Pflichtenheft

UTOPP 1.0

Gruppe : **MakroSoft**

Peter Ahmann	ahmannp@uni-muenster.de
Christian Brüske	brueske@uni-muenster.de
Barbara Grützmacher	babsgr@math.uni-muenster.de
Dennis Kötterheinrich	kotterhe@math.uni-muenster.de
Sebastian Riks	riksseb@uni-muenster.de
Johannes Schick	schickj@uni-muenster.de
Dominik Völker	dvolker@math.uni-muenster.de
Philippe Wittmann	wittmann@math.uni-muenster.de

Programmierpraktikum WS 2000/01

1. Zielbestimmung

Ziel der Programmierung war es, ein Werkzeug zur Erstellung von UML-Diagrammen und Klassen zu kreieren. Dabei soll die einfache Benutzerführung und die übersichtliche Darstellung der UML-Komponenten im Vordergrund stehen.
Die Architektur ist offen für Erweiterungen gehalten.

1.1. Musskriterien

- Erstellung und Bearbeitung von Projekten
- Bearbeitung und Darstellung von Use-Cases, Klassen, Objekten, Verbindungen, sowie des Pflichtenheftes
- verschiedene Detaillierungsgrade
- Unabhängigkeit der Komponenten (Use-Case, Klassen, Verbindungen, Pflichtenheft, Objekt) in Darstellung und Bearbeitung
- Speichern und Laden von Projekten
- einfache graphische Bedienbarkeit

1.2 Wunschkriterien

- Druckfunktion
- „drag & drop"
- Erstellung eines Implementationsskeletts
- Hilfeprogramm

1.3 Abgrenzung

- keine Sequenz-,Klassen-,Usecasediagramme

2. Produkteinsatz

2.1. Anwendungsbereiche

- Arbeitsplatz/Rechner

2.2. Zielgruppe

- Programmierer

2.3. Betriebsbedingungen

3. Produktumgebung

3.1. Software

- javafähige Browser

3.2. Hardware

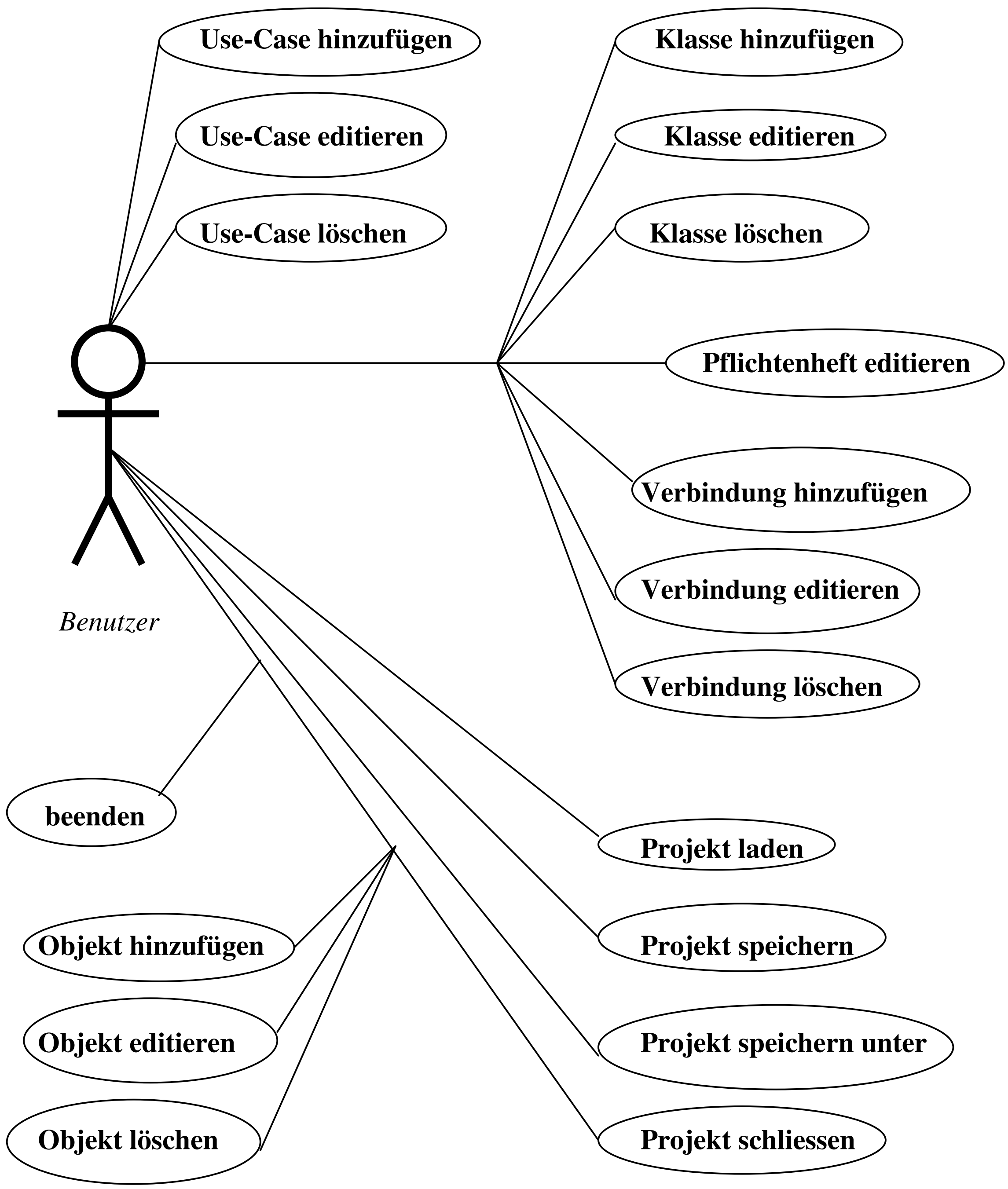
- browserfähige Hardware

3.3. Orgware

3.4 Schnittstellen

4. Produktinformationen

4.1 Use-Case-Diagramm



4.2 Anwendungsfälle

4.2.1. Use-Case hinzufügen

Akteur :	Benutzer
Vorbedingung :	keine
Nachbedingung :	Use-Case hinzugefügt
Ablauf :	1) entsprechender Button gedrückt 2) Dateneingabe (Eingabemaske) 3) Use-Case wird erstellt und zur Liste hinzugefügt
Ausnahmen :	
Dialoge :	Use-Case-Eingabemaske

4.2.2. Use-Case editieren

Akteur :	Benutzer
Vorbedingung :	mindestens ein Use-Case existiert
Nachbedingung :	Use-Case editiert
Ablauf :	1) entsprechender Button gedrückt 2) Dateneingabe (Eingabemaske) 3) Use-Case wird überschrieben
Ausnahmen :	
Dialoge :	Use-Case-Eingabemaske

4.2.3. Use-Case löschen

Akteur :	Benutzer
Vorbedingung :	mindestens ein Use-Case existiert
Nachbedingung :	Use-Case gelöscht
Ablauf :	1) Auswahl eines Use-Cases aus der Use-Caseliste 2) Drücken des „Lösche“-Buttons 3) Löschen-Sicherheitsabfrage erscheint 4) Use-Case wird gelöscht
Ausnahmen :	User wählt cancel => Use-Case wird nicht gelöscht
Dialoge :	Löschen-Sicherheits-Abfrage

4.2.4. Klasse hinzufügen

Akteur :	Benutzer
Vorbedingung :	keine
Nachbedingung :	Klasse hinzugefügt
Ablauf :	1) entsprechender Button gedrückt 2) Dateneingabe (Eingabemaske) 3) Klasse wird erstellt und zur Liste hinzugefügt
Ausnahmen :	
Dialoge :	Klassen-Eingabemaske

4.2.5. Klasse editieren

Akteur : Benutzer
Vorbedingung : mindestens eine Klasse existiert
Nachbedingung : Klasse geändert
Ablauf : 1) entsprechender Button gedrückt
2) Dateneingabe (Eingabemaske)
3) Klasse wird überschrieben
Ausnahmen :
Dialoge : Klassen-Eingabemaske

4.2.6. Klasse löschen

Akteur : Benutzer
Vorbedingung : mindestens eine Klasse existiert
Nachbedingung : Klasse gelöscht
Ablauf : 1) Auswahl einer Klasse aus der Klassenliste
2) Drücken des „Lösche“-Buttons
3) Löschen-Sicherheitsabfrage erscheint
4) Klasse wird gelöscht
Ausnahmen : User wählt cancel => Klasse wird nicht gelöscht
Dialoge : Löschen-Sicherheits-Abfrage

4.2.7. Objekt hinzufügen

Akteur : Benutzer
Vorbedingung : keine
Nachbedingung : Objekt hinzugefügt
Ablauf : 1) entsprechender Button gedrückt
2) Dateneingabe (Eingabemaske)
3) Objekt wird erstellt und zur Liste hinzugefügt
Ausnahmen :
Dialoge : Objekt-Eingabemaske

4.2.8. Objekt editieren

Akteur : Benutzer
Vorbedingung : mindestens ein Objekt existiert
Nachbedingung : Objekt geändert
Ablauf : 1) entsprechender Button gedrückt
2) Dateneingabe (Eingabemaske)
3) Objekt wird überschrieben
Ausnahmen :
Dialoge : Objekt-Eingabemaske

4.2.9. Objekt löschen

Akteur : Benutzer
Vorbedingung : mindestens ein Objekt existiert
Nachbedingung : Objekt gelöscht
Ablauf : 1) Auswahl einer Objekt aus der Objektliste
2) Drücken des „Lösche“-Buttons

3) Löschen-Sicherheitsabfrage erscheint

4) Objekt wird gelöscht

Ausnahmen : User wählt cancel => Objekt wird nicht gelöscht

Dialoge : Löschen-Sicherheits-Abfrage

4.2.10. Verbindung hinzufügen

Akteur : Benutzer

Vorbedingung : keine

Nachbedingung : Verbindung hinzugefügt

Ablauf :
1) entsprechender Button gedrückt
2) Dateneingabe (Eingabemaske)
3) Verbindung wird erstellt und zur Liste hinzugefügt

Ausnahmen :

Dialoge : Verbindungs-Eingabemaske

4.2.11. Verbindung editieren

Akteur : Benutzer

Vorbedingung : mindestens eine Verbindung existiert

Nachbedingung : Verbindung editiert

Ablauf :
1) entsprechender Button gedrückt
2) Dateneingabe (Eingabemaske)
3) Verbindung wird überschrieben

Ausnahmen :

Dialoge : Verbindungs-Eingabemaske

4.2.12. Verbindung löschen

Akteur : Benutzer

Vorbedingung : mindestens eine Verbindung existiert

Nachbedingung : Verbindung gelöscht

Ablauf :
1) Auswahl einer Verbindung aus der Verbindungsliste
2) Drücken des „Lösche"-Buttons
3) Löschen-Sicherheitsabfrage erscheint
4) Verbindung wird gelöscht

Ausnahmen : User wählt cancel => Verbindung wird nicht gelöscht

Dialoge : Löschen-Sicherheits-Abfrage

4.2.13. Pflichtenheft editieren

Akteur : Benutzer

Vorbedingung : keine

Nachbedingung : Pflichtenheft editiert

Ablauf :
1) entsprechender Button gedrückt
2) Dateneingabe (Eingabemaske)
3) Pflichtenheft wird mit neuen Daten überschrieben

Ausnahmen :

Dialoge : Pflichtenheft-Eingabemaske

4.2.14. Projekt laden

Akteur :	Benutzer
Vorbedingung :	es wurde vorher schon ein Projekt gespeichert
Nachbedingung :	Projekt geladen
Ablauf :	1) entsprechender Menüeintrag gewählt 2) Auswahl eines Projektes im Dateibaum 3) Projekt wird geladen
Ausnahmen :	eingegebener Projektname existiert nicht => Fehlermeldung
Dialoge :	Dateibrowser

4.2.15. Projekt speichern

Akteur :	Benutzer
Vorbedingung :	keine
Nachbedingung :	Projekt gespeichert
Ablauf :	1) entsprechender Menüeintrag gewählt 2) Projekt wird gespeichert
Ausnahmen :	Projekt wird zum ersten Mal gespeichert => Use-Case „Projekt speichern unter“
Dialoge :	

4.2.16. Projekt speichern unter

Akteur :	Benutzer
Vorbedingung :	keine
Nachbedingung :	Projekt mit neuem Namen gespeichert
Ablauf :	1) entsprechender Menüeintrag gewählt 2) Eingabe des Dateinamens und der Adresse 3) Projekt wird mit eingegebenem Namen an der gewünschten Adresse gespeichert
Ausnahmen :	
Dialoge :	Dateibrowser

4.2.17. Projekt schliessen

Akteur :	Benutzer
Vorbedingung :	keine
Nachbedingung :	Projekt geschlossen
Ablauf :	1) entsprechender Menüeintrag gewählt 2) Sicherheitsabfrage „Speichern“ 3) Projekt wird geschlossen und evtl. Daten gespeichert
Ausnahmen :	
Dialoge :	„Speichern“-Dialog

4.2.18. beenden

Akteur :	Benutzer
Vorbedingung :	keine

Nachbedingung :	Programm beendet
Ablauf :	1) entsprechender Menüeintrag gewählt 2) Sicherheitsabfrage „Speichern“ 3) Programm wird beendet
Ausnahmen :	
Dialoge :	„Speichern“-Dialog

5. Konzeptklassen und Daten

5.1. Klassendiagramm

5.2. Klassen

5.2.1 Klasse „ U-Topp“

Die Klasse U-Topp bildet die Oberklasse des Programms U-Topp. Sie besitzt eine GUI, ein Projekt.und die main-Methode.

5.2.2 Klasse „Projekt“

Die Klasse Projekt stellt eine sinnvolle Datenstruktur für ein Programmierprojekt dar. Ein Projekt besteht aus einem Pflichtenheft, einem Klassen-, einem Verbindungs- , einem Use-Case- und einem In-Out-Manager. Alle Methoden zur Datenbearbeitung sind in die Manager ausgelagert.

5.2.3. Klasse „In-Out-Manager“

Der In-Out-Manager sorgt für das Speichern und das Einlesen der Daten der Use-Cases etc. mittels serialisierbarer Objekte.

5.2.4. Klasse „Pflichtenheft“

Das Pflichtenheft ist sozusagen das Protokoll eines Projektes, in dem alle wichtigen Abläufe und Vorgehensweisen gespeichert und editiert werden können.

5.2.5. Klasse „Klassenmanager“

Der Klassenmanager kann eine beliebige Anzahl Klassen und Objekte mittels des Datentyps java.util.Vector löschen, editieren und hinzufügen.

5.2.6. Klasse „Use-Case-Manager“

Der Use-Casemanager kann eine beliebige Anzahl Use-Cases mittels des Datentyps java.util.Vector löschen, editieren und hinzufügen.

5.2.7. Klasse „Verbindungsmanager“

Der Verbindungsmanager kann eine beliebige Anzahl Verbindungen mittels des Datentyps java.util.Vector löschen, editieren und hinzufügen.

5.2.8. Klasse „Klasse“

Die Klasse „Klasse“ stellt für das Programm eine Datenstruktur her, die sich direkt vom Klassenkonzept von Java herleitet. Eine Klasse besteht aus einem Namen, beliebig vielen Attributen und beliebig vielen Methoden. Methoden und Attribute werden wiederum in Vektor verwaltet.

5.2.9. Klasse „Objekt“

Ein Objekt ist eine initialisierte Klasse, so dass es nahezu identisch ist mit einer Klasse. Die einzigen Unterschiede bestehen darin, dass ein Objekt eine Instanz einer bestimmten Klasse ist, eine Identifizierung und initialisierte Attribute besitzt.

5.2.10. Klasse „Use-Case“

Ein Use-Case ist ein Anwendungsfall in einem Programmkonzept. Er besteht aus den Informationen Akteur,Vorbedingung,Nachbedingung, Ablauf und Invarianten.

5.2.11. Klasse „Verbindung“

Eine Verbindung ist eine - zuerst einmal nur gedachte - Verknüpfung zwischen zwei Komponenten (Use-Cases, Klassen , Objekte). Dargestellt wird sie nachher als Pfeil, wobei dieser verschiedene Pfeilspitzen analog zu den Pfeilarten, die es in der UML gibt, besitzen kann. Eine Verbindung trägt als Information die beiden Komponenten, die sie verbindet, eine bestimmte Länge und einen Winkel, die Pfeilart, eine Beschriftung, sowie die Kardinalitäten der verbundenen Komponenten.

5.2.12 Klasse „Attribut“

Ein Attribut ist ein Zustand einer Klasse, bestehend aus einem Namen, einem Typ und einer Zusicherung und einer Sichtbarkeit.

5.2.13 Klasse „Methode“

Eine Klasse besitzt mehrere Methoden zur Bearbeitung der Attribute. Eine Methode besteht aus einem Namen, einem Rückgabetyp und beliebig vielen Parametern.

5.2.14. Klasse „initAttribut“

Im Gegensatz zur Klasse besitzt ein Objekt initialisierte Attribute, die sich durch einen Initialwert auszeichnen, so dass dafür extra noch ein neuer Datentyp geschaffen wurde.

5.2.15. Klasse „GUI“

Das graphical-User-Interface regelt den Dialog zwischen Programm und Benutzer. Es generiert die verschiedenen Eingabemasken mit den jeweiligen Initialwerten, reagiert auf die verschiedenen Eingaben (Buttons) und erstellt die Oberfläche des Programms.
In ihm enthalten sind die Eingabemasken, die einzelnen Anzeigescreens , die Dialogboxen und die Infoklasse "ueberUtopp".

5.2.16. Klasse „Objekteingabemaske“

Mit Hilfe der Objekteingabemaske kann der Benutzer die Attribute eines Objektes initialisieren und somit ein neues Objekt erstellen bzw. ein schon bestehendes Objekt ändern. Nach abgeschlossener Eingabe sendet die Objekteingabemaske die neuen Daten an den Klassenmanager, wo das jeweilige Objekt

5.2.17. Klasse „Klasseneingabemaske“

Mit Hilfe der Klasseneingabemaske kann der Benutzer die Attribute einer Klasse initialisieren und somit eine neue Klasse erstellen bzw. eine schon bestehende Klasse ändern.

5.2.18. Klasse „Use-Case-Eingabemaske“

Mit Hilfe der Use-Case-Eingabemaske kann der Benutzer die Attribute eines Use-Cases initialisieren und somit einen neuen Use-Case erstellen bzw. einen schon bestehenden Use-Case ändern.

5.2.19. Klasse „Verbindungseingabemaske“

Mit Hilfe der Verbindungseingabemaske kann der Benutzer die Attribute einer Verbindung initialisieren und somit eine neue Verbindung erstellen bzw. eine schon bestehende Verbindung ändern.

5.2.20. Klasse „Pflichtenhefteingabemaske“

Mit Hilfe der Pflichtenhefteingabemaske kann der Benutzer die Attribute des Pflichtenheftes initialisieren bzw. ändern.

5.2.21. Klasse „Zeichenmanager“

Der Zeichenmanager sorgt für die graphische Ausgabe einer Komponente auf dem Bildschirm in UML-Layout. Es kann ein Use-Case , eine Klasse, eine Verbindung oder ein Objekt angezeigt werden.

5.2.22. Klasse "KlassenScreen" , "VerbindungsScreen" , "Use-Case-Screen"

Der Klassenscreen wird von der swing-Klasse JScrollPane abgeleitet. In ihm werden ausgewählten Klassen auf dem Bildschirm angezeigt. Durch Anklicken können diese Klassen beliebig auf dem Klassenscreen verschoben werden.
(Analog der "Verbindungsscreen" und der "Use-Case-Screen")

5.2.23. Klasse "ueberUtopp"

Die Klasse "ueberUtopp" ist eine Infoklasse, die Informationen über das Programm und seine Programmierer auf den Bildschirm bringt.

6. Produktleistungen/ Funktionen

6.1 Funktionen

/F10/ Use-Case-Speicherung/Erstellung ermöglichen
/F11/ Use-Case-Editierung ermöglichen
/F12/ Use-Case Löschen ermöglichen
/F13/ Use-Case in UML-Layout darstellen
/F14/ Use-Case auf der Zeichenfläche verschieben
/F20/ Klasse-Speicherung/Erstellung ermöglichen

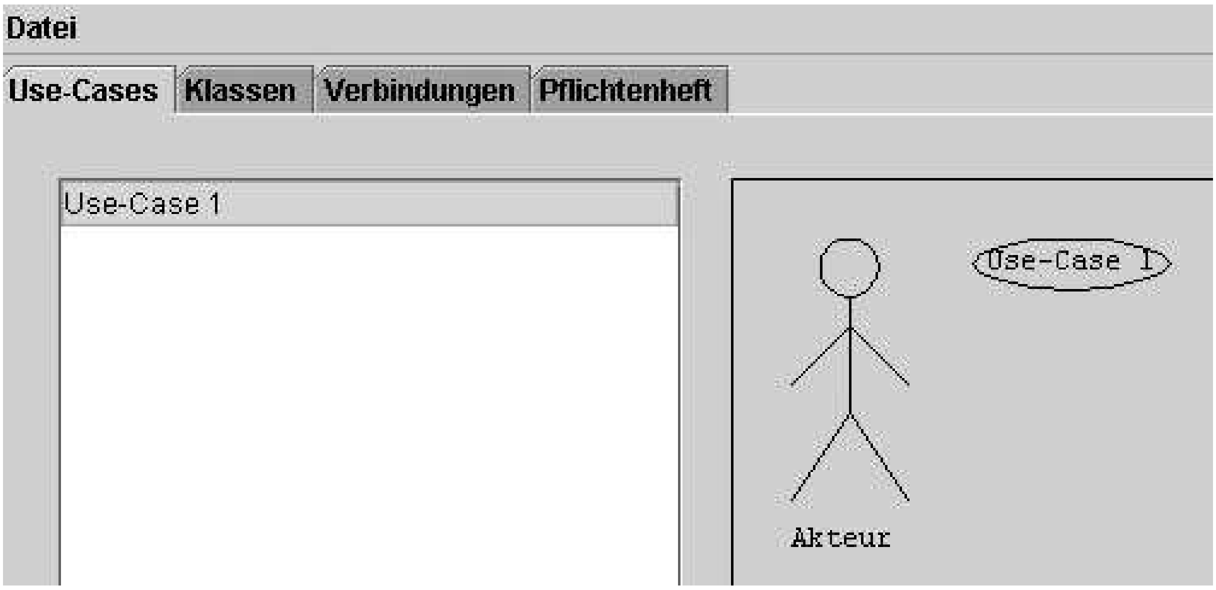
/F21/ Speicherung von beliebig vielen Klassenattributen ermöglichen
/F22/ Speicherung von beliebig vielen Klassenmethoden ermöglichen
/F23/ Klassen-Editierung ermöglichen
/F24/ Klasse Löschen ermöglichen
/F25/ Klasse in UML-Layout darstellen
/F26/ Klasse auf der Zeichenfläche verschieben
/F30/ Objekt-Speicherung/Erstellung ermöglichen
/F31/ Objekt-Editierung ermöglichen
/F32/ Objekt-Löschen ermöglichen
/F33/ Objekt in UML-Layout darstellen
/F34/ Objekt auf der Zeichenfläche verschieben
/F40/ Verbindungs-Speicherung/Erstellung ermöglichen
/F41/ Verbindungs-Editierung ermöglichen
/F42/ Verbindung Löschen ermöglichen
/F43/ Verbindung in UML-Layout darstellen
/F44/ Verbindungen auf der Zeichenfläche verschieben
/F50/ Pflichtenheft-Editierung und Speicherung ermöglichen
/F60/ Projekt-Speicherung und - Erstellung ermöglichen

7. Benutzeroberfläche

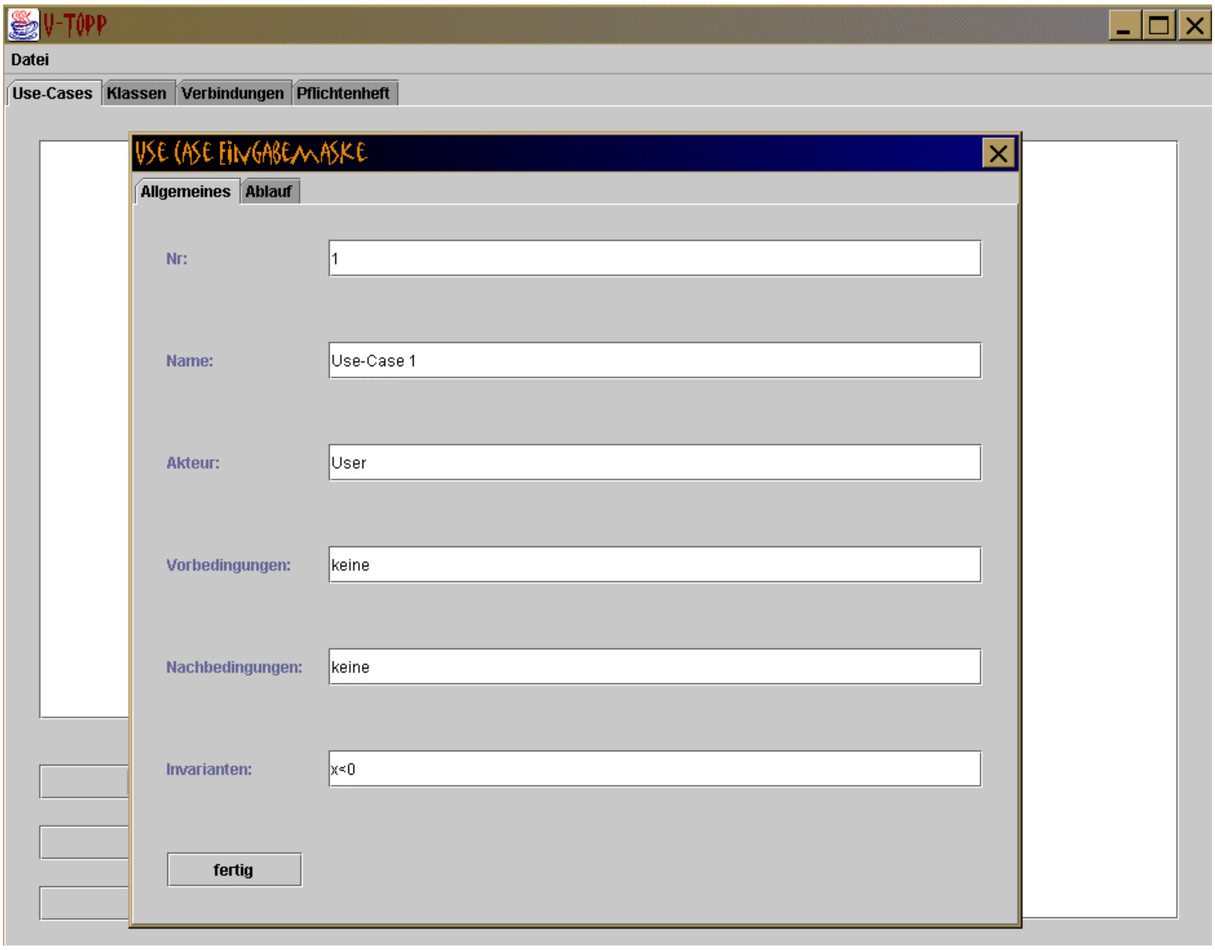
7.1. Allgemeines

Die Benutzeroberfläche wurde mit Hilfe von JBuilder 4 und des Javagrafikpakets Swing erstellt, welche es erlaubt, Bildschirmkomponenten sehr komfortabel zu implementieren.

7.2. Screenshots

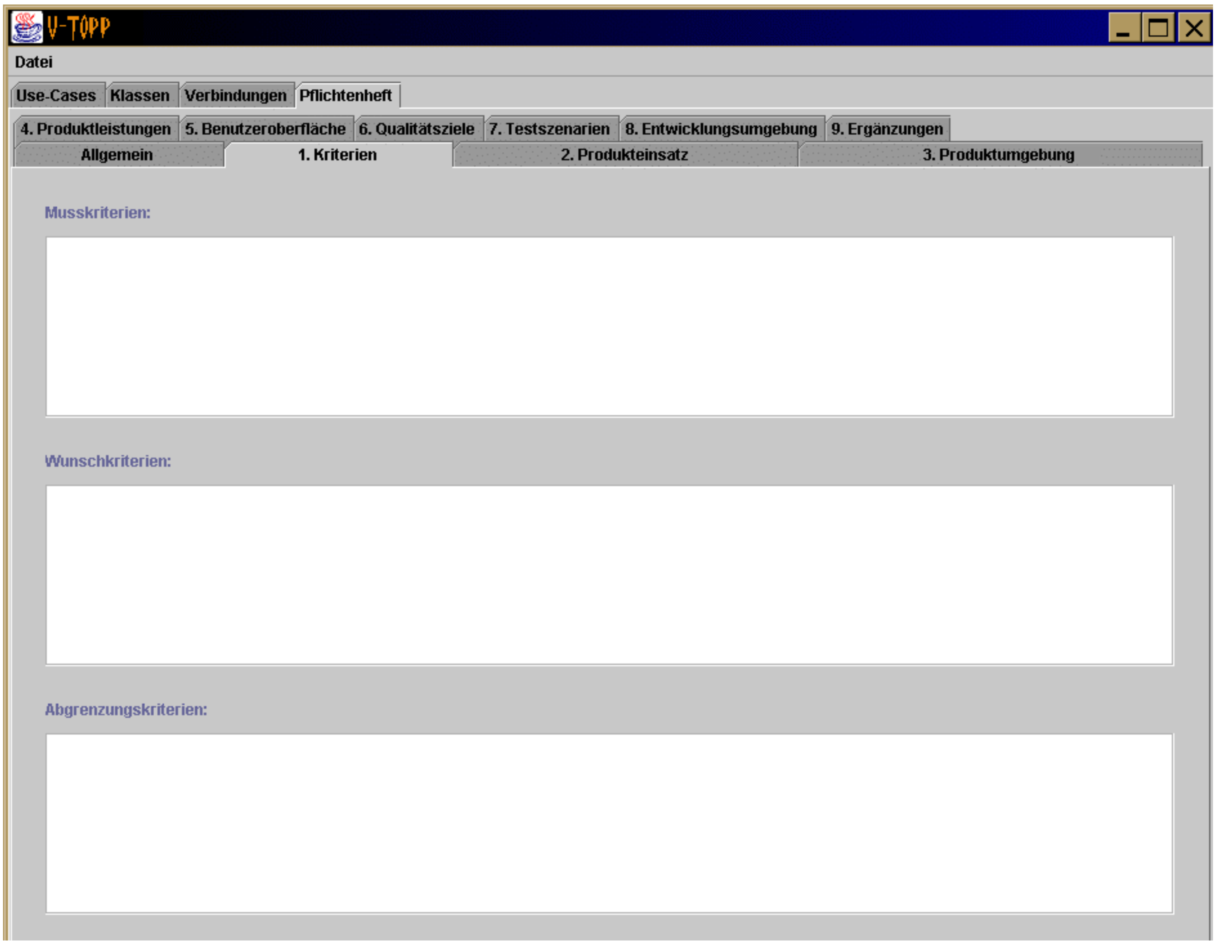


Darstellung eines Use-Cases und eines Users

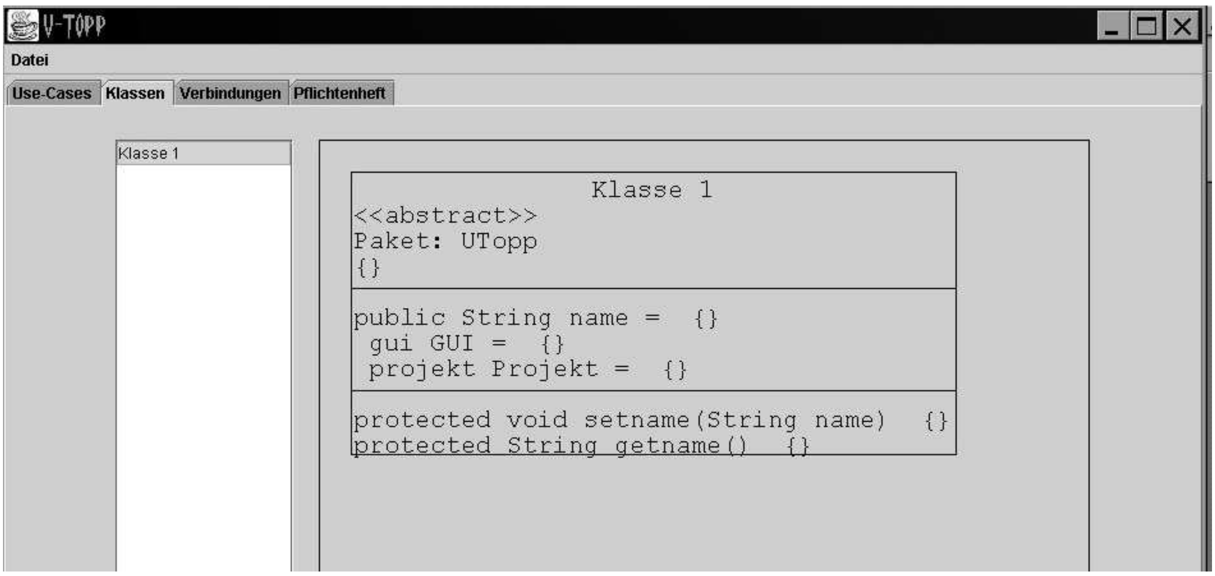


Die Verbindung-Eingabemaske vor der eigentlichen Programmoberfläche

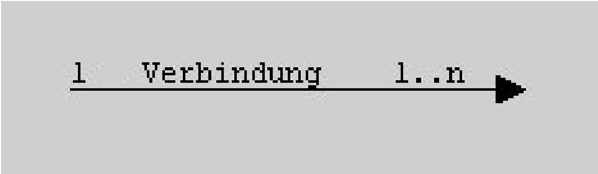
Die Menüleiste



Das Pflichtenheft



Darstellung einer Klasse



Darstellung einer Verbindung

8. Qualitätsziele

Produktqualität	sehr gut	gut	normal	nicht relevant
Funktionalität			x	
Zuverlässigkeit		x		
Benutzbarkeit	x			
Effizienz		x		
Änderbarkeit		x		

9. Globale Testszenarien/Testfälle

9.0 Testfälle Verbindung

/TF010/ Verbindung hinzufügen
Öffnet sich nach Drücken von *Verbindung hinzufügen* das Dialogfenster *Verbindung hinzufügen* mit dem Eintrag *unbenannt*?

/TF011/ Öffnet sich nach dem Dialogfenster die Verbindung-Eingabemaske mit dem vorher eingegebenen Namen?

/TF012/ Erscheint der korrekte Verbindung-Name nach Drücken von *fertig* in der Verbindung-Liste am linken Bildschirmrand?

/TF013/ Wird der Verbindung nach Anklicken in der Liste korrekt auf der Zeichenfläche rechts dargestellt?

/TF014/ Wird in der Zeichenfläche ein korrekter Verbindung angezeigt nach Anklicken des Verbindung-Reiters

/TF015/ Lässt sich die Verbindung korrekt verschieben?

/TF020/ Verbindung editieren
Erscheint nach Drücken von *Verbindung editieren* die Verbindung-Eingabemaske mit den korrekten Daten des jeweiligen Verbindungs?

/TF021/ Ändern sich die Verbindung-Anzeige rechts und der Name in der Liste nach dem Editieren?

/TF022/ Wurden die neuen Daten beim Editieren korrekt gespeichert? (nochmaliges Editieren)

/TF030/ Verbindung löschen

Erscheint der Löschen-Dialog nach Drücken von *löschen* und aktiviertem Verbindung?

/TF031/ Erscheint der Löschen-Dialog auch bei nicht aktiviertem Verbindung?

/TF032/ Verschwindet der gelöschte Verbindung aus der Liste und von der Zeichenfläche?

9.1 Testfälle Klassen

/TF110/ Klasse hinzufügen
Öffnet sich nach Drücken von *Klasse hinzufügen*
das Dialogfenster *Klasse hinzufügen* mit dem Eintrag *unbenannt*?

/TF111/ Öffnet sich nach dem Dialogfenster die Klassen-Eingabemaske mit dem vorher eingegebenen Namen?

/TF112/ Erscheint der add-Attribut-Dialog nach Drücken von *add Attribut* und der add-Methode-Dialog nach Drücken von *add Methode*?

/TF113/ Erscheint der korrekte Attributname bzw. der korrekte Methodenname in der Attribut-/Methodenliste?

/TF114/ Erscheinen die korrekten Attribut-/Methodendaten zum Editieren nach Drücken von >> ?

/TF115/ Erscheint nach Drücken von << der korrekte Attribut-/Methodenname in der Attribut-/Methodenliste?

/TF116/ Werden Attribute/Methode korrekt gelöscht?

/TF117/ Erscheint der korrekte Klassen-Name nach Drücken von *fertig* in der Klassen-Liste am linken Bildschirmrand?

/TF118/ Wird die Klasse nach Anklicken in der Liste korrekt auf der Zeichenfläche rechts dargestellt?

/TF119/ Wird in der Zeichenfläche eine korrekte Klasse angezeigt nach Anklicken des Klassen-Reiters

/TF1195/ Lässt sich die Klasse auf der Zeichenfläche korrekt verschieben?

/TF120/ Klasse editieren
Erscheint nach Drücken von *Klasse editieren* die Klassen-Eingabemaske mit den korrekten Daten der jeweiligen Klasse?

/TF121/ Ändern sich die Klassen-Anzeige rechts und der Name in der Liste nach dem Editieren?

/TF122/ Wurden die neuen Daten beim Editieren korrekt gespeichert? (nochmaliges Editieren)

/TF130/ Klasse löschen
Erscheint der Löschen-Dialog nach Drücken von *löschen* und aktivierter Klasse?

/TF131/ Erscheint der Löschen-Dialog auch bei nicht aktivierter Klasse?

/TF132/ Verschwindet die gelöschte Klasse aus der Liste und von der Zeichenfläche?

9.2 Testfälle Use-Case

/TF210/ Use-Case hinzufügen
Öffnet sich nach Drücken von *Use-Case hinzufügen*
das Dialogfenster *Use-Case hinzufügen* mit dem Eintrag *unbenannt*?

/TF211/ Öffnet sich nach dem Dialogfenster die Use-Case-Eingabemaske mit dem vorher eingegebenen Namen?

/TF212/ Erscheint der korrekte Use-Case-Name nach Drücken von *fertig* in

	der Use-Case-Liste am linken Bildschirmrand?
/TF213/	Wird der Use-Case nach Anklicken in der Liste korrekt auf der Zeichenfläche rechts dargestellt?
/TF214/	Wird in der Zeichenfläche ein korrekter Use-Case angezeigt nach Anklicken des Use-Case-Reiters?
/TF215/	Lässt sich der Use-Case auf der Zeichenfläche korrekt verschieben?
/TF220/	Use-Case editieren Erscheint nach Drücken von *Use-Case editieren* die Use-Case-Eingabemaske mit den korrekten Daten der jeweiligen Use-Case?
/TF221/	Ändern sich die Use-Case-Anzeige rechts und der Name in der Liste nach dem Editieren?
/TF222/	Wurden die neuen Daten beim Editieren korrekt gespeichert? (nochmaliges Editieren)
/TF230/	Use-Case löschen Erscheint der Löschen-Dialog nach Drücken von *löschen* und aktivierter Use-Case?
/TF231/	Erscheint der Löschen-Dialog auch bei nicht aktivierter Use-Case?
/TF232/	Verschwindet der gelöschte Use-Case aus der Liste und von der Zeichenfläche?

9.3 Testfälle Pflichtenheft

/TF310/	Pflichtenheft editieren Erscheint das Pflichtenheft mit allen Reitern, leeren Einträgen und dem korrekten Projektnamen beim erstmaligen Editieren?
/TF311/	Wurden die Pflichtenhefteinträge korrekt gespeichert, also beim nächsten Editieren korrekt angezeigt?

9.4 Testfälle Projekt

/TF410/	Projekt laden Erscheint der open-Projekt-Dialog mit Verzeichnisstruktur nach Auswahl des Menüpunktes *Projekt laden*?
/TF420/	Projekt speichern Wird das Projekt korrekt gespeichert nach Auswahl des Menüpunktes *Projekt speichern*?
/TF421/	Erscheint der Projekt-speichern-unter-Dialog beim erstmaligen Speichern?
/TF430/	Erscheint der Projekt-speichern-unter-Dialog mit dem korrekten Projektnamen nach Auswahl des Menüpunktes *Projekt speichern unter*?
/TF440/	Projekt schliessen Erscheint die Projekt-speichern-Sicherheitsabfrage nach Auswahl des Menüpunktes *Projekt schliessen*?
/TF441/	Erscheint der Projekt-speichern-unter-Dialog mit dem korrekten Projektnamen nach Drücken von *OK*?
/TF450	Programm beenden Lässt sich das Programm korrekt beenden?
/TF451/	Wurden die erstellten Projekte korrekt gespeichert, also wieder korrekt angezeigt bei der nächsten Programmbenutzung?