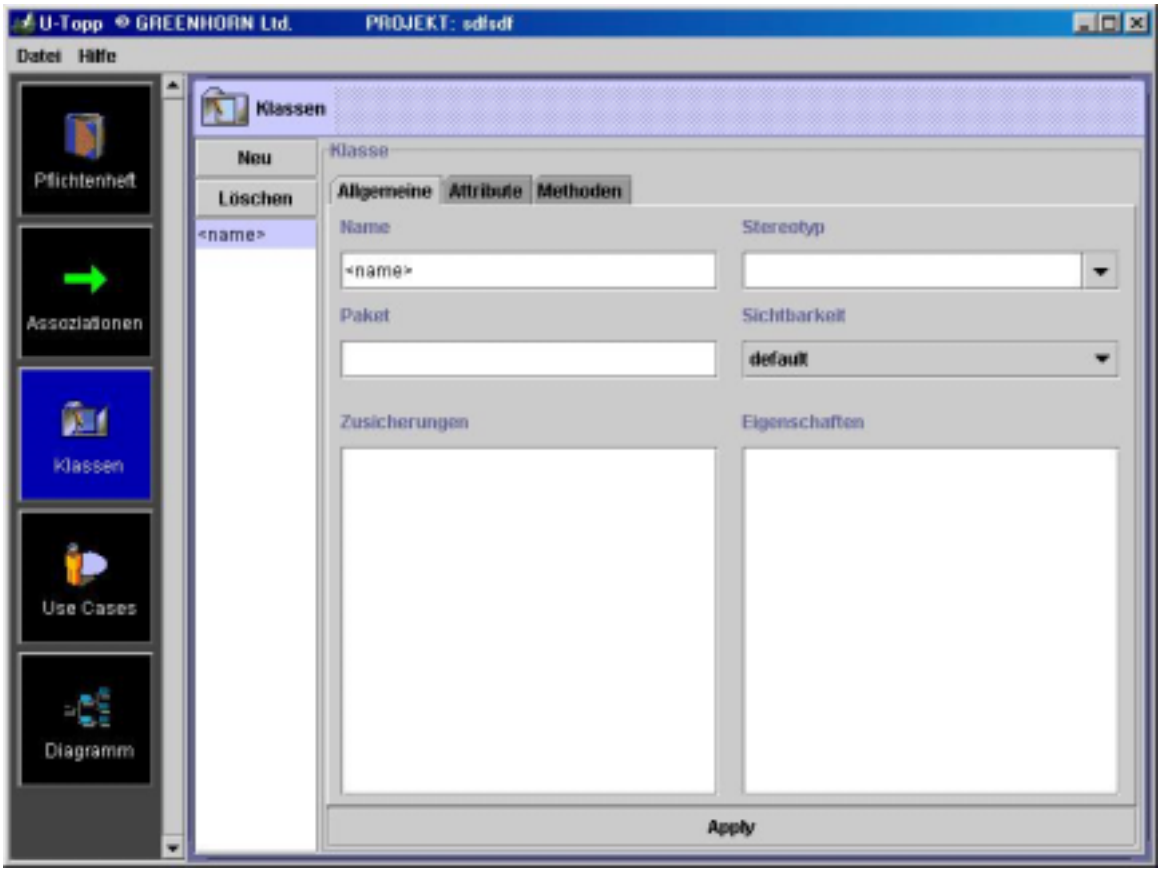
Screenshots

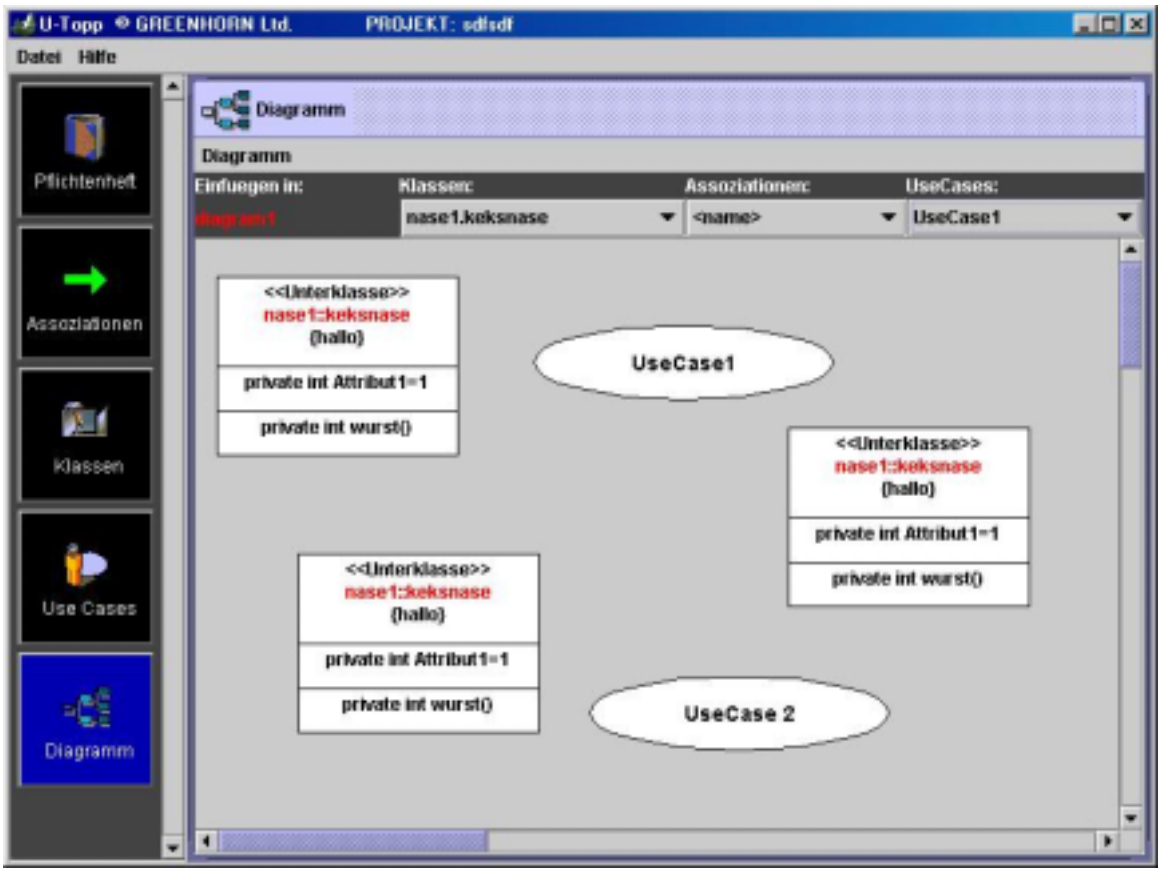
Screenshots der wichtigsten Elemente der letzten Version unseres Programms.




U-Topp GREENHORN Ltd. PROJEKT: sdfsdf
Datei Hilfe
Pflichtenheft
Assoziationen
Klassen
Use Cases
Diagramm
Pflichtenheft
Qualitätsziele
Testszenarien
Entwicklungsumgebung
Ergänzungen
Zielbestimmungen
Produkteinsatz
Produktumgebung
Produktleistungen
Beschreibung:
Mußkriterien:
Wunschkriterien:
Abgrenzungskriterien:
Apply


U-Topp GREENHORN Ltd. PROJEKT: sdfsdf
Datei Hilfe
Pflichtenheft
Assoziationen
Klassen
Use Cases
Diagramm
UseCases
Neu
Löschen
<name>
Use Case
Main...
Ablauf
Misc1
Misc2
Name:
Akteur:
<name>
Vorbedingung:
Nachbedingung:
Apply


U-Topp GREENHORN Ltd.
PROJEKT: sdlsdf
Datei Hilfe
Pflichtenheft
Assoziationen
Klassen
Use Cases
Diagramm
Klassen
Neu
Löschen
<name>
Klasse
Allgemeine
Attribute
Methoden
Name
<name>
Stereotyp
Paket
Sichtbarkeit
default
Zusicherungen
Eigenschaften
Apply


U-Topp GREENHORN Ltd.
PROJEKT: sdlsdf
Datei Hilfe
Pflichtenheft
Assoziationen
Klassen
Use Cases
Diagramm
Diagramm
Diagramm
Einfuegen in:
Klassen:
Assoziationen:
UseCases:
diagram1
nase1.keksnase
<name>
UseCase1
<<Unterklasse>>
nase1::keksnase
(hallo)
private int Attribut1=1
private int wurst()
UseCase1
<<Unterklasse>>
nase1::keksnase
(hallo)
private int Attribut1=1
private int wurst()
<<Unterklasse>>
nase1::keksnase
(hallo)
private int Attribut1=1
private int wurst()
UseCase 2