

I) Ziele des OOAD-Praktikums

Im Rahmen des OOAD-Praktikums sollen Sie für das Anwendungsszenario „**Terminverwaltung für Besprechungen**“ eine objektorientierte Analyse (OOA) und ein objektorientiertes Design (OOD) sowie eine initiale Programmierung (OOP) durchführen. Der fachliche Inhalt der einzelnen Sitzungen baut aufeinander auf, so dass Sie immer auf demselben Problembereich arbeiten werden. Im Laufe des Praktikums erstellen Sie mit Hilfe des Werkzeugs MagicDraw ein **UML-Gesamtmodell**. Im letzten Praktikum programmieren Sie Teilfunktionen (in C++ v11 mit der NetBeans IDE).

II) Testat und Regeln für das Praktikum

Die Praktikumsaufgaben werden in Gruppen von zwei Teilnehmern bearbeitet und gemeinsam verantwortet. Die erfolgreiche Bearbeitung aller Aufgaben ist Voraussetzung für das Testat von OOAD und somit zur Zulassung zur schriftlichen Prüfung am Ende des Semesters. Bei erfolgreichem Bestehen aller Praktikumsaufgaben wird das Bestehen des Praktikums in OBS vermerkt (siehe OBS-Notenliste).

Bitte beachten Sie die in der OBS-Terminliste eingetragenen Praktikumstermine. Es besteht Anwesenheitspflicht. Unentschuldigtes Fehlen führt unmittelbar zum Nichtbestehen des Praktikums! Atteste müssen spätestens 3 Tage nach dem versäumten Praktikumstermin an den Dozenten per E-Mail (frank.buehler@h-da.de) geschickt werden.

Es wird erwartet, dass Sie die Praktikumsaufgaben **zu Hause vorbereiten** und diese während des Praktikumstermins fertig stellen. Die Modellierungsaufgaben müssen auf Grundlage des **bereitgestellten MagicDraw-Modells** (Template-Modell) erfolgen. Sollten Sie das bereitgestellte Modell nicht verwenden, wird die Abnahme verweigert.

Zum Beginn des Praktikums besteht die Möglichkeit, Fragen zu stellen. Nutzen Sie die Möglichkeit Unklarheiten oder Probleme rechtzeitig zu klären.

Die Abnahme der Praktikumsaufgaben erfolgt ausschließlich auf den Laborrechnern (mit 2 Bildschirmen) des CASE-Labors. Sofern eine Abnahme aus zeitlichen Gründen während des Praktikums nicht möglich ist, schicken Sie das MagicDraw-Modell per E-Mail (frank.buehler@h-da.de) an den Dozenten. Die Abnahme erfolgt dann zum Beginn des nächsten Praktikumstermins auf Grundlage des zugesandten Modells.

Täuschungsversuche führen zum sofortigen Ausschluss aus dem Praktikum.

Sollten bei der Abnahme gravierende Mängel festgestellt werden, führt dies zum Nichtbestehen des Praktikums und Sie erhalten kein Testat. Gravierende Mängel sind beispielsweise:

- **wichtige Anforderungen/Funktionen wurden nicht modelliert**
- **das Praktikumsteam kann das Modellerte nicht erläutern**
- **es können wichtige Fachbegriffe, die in der Vorlesung behandelt wurden, nicht erklärt werden**
- **das entwickelte Programm enthält Compilefehler oder funktioniert nicht**

III) Gliederung des Praktikums

Das OOAD-Praktikum wird im CASE-Labor (D14/211) durchgeführt. Laborleiter ist Dipl. Ing. Michael Guist, der bei technischen Problemen sowie bei Fragen zu MagicDraw zur Verfügung steht. Sein Büro befindet sich im Raum D14/2.09.

Informationen zum CASE-Labor und MagicDraw finden Sie hier:

<https://www.fbi.h-da.de/labore/case.html>

Das CASE-Labor können Sie auch außerhalb der Praktikumszeiten nutzen. Klären Sie vorab, wann das Labor offen ist.

Das OOAD-Praktikum gliedert sich in drei Teile:

Teil 1 (OOA)

1. Termin: Einführung in MagicDraw und die Praktikumsaufgaben, statisches Prototyping (Skizzen für Anwendungsfenster und Dialoge)
- 2./3. Termin: Analyse-Modell: Anwendungsfalldiagramm, Analyse-Klassendiagramm

Teil 2 (OOD)

- 4./5. Termin: Design-Modell: Design-Klassendiagramm, 2 Sequenzdiagramme

Teil3 (OOP)

6. Termin: Testat für C++-Programm

Hinweis: NetBeans-IDE mit C++ Version 11

Nach jedem Praktikumstermin sollten Sie das MagicDraw-Projekt zu ihrem Praktikumpartner (mit Benutzername des ist-Accounts) kopieren.

Hierzu öffnen Sie eine Konsole und wechseln in das MagicDraw-Verzeichnis:

`cd MagicDrawProjekte`

Dann kopieren Sie die mdzip-Datei mittels Secure Copy :

`scp *.mdzip <istAccountName>@userv:`

Danach müssen Sie dem Aufbau der Verbindung zustimmen und das Passwort des ist-Accounts (ihres Praktikumpartners) eingeben.

Das MagicDraw-Modell können Sie auch zu Hause weiter bearbeiten. Hierzu verbinden Sie sich z. B. mittels FileZilla mit dem userv-Server (<istAccountName@userv.fbi.h-da.de>) per sftp. Danach können Sie die Modelldatei in ein lokales Verzeichnis kopieren und bearbeiten. Nach der Bearbeitung des Modells zu Hause können Sie die Modelldatei auch wieder auf den userv zurück kopieren, damit Sie das UML-Modell im CASE-Labor ebenfalls verwenden können.

IV) Fachliche Anforderungen

In diesem Abschnitt finden Sie die *fachlichen Anforderungen* für das OOAD-Projekt beschrieben. Diese sind bewusst an einigen Stellen *ungenau* formuliert, um verschiedene studentische Lösungen zu ermöglichen. Falls Angaben fehlen, sollen Sie sinnvolle Annahmen treffen und diese modellieren.

Als durchgängiges Beispiel für das Praktikum wird ein **Terminverwaltungssystem für Besprechungen** mit dem Namen „**MeetNow**“ verwendet. Das Terminverwaltungssystem „MeetNow“ dient der Organisationsunterstützung von Besprechungen in einem Unternehmen. Das Projektziel ist die Entwicklung einer Anwendung in C++ (Version 11).

Anforderungen

Die Anwendung „MeetNow“ soll die Organisation von Besprechungen erleichtern. Hierzu sollen Räume und Ausstattung sowie die Teilnehmer der Besprechungen verwaltet werden können. Beim Anlegen einer neuen Besprechung ist sicherzustellen, dass es bei teilnehmenden Personen nicht zu zeitlichen Überschneidungen kommt (→ Terminkonfliktprüfung).

Beim Aufruf der Anwendung sollen zunächst initial die verfügbaren Räume (inklusive der zeitlichen Verfügbarkeit) sowie mobile Ausstattungsgegenstände (Beamer, Flipchart, Pinnwand, Schreibutensilien) aus einer Textdatei geladen werden. Ein Besprechungsraum enthält eine Ortsangabe (Gebäude, Stockwerk, Raumnummer) und Informationen zur Größe (Anzahl Sitzplätze) sowie zu festen Ausstattungsmerkmalen (Anzahl Tische, Anzahl Stühle, Anzahl Laptops, Whiteboard: ja/nein, barrierefrei: ja/nein, klimatisiert: ja/nein) und zur zeitlichen Verfügbarkeit. In einem Besprechungsraum stehen entweder ein großer runder Konferenztisch oder mehrere Einzeltische zur Verfügung.

Nutzer des Systems können sich nach Start der Anwendung registrieren. Registrierte Nutzer können sich in dem System mittels Benutzername und Passwort einloggen, um zu erfahren, an welchen Besprechungen sie teilnehmen werden oder um neue Besprechungen anzulegen. Registrierte Nutzer werden automatisch in einer Datei gespeichert.

Eine Besprechung enthält verschiedene Informationen. Für eine Besprechung muss ein Besprechungsthema, der Zeitraum, ein freier Raum (inkl. Ausstattungsgegenstände) sowie die Teilnehmer festgelegt werden. Teilnehmer müssen registrierte Nutzer sein. Bei Bedarf können neue Nutzer beim Einrichten des Termins registriert werden. Registrierte und angemeldete Nutzer können Termine für selbst erstellte Besprechungen verwalten.

Der Administrator des Systems soll nach Start der Anwendung sowohl die Räume als auch die Ausstattungsgegenstände verwalten (d.h. einfügen, ändern und löschen) können.

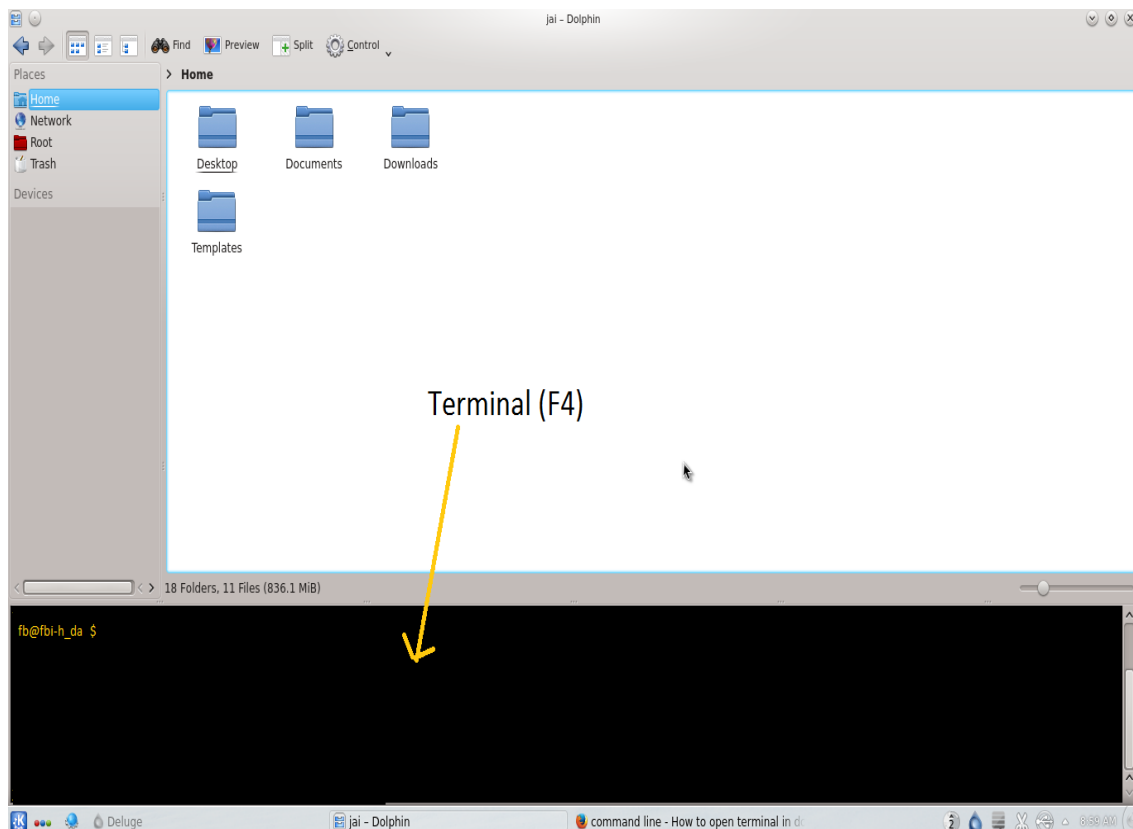
Wenn das System beendet werden soll, muss sich der Nutzer zuvor ausloggen.

V) Projekteinführung - 1. Praktikumstermin (CASE-Labor, Praktikums Umgebung, MagicDraw, Prototyping, ...)

Beim ersten Praktikumstermin erhalten Sie einen Überblick über das CASE-Labor und die Praktikums Umgebung. Insbesondere lernen Sie den Umgang mit dem kommerziellen CASE-Tool *MagicDraw* (siehe <https://www.nomagic.com/products/magicdraw>) kennen. Schließlich wird die Vorgehensweise und Durchführung des Projekts besprochen. Am Ende erstellen Sie Skizzen, die eine mögliche Oberfläche der Anwendung zeigt.

Initialisierung der Praktikums Umgebung

Melden Sie sich mit Ihrem ist-Account auf dem Rechner an. Klicken Sie auf das Symbol  Starten Sie anschließend den File-Explorer Dolphin. Sofern kein schwarzes Terminal-Fenster sichtbar ist, drücken Sie die F4-Taste.



Warten Sie auf weitere Instruktionen von Ihrem Praktikumsbetreuer, um ein **vorkonfiguriertes MagicDraw Modell** zu kopieren und **erste Schritte mit dem Werkzeug MagicDraw** durchzuführen.

Erstellen von GUI-Skizzen für die geplante Anwendung („Oberflächenprototyp“)

Um die Anforderungen an die Terminverwaltung besser verstehen zu können, lesen Sie sich die fachliche Beschreibung (siehe Punkt IV) sorgfältig durch. Skizzieren Sie danach eine Oberfläche für die Anwendung mit den wichtigsten Funktionen (→ statischer Oberflächen-Prototyp mit Anwendungsfenster und wichtigen Dialogen) auf Papier. Diese Vorarbeiten helfen, um anschließend das UML-Analysemodell im 2. und 3. Praktikumstermin zu erstellen. Falls der Platz nicht ausreichend ist, verwenden Sie die Rückseite des Blattes. Die **Abnahme der Skizzen erfolgt am 3. Praktikumstermin.**

VI) Praktikumsaufgaben für OOA - 2. und 3. Praktikumstermin (Analyse-Modell)

Ziele und Aufgaben

Eine zentrale Aufgabe der objektorientierten Software-Entwicklung während der *Analysephase* ist die Darstellung der Anforderungen an das zu entwickelnde System in Form eines UML-Anwendungsfallmodells (plus Anforderungsdokument, Lasten-/Pflichtenheft etc.) sowie die Modellierung eines Domänenmodells in Form von UML-Klassendiagrammen. **Bis zum 3. Praktikumstermin** sollen Sie ein **UML-Analysemodell** (bestehend aus einem Anwendungsfall- und einem Klassendiagramm) erstellen, das die fachlichen Anforderungen für das System berücksichtigt.

Durchführung der Aufgaben

Die objektorientierte Vorgehensweise sieht vor, dass aus den fachlichen Anforderungen zunächst ein **UML-Anwendungsfalldiagramm** erstellt wird. Auf Grundlage der fachlichen Beschreibung sollen Sie zunächst die **Anwendungsfälle** und **Akteure** identifizieren und dann ein möglichst vollständiges UseCase-Modell erstellen. Hierzu lesen Sie sich die fachliche Beschreibung (**siehe IV**) genau durch und notieren sich in der nachfolgenden Tabelle die geforderten Funktionen/Anwendungsfälle und den jeweiligen Akteur, der die Funktionalität benötigt. Hilfreich ist auch der Papier-Oberflächenprototyp vom 1. Termin.

Funktion/Anwendungsfall	Akteur

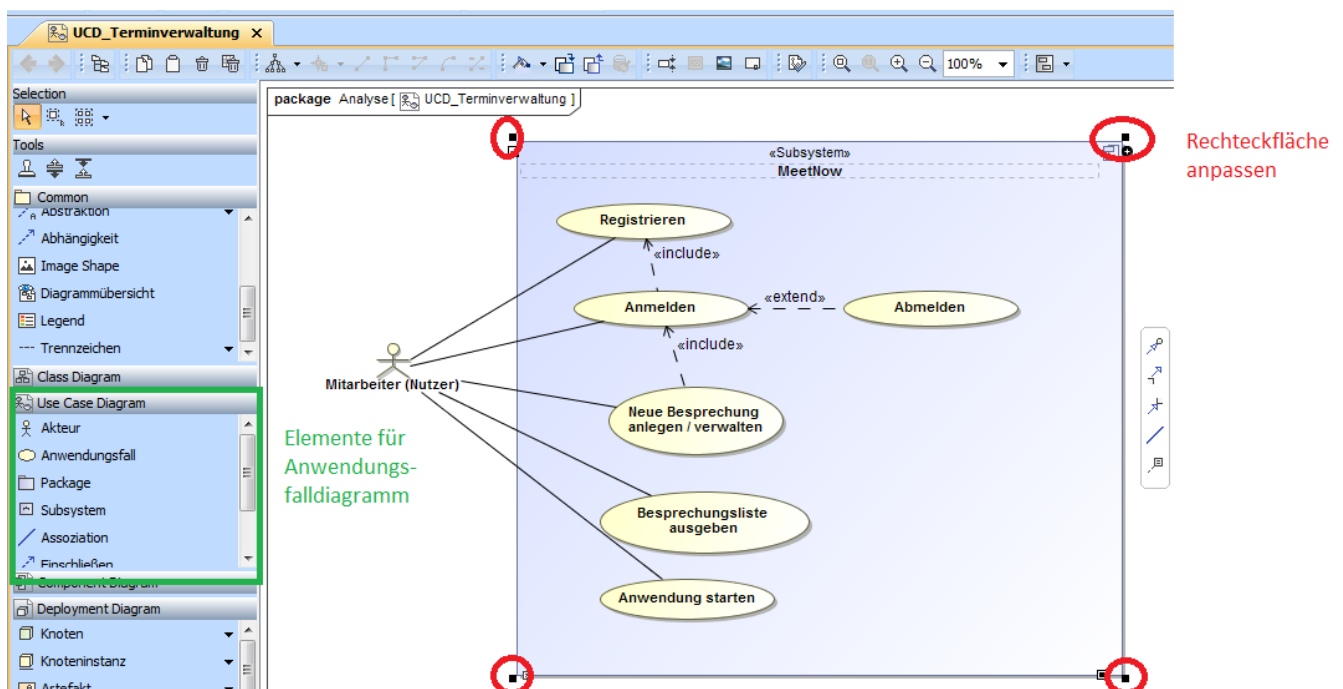
Hinweis: Es sollen 10 bis 15 Anwendungsfälle und 2 Akteure identifiziert werden.

Nun können Sie in MagicDraw ein Anwendungsfalldiagramm mit dem Namen „UCD_Terminverwaltung“ im Paket „Terminverwaltung\Analyse“ anlegen.

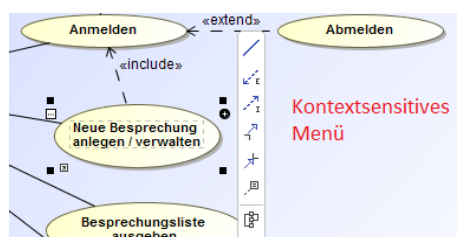
In der Toolbar selektieren Sie zunächst das Element „Subsystem“ und ziehen dieses auf die Arbeitsfläche. Mit der rechten Maustaste klicken Sie auf das Element und öffnen das Spezifikationsfenster über „Spezifikation“ im Kontextmenü. In dem Feld „Name“ des Dialogs können Sie den Namen des Systems „MeetNow“ eintragen. Mit der Schaltfläche „Schließen“ wird der Dialog beendet.

Als nächstes fügen Sie die Anwendungsfälle ein. Hierzu wählen Sie das Element „Anwendungsfall“ aus und ziehen dieses in den Bereich des Subsystems. Sollte die Rechteckfläche des Subsystems zu klein sein, dann können Sie diese über die schwarzen, kleinen Begrenzern an den Ecken des Symbols vergrößern (bzw. bei Bedarf verkleinern).

Schließlich können Sie die Akteure einfügen und diese mit den Hauptanwendungsfällen verbinden. Anschließend verbinden Sie die Anwendungsfälle über include-/extend-Beziehungen. Achten Sie hierbei darauf, dass include-Beziehungen eine starke funktionale Abhängigkeit darstellt, die zwingend erforderlich ist und extend-Beziehungen funktionale Erweiterungen darstellen.



Hinweis: wenn Sie auf ein Anwendungsfallsystem klicken, erscheint ein kontextsensitives Menü.



Wenn Sie den Pfeil mit der Beschriftung „E“ oder „I“ auswählen, können Sie sehr einfach Extend- bzw. Include-Beziehungen festlegen, indem Sie die erscheinende gestrichelte Linie auf ein anderes Anwendungsfallsymbol ziehen und dann auf das Zielsymbol klicken.

Auf Grundlage der fachlichen Anforderungen und des erstellten Anwendungsfalldiagramms können jetzt die *Klassenkandidaten* identifiziert werden. Hierzu kann insbesondere die Technik der grammatikalischen Inspektion (→ Substantivmethode) angewandt werden.

Notieren Sie in der nachfolgenden Tabelle die möglichen Analyseklassen inkl. 2 Attributen.

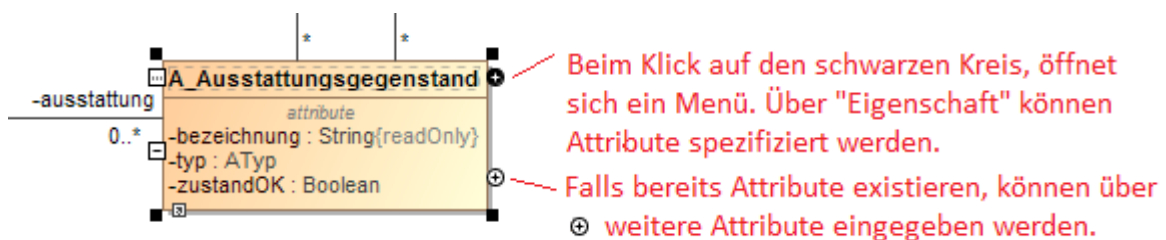
Analyseklasse (Kandidat)	Attribut 1	Attribut 2

Hinweis: es sollen 6 bis 10 Analyseklassen gefunden und dann modelliert werden.

Im nächsten Schritt erstellen Sie ein Klassendiagramm mit dem Namen „AKD_Terminverwaltung“ im Paket „Analyse“. Ihre Aufgabe ist die Erstellung eines möglichst vollständigen und konsistenten Analyseklassendiagramms als initiales Domänenmodell (= deskriptives Modell für die Anforderungsanalyse).

Anschließend fügen Sie die Klassen in den Arbeitsbereich ein. Wählen Sie hierzu das Symbol „Klasse“ und ziehen dieses in den Arbeitsbereich. Alle Analyseklassennamen im Klassendiagramm sollen mit dem Präfix „A_“ beginnen.

Um Attribute einzugeben, klicken Sie auf den schwarzen Kreis,

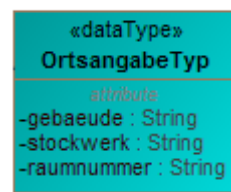
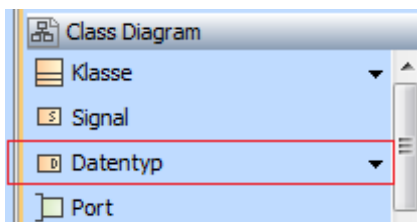


Achten Sie bei der Eingabe der Attribute auf folgende Syntax:

- <Attributname> : <Datentyp>

Folgende Datentypen können u.a. verwendet werden: Integer, Double, Boolean, String, Date, Time.







Eigene Datentypen (z. B. ATyp) können ebenfalls definiert werden. Hierzu muss in der Auswahlbox das Element „Datentyp“ ausgewählt werden. Anschließend können Datentyp-Klassen in den Arbeitsbereich eingefügt und über eine uses-Beziehung verbunden werden.



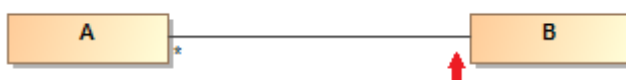
Enumeration-Klassen befinden sich in dem gleichen Auswahlbereich wie Datentyp-Klassen.

Anschließend werden die gefundenen Klassen zur Konstruktion eines möglichst aussagefähigen *UML-Analyseklassendiagramms* verwendet. Hierbei können evtl. Klassen zu Attributen umgewandelt werden oder neue Attribute und Klassenbeziehungen gefunden und modelliert werden.

Schließlich müssen die Klassen über Assoziationen miteinander verbunden werden. Um eine Assoziation auszuwählen, klickt man zunächst auf das Klassensymbol und wählt dann eine mögliche Assoziationsform aus. Folgende Beziehungen sind im Analyseklassendiagramm erlaubt:

-  Generalisierung
-  Spezialisierung
-  uses-Beziehung
-  einfache Assoziationsbeziehung
-  Aggregationsbeziehung
-  Kompositionsbeziehung

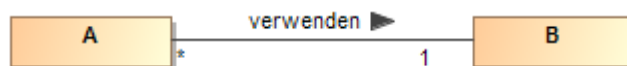
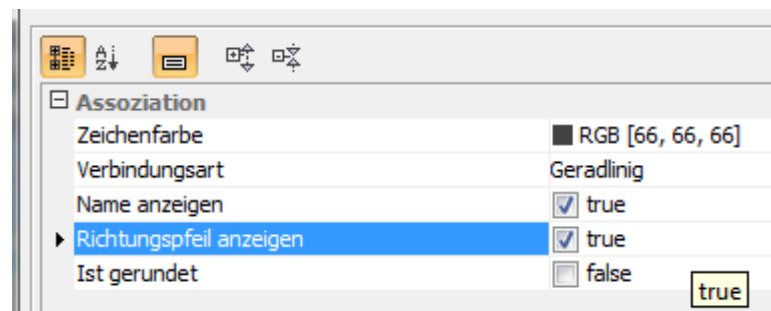
Um die Multiplizitäten festzulegen, klickt man mit der rechten Maustaste auf das Assoziationsende.



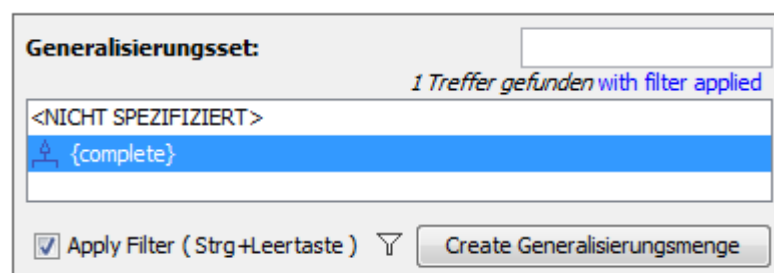
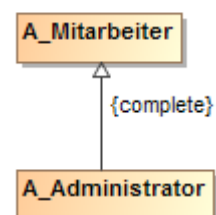
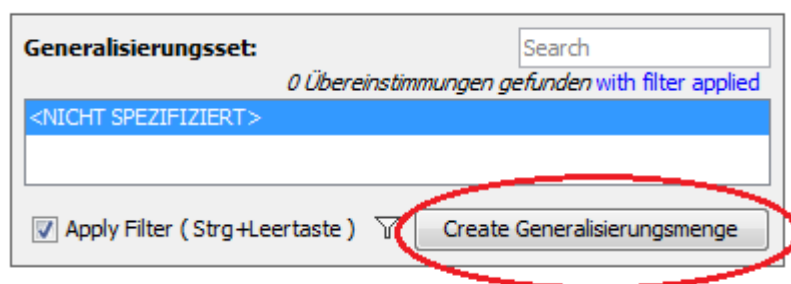
Mit der rechten Maustaste
hier klicken. Im Kontextmenü
können die verschiedenen
Multiplizitäten (0, 1, *,...)
ausgewählt werden.

Um den Assoziationsnamen einzugeben, öffnet man das Spezifikationsfenster durch einen Doppelklick auf die Linie und gibt im Dialog im Feld „Name“ den Namen der Assoziation ein.

Um die Leserichtung anzuzeigen, öffnet man den Symboleigenschaften-Dialog (mit Klick der rechten Maustaste auf die Linie und Auswahl des Punktes „Symboleigenschaften“). Die Option „Richtungspfeil anzeigen“ muss auf „true“ gesetzt werden. Nach Schließen des Dialogs wird ein schwarzer Pfeil dargestellt.



Um ein Constraint für die Vererbungsbeziehung anzugeben, wählt man die Generalisierungsbeziehung aus und wählt im Kontextmenü „Generalisierungsset“ aus. Über „Create Generalisierungsmenge“ können entsprechende Constraints angelegt werden.



Weitere Informationen zur Nutzung von MagicDraw finden Sie hier:

<https://www.fbi.h-da.de/labore/case/laborausstattung/magicdraw/magicdraw-zu-hause.html>

Abnahme des OOA-Modells

Bei der Erstellung des Anwendungsfalldiagramms ist auf die korrekte Verwendung der include- und extend-Beziehungen zu achten. Akteure sollen mit den Hauptanwendungsfällen verbunden sein. Beachten Sie auch die „Handfeste Regeln zum Anwendungsfalldiagramm“.

Für das Analyse-Klassendiagramm müssen alle zentralen Entity-Klassen (inkl. der wichtigsten Attribute) im Paket „Analyse“ spezifiziert werden. Die Attribute müssen „sprechende“ Namen aufweisen. Die Angabe von Datentypen ist optional. Sofern Listen-Klassen modelliert werden, müssen diese Methoden enthalten.

Des Weiteren sind die Assoziationsbeziehungen festzulegen. Es dürfen keine gerichtete Assoziationen verwendet werden. Für jede Assoziation sind der Assoziationsname, die Leserichtung und die Multiplizitäten anzugeben. Rollennamen sind optional. Bei Vererbungsbeziehungen sind die entsprechenden „Constraints“ (Einschränkungen) „overlapping/disjoint“, „in/complete“ zu verwenden. Ansonsten sind die „Handfeste Regeln für das Analyseklassendiagramm“ zu beachten.

Die Abnahme des OOA-Modells **zum 3. Praktikumstermin** umfasst somit folgende Abnahmepunkte:

Nr.	Abnahmepunkt
1	Das UML-Analysemodell wurde auf Grundlage des vorkonfigurierten UML-Modells erstellt. Die Diagramme befinden sich im Analyse-Paket.
2	Skizzen für Oberflächenprototyp sind erstellt.
3	Tabellen mit Anwendungsfällen (10-15 Anwendungsfälle und 2 Akteuren) und Analyseklassen (6 bis 10 Analyseklassen mit jeweils 2 Attributen) sind ausgefüllt.
4	Vollständiges Anwendungsfalldiagramm (mit allen geforderten Anwendungsfälle sowie Akteure, Modellierungsregeln sind eingehalten). Das Anwendungsfalldiagramm kann von beiden Praktikumssteilnehmern unter Verwendung der Fachsprache erläutert werden. Getroffene Entscheidungen bei der Modellierung sollen begründet werden.
5	Vollständiges und korrektes Analyseklassendiagramm (mit allen geforderten Klassen sowie hinreichender Spezifikation der Attribute = aussagekräftige Attributnamen und korrekt und vollständig spezifizierte Assoziationen, die Modellierungsregeln sind eingehalten). Das Anwendungsfalldiagramm kann von beiden Praktikumssteilnehmern unter Verwendung der Fachsprache erläutert werden. Getroffene Entscheidungen bei der Modellierung sollen begründet werden.

VII) Praktikumsaufgaben für OOD - 4. und 5. Praktikumstermin (Design-Modell)

Ziele und Aufgaben

Bisher haben Sie ein Analyse-Modell erstellt. Nun sollen Sie ein Designmodell (als Teil-Modell des UML-Gesamtmodells) erstellen. Grundlage für das Designmodell sind die in der ersten Praktikumsaufgabe entwickelten Anwendungsfall- und Analyse-Klassendiagramme. Das Design-Modell muss die Designklassen enthalten, um alle fachlichen Anforderungen erfüllen zu können. Es muss so modelliert werden, dass hieraus die Anwendung entwickelt werden kann (-> Implementations-Spezifikation).

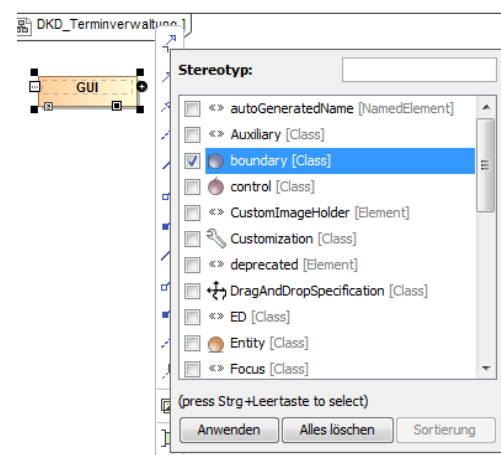
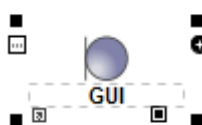
Im **Design-Klassendiagramm** sind bei den Designklassen nun die Sichtbarkeit und die Datentypen der Attribute sowie die Methoden mit den erforderlichen Parametern anzugeben. Pro Klasse sind mindestens drei Attribute zu modellieren. Es dürfen nur gerichtete Assoziationen verwendet werden. Für jede gerichtete Assoziation sind die Multiplizitäten anzugeben. Rollennamen sind anzuzeigen. Bei Vererbungsbeziehungen sind die entsprechenden „Constraints“ (Einschränkungen) „*overlapping/disjoint*“, „*in/complete*“ zu verwenden. Des weiteren sollen die Regeln guten objektorientierten Designs befolgt werden und auch notwendige Controller-Klassen verwendet werden. Auch sind die „Handfeste Regeln für das Design-Klassendiagramm“ einzuhalten.

Schließlich sollen **zwei Sequenzdiagramme** modelliert werden. Das erste Sequenzdiagramm soll den Ablauf der Initialisierung der Anwendung darstellen. Das zweite Sequenzdiagramm soll den vollständigen Ablauf „Anlegen eines Besprechungstermins“ spezifizieren. Dieser Ablauf zeigt u.a. das Einloggen eines Nutzers und das Erstellen eines Termins inklusive Auswahl der Teilnehmer und eines Raumes. Zum Schluss soll geprüft werden, ob es Termin-überschneidungen bei den eingeladenen Teilnehmern gibt.

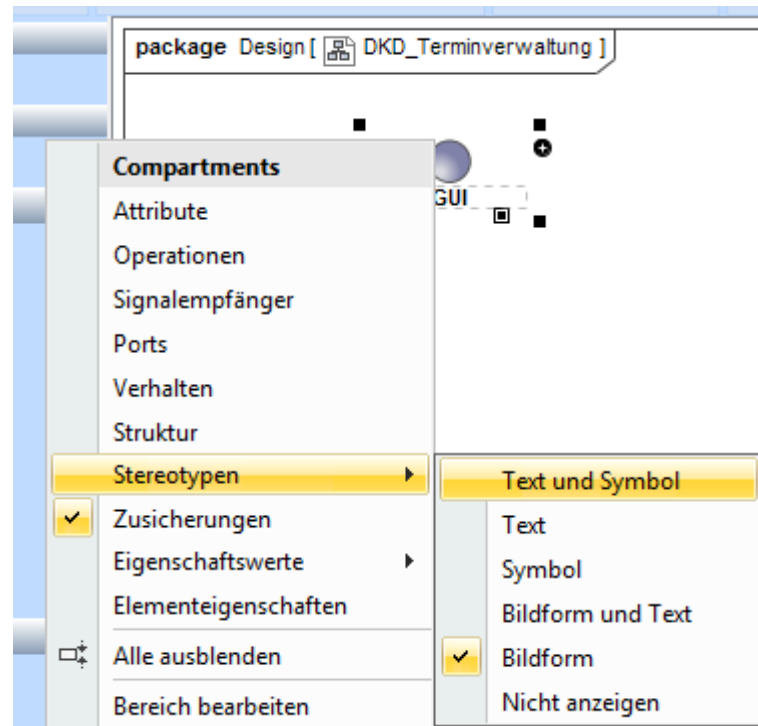
Durchführung der Aufgaben

Zunächst wird in dem Paket „Design“ das Klassendiagramm mit dem Namen „DKD_Terminverwaltung“ angelegt. Als nächstes fügen Sie die Klasse „GUI“ in das Diagramm ein. Nun legen Sie den Stereotyp für die Klasse fest, indem Sie im Kontextmenü auf „Stereotyp“ klicken und den Stereotyp „boundary“ auswählen und auf anwenden klicken.

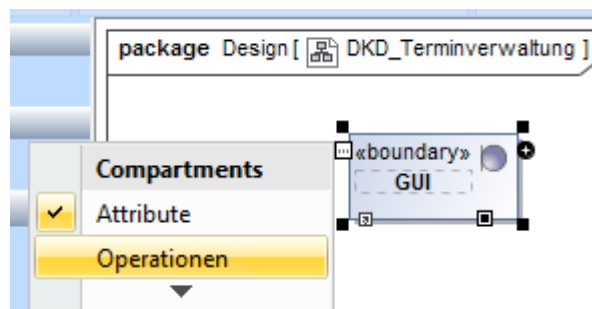
Nun ändert sich die Darstellung der Klasse.



Als nächstes wählen Sie die Klasse „GUI“ aus und klicken auf das Kästchen mit den drei Punkten (→ Compartments). Wählen Sie dann beim Punkt „Stereotypen“ die Anzeigeeoption „Text und Symbol“ aus. Nun wird das Klassensymbol wieder auf die ursprüngliche Anzeige gesetzt.

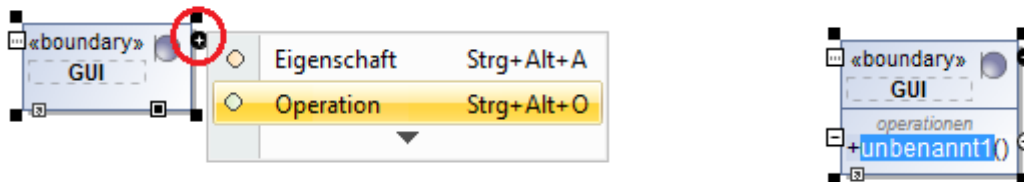


Wählen Sie auch die Anzeigeeoptionen für Attribute und Operationen aus, damit alle spezifizierten Attributen und Methoden sichtbar sind.



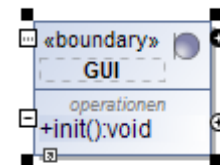
Nun können Sie die Methoden für die Klasse „GUI“ festlegen.

Klicken Sie auf den schwarzen Kreis (rechts oben beim Klassensymbol) und wählen Sie den Punkt „Operationen“. Es wird eine Methode automatisch eingefügt, die nun bearbeitet werden kann.

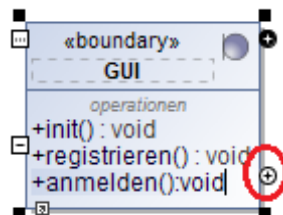


Geben Sie den Methodennamen „init“ ein. Die Syntax für eine öffentliche Methode lautet wie folgt:

+ <Methodenname> : (<Datentyp> | void)

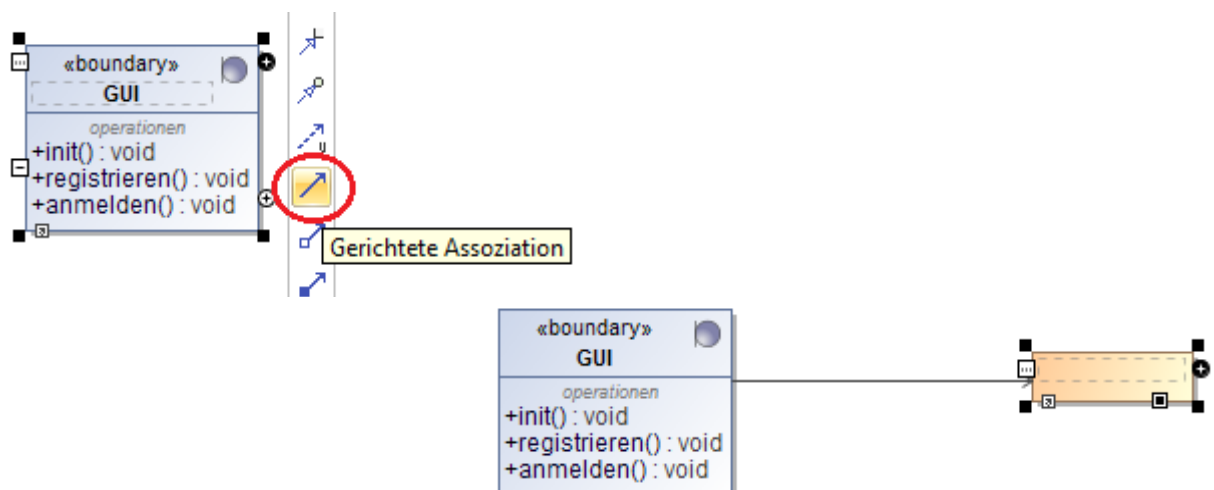


Geben Sie weitere Methoden ein, indem Sie auf den weißen Kreis klicken.

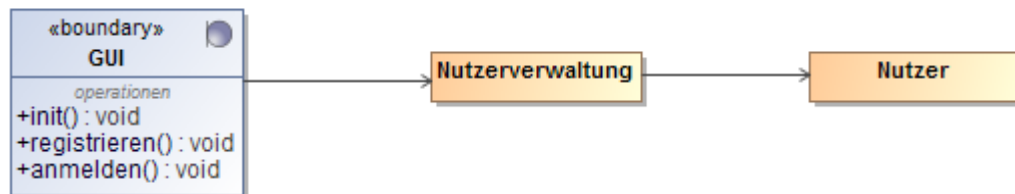


Um Nutzer verwalten zu können, benötigen wir eine Control-Klasse mit dem Namen „Nutzerverwaltung“ und eine Entity-Klasse „Nutzer“.

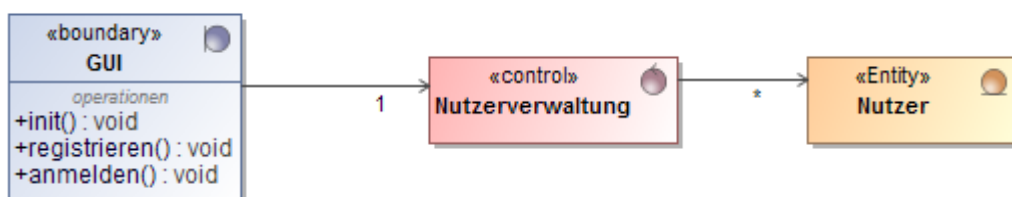
Wählen Sie im Kontextmenü den Pfeil mit der gerichteten Assoziation aus und ziehen Sie den Pfeil an die Stelle, an der Sie die neue Klasse platzieren wollen. Geben Sie nun den Klassennamen „Nutzerverwaltung“ ein.



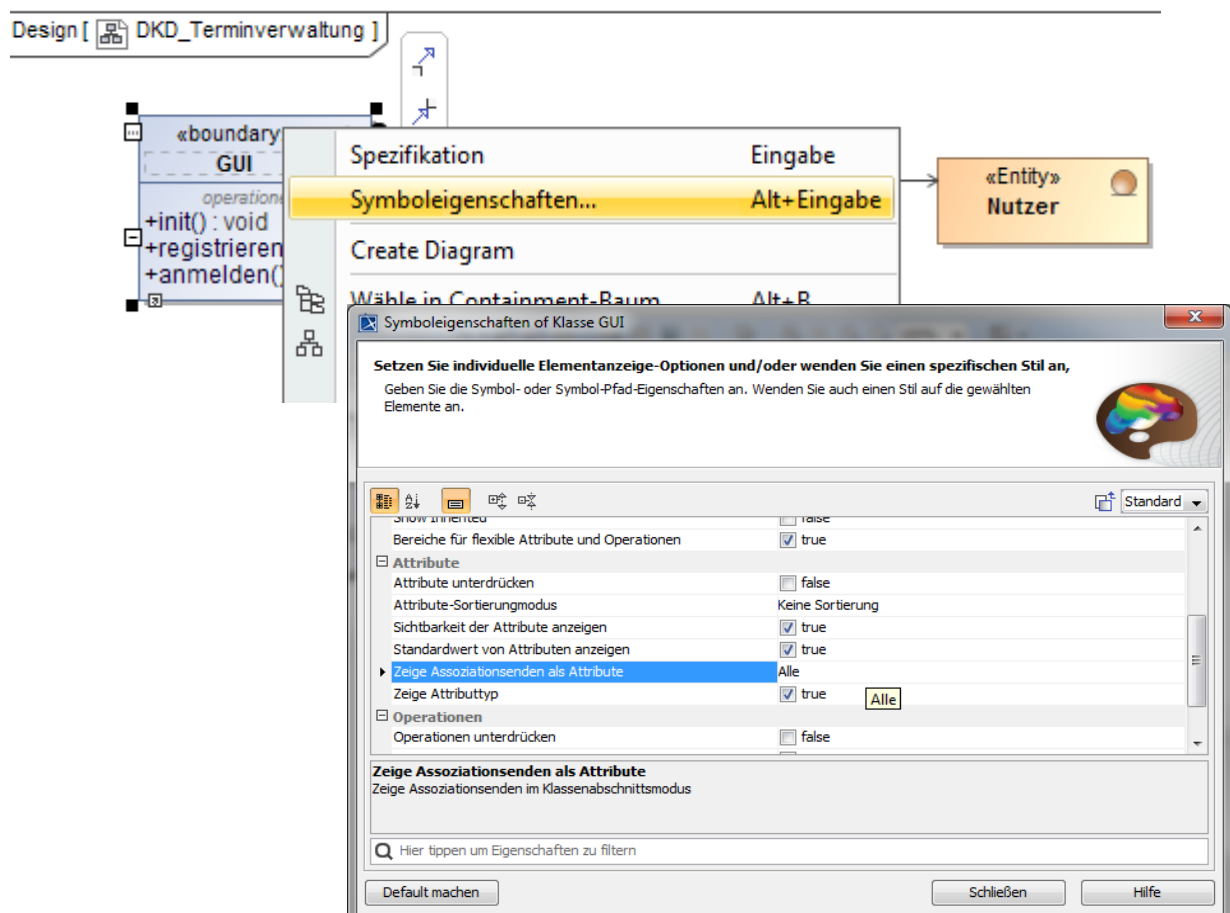
Analog gehen Sie bei der Erstellung der Klasse „Nutzer“ vor.



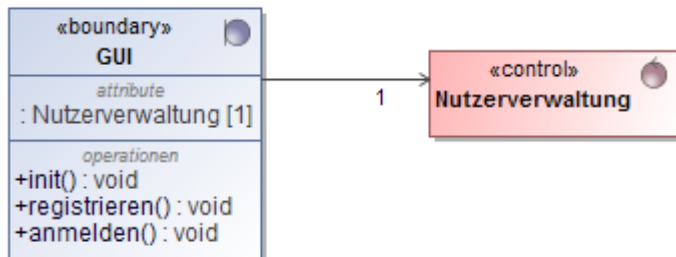
Im nächsten Schritt werden die Multiplizitäten (→ rechter Mausklick am Ende der Assoziation) und Stereotypen (Punkt „Stereotyp“ im Kontextmenü) festgelegt.



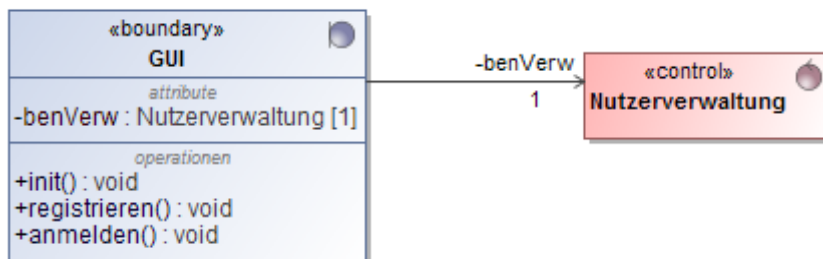
Schließlich sollen die Assoziationen als Attribute angezeigt werden. Klicken Sie auf den Punkt „Symboleigenschaften“ im Kontextmenü und setzen Sie den Wert von „Zeige Assoziationsenden als Attribute“ auf „Alle“.



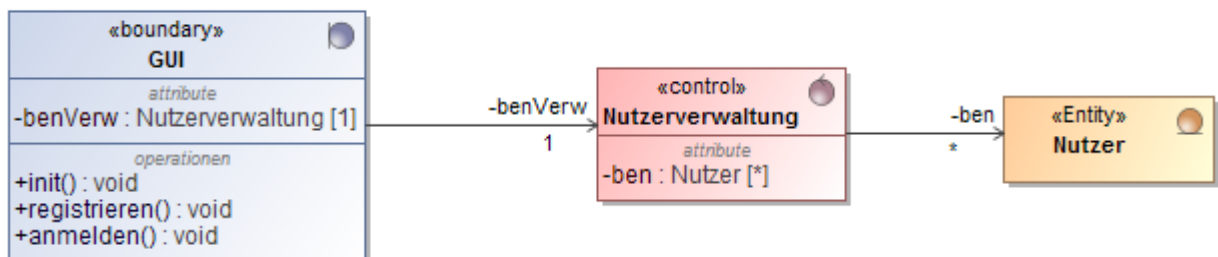
Nach dem Schließen des Dialogs wird ein Attribut für die Beziehung angezeigt.



Geben Sie noch einen Attributnamen ein und legen die Sichtbarkeit fest. Nun erscheint der Attributname als Rollenname am Ende der Beziehung.

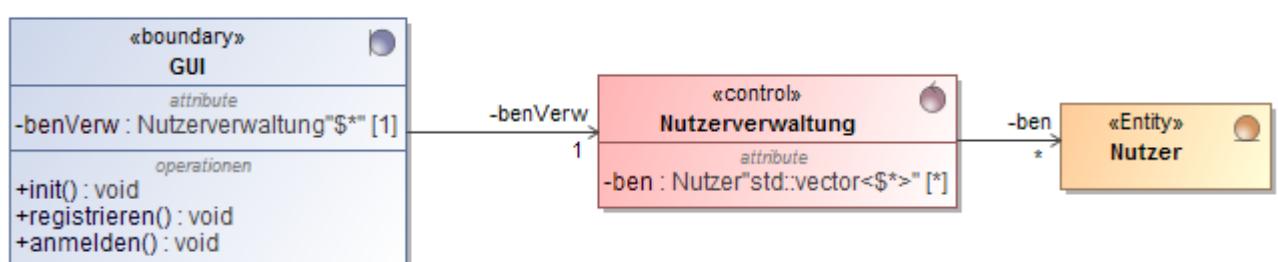


Analog gehen Sie bei der Beziehung zwischen der Klasse „Nutzerverwaltung“ und „Nutzer“ vor.



Um aus dem Klassendiagramm C++-Code erzeugen zu können, werden noch die zu erzeugenden Datentypen eingegeben. Öffnen Sie mit einem Doppelklick auf das Attribut „benVerw“ das Spezifikationsfenster.

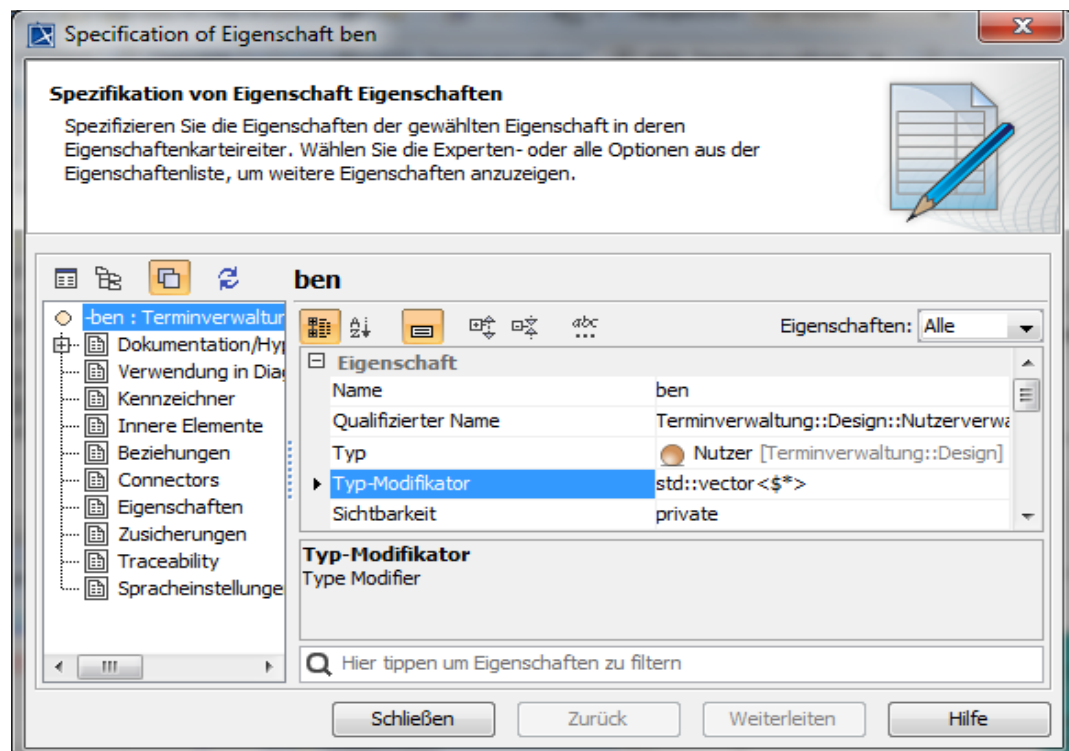
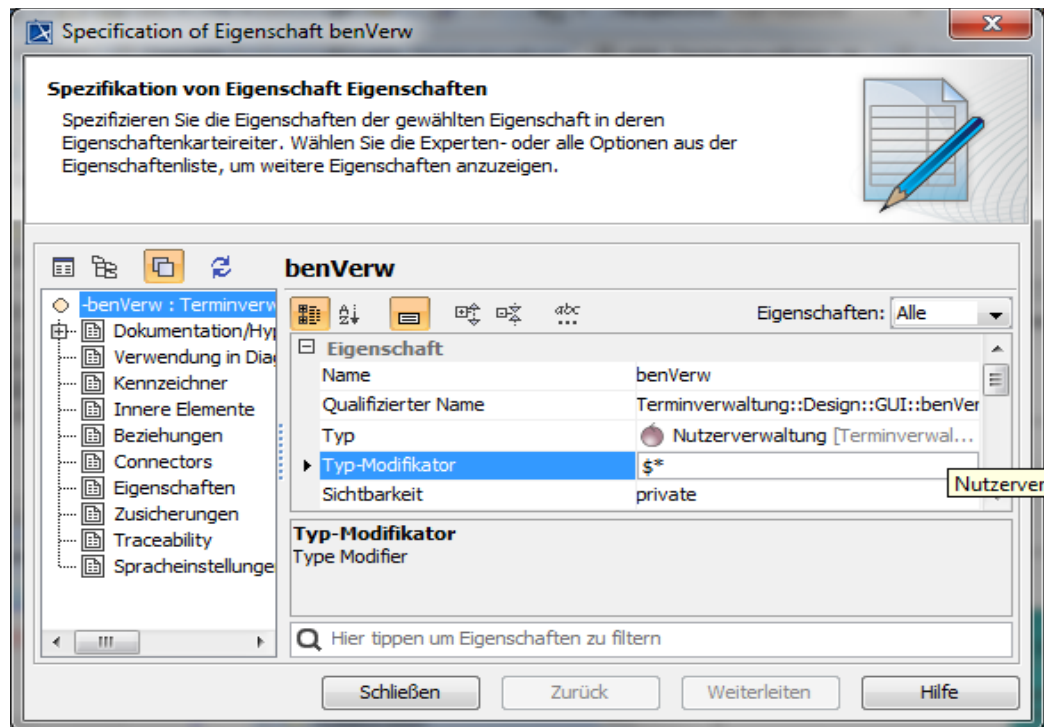
Geben Sie beim Attribut „benVerw“ als TypModifizierer für C++ „\$*“ ein. Beim Attribut „ben“ sollte als TypModifizierer für C++ „std::vector<\$*>“ eingegeben werden.



Weitere Informationen zur Codeerzeugung mit MagicDraw finden Sie hier:

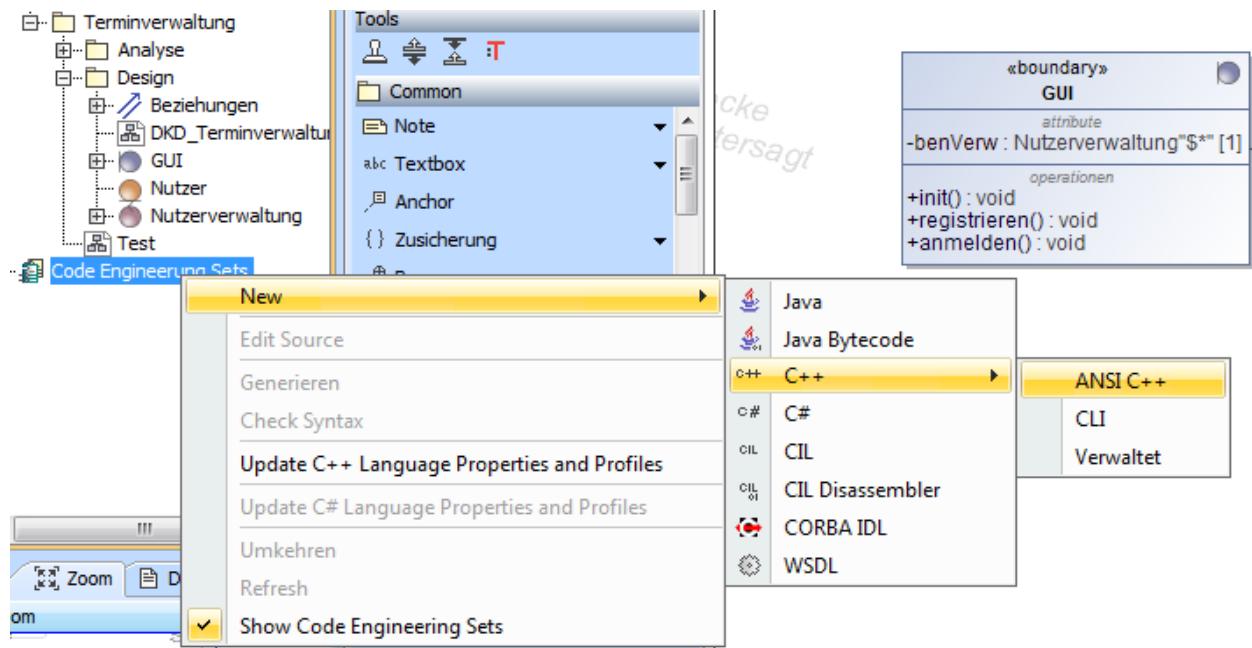
<https://www.nomagic.com/files/manuals/MagicDraw%20CodeEngineering%20UserGuide.pdf>

Spezifikation für C++

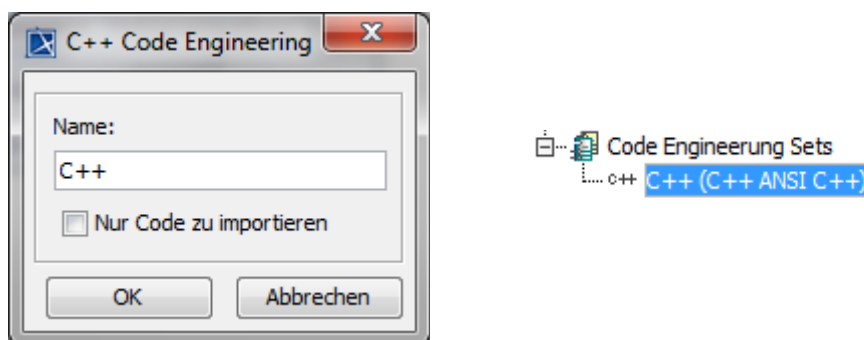


Nun kann Code aus dem Design-Klassendiagramm erzeugt werden. Hierzu legen wir im Containment-Bereich ein Code Engineering-Set für C++ an. Nachfolgend werden die notwendigen Schritte dargestellt.

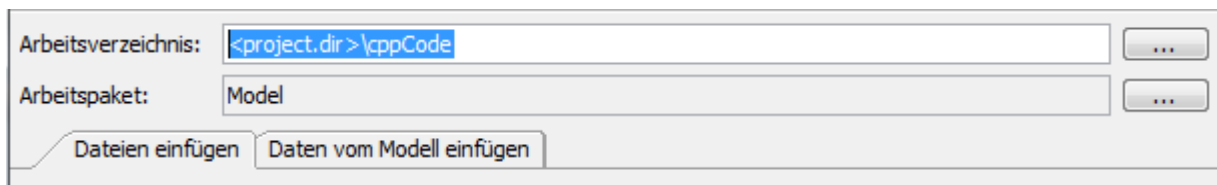
Klicken Sie mit der rechten Maustaste auf „Code Engineering Set“ und wählen New\C++\ANSI C++ aus.



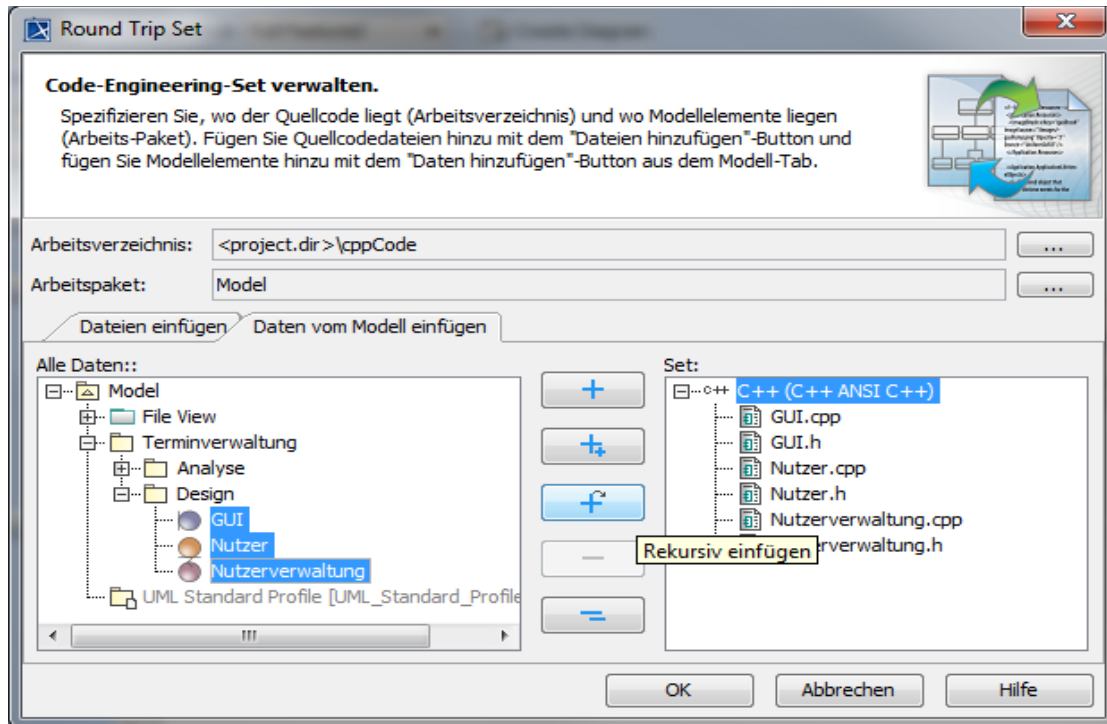
Geben Sie dann „C++“ als Namen für das Code Engineering Set ein und klicken auf „OK“. Nun sehen Sie, dass das neue Set angelegt wurde.



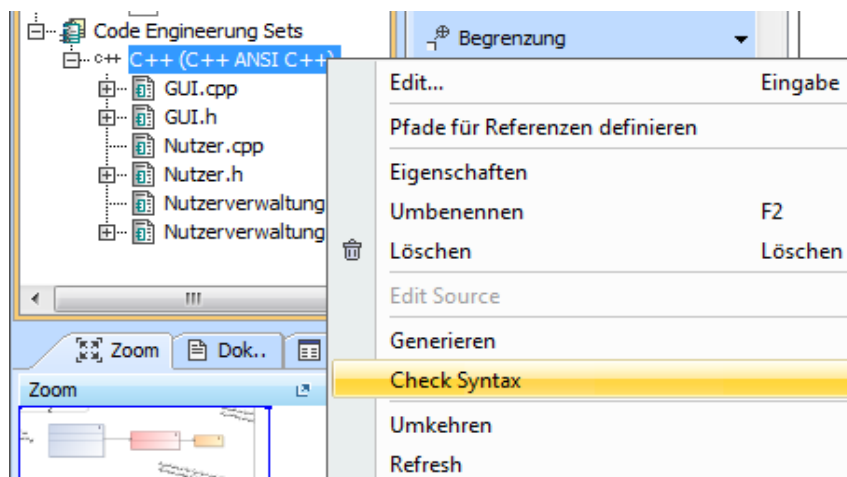
Dieses muss nun konfiguriert werden. Hierzu doppelklicken Sie auf „C++“. Legen Sie anschließend das Arbeitsverzeichnis fest, in das die Dateien erzeugt werden.



Klicken Sie nun auf den Reiter „Daten vom Modell“ einfügen. Fügen Sie aus dem Paket „Design“ alle Klassen ein und klicken auf „OK“.



Im nächsten Schritt muss auf Einhaltung der Syntax geprüft werden. Wählen Sie das C++ Code Engineering Set aus und wählen Sie dann den Punkt „Check Syntax“ im Kontextmenü. Wenn keine Fehler im Modell enthalten sind, dann können Sie mittels „Generieren“ die C++ Codedateien erstellen.



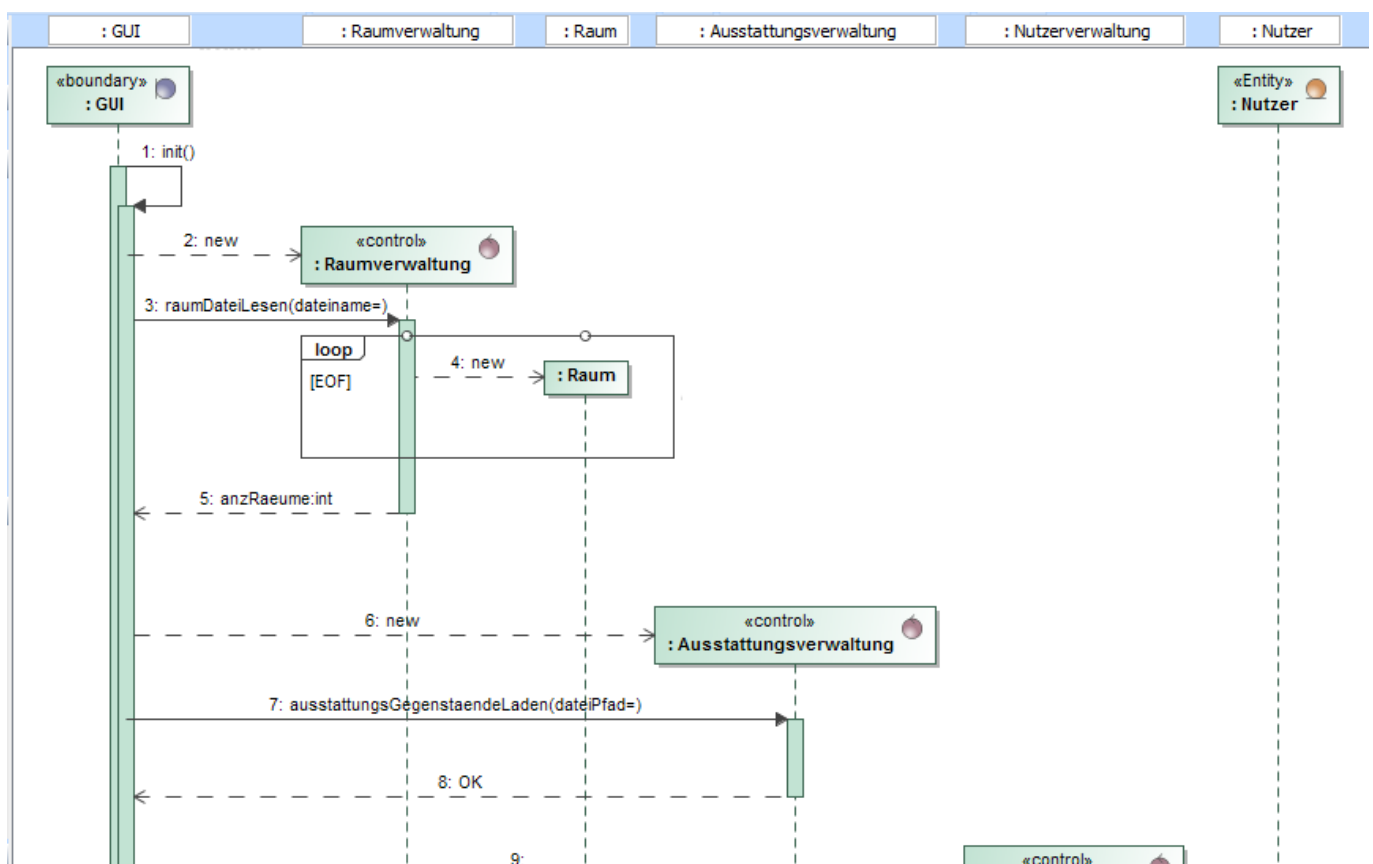
Nachdem die Funktionalität für das Registrieren und Anmelden realisiert sind, können nun die **restlichen Anwendungsfälle** betrachtet und spezifiziert werden. Achten Sie beim Modellieren darauf, dass Sie die verschiedenen Funktionen jeweils startend von der Klasse „GUI“ umsetzen und dass **Ein- und Ausgaben** ausschließlich in dieser Boundary-Klasse spezifiziert werden.

Nun können die **Sequenzdiagramme** realisiert werden.

Erstellen Sie zunächst im Design-Paket die Sequenzdiagramme „**SD_Terminverwaltung_init**“ und „**SD_Terminverwaltung_Besprechung_erstellen**“. Wählen Sie anschließend das Sequenzdiagramm für die Initialisierung der Terminverwaltung aus. Folgende Funktionen müssen modelliert werden:

- Initialisierung der Raumverwaltung (inkl. Laden der Räume)
- Laden der Ausstattungsgegenstände
- Initialisierung der Nutzerverwaltung (inkl. Anlegen eines Administrators)
- Initialisierung der Besprechungsverwaltung

Beim Modellieren gehen wir davon aus, dass ein GUI-Objekt bereits erzeugt wird und die Methode init() für die Initialisierung der kompletten Anwendung notwendig ist. Nachfolgend finden Sie einen Ausschnitt für das zu erstellende Sequenzdiagramm.

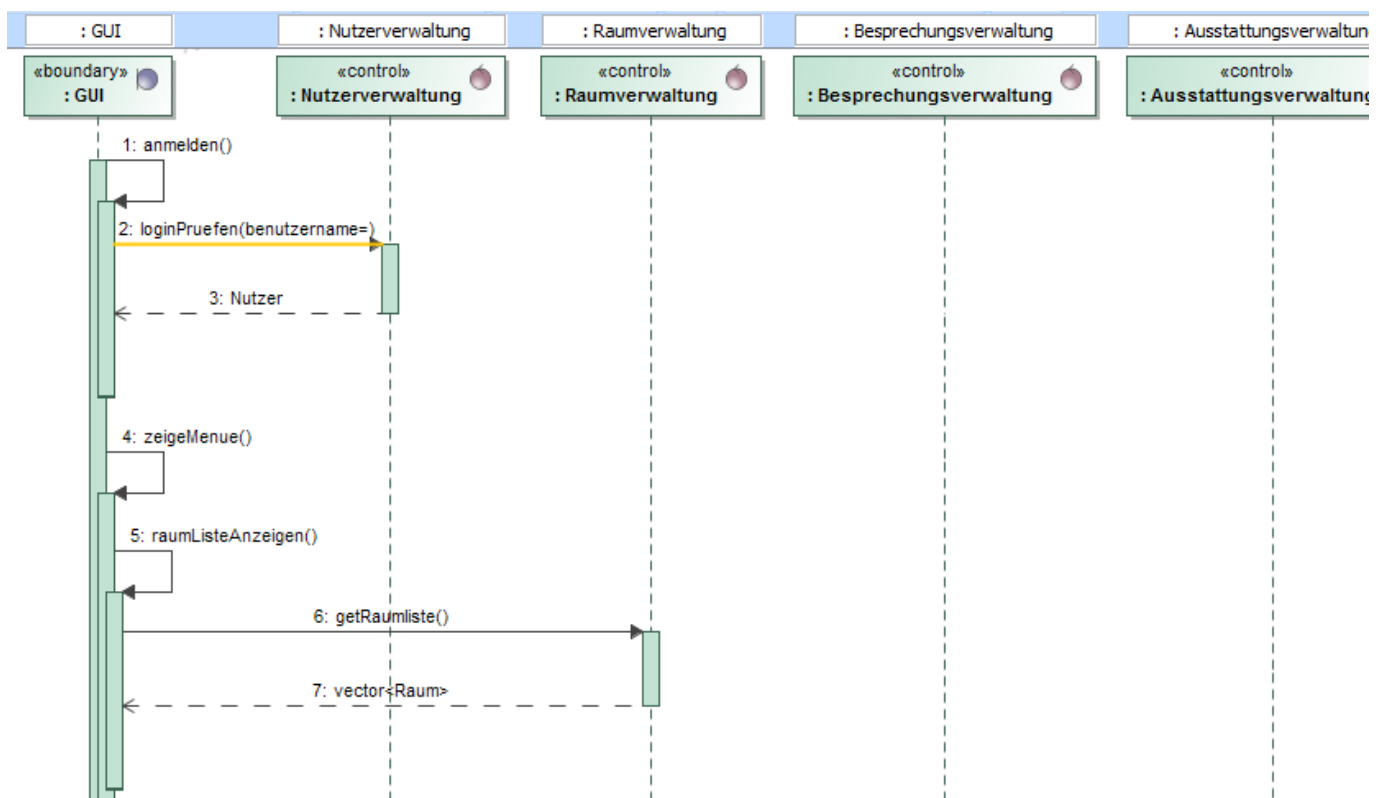


Im nächsten Schritt erstellen Sie das Sequendiagramm für das Anlegen einer Besprechung.

Folgende Funktionen müssen modelliert werden:

- Anmelden eines Nutzers
- Menü zur Anzeige und Auswahl von Funktionen
- Anzeige der vorhandenen Räume
- Registrieren von weiteren Nutzern (potentielle Teilnehmer einer Besprechung)
- Erstellen einer Besprechung (inkl. Zuordnung eines freien Raumes und Zuordnung von Teilnehmern sowie eines Beamers) zu der Besprechung
- Terminkonflikt für eine Besprechung prüfen

Nachfolgend finden Sie einen Ausschnitt für das zu erstellende Sequenzdiagramm. Verwenden Sie grundsätzlich nur synchrone Nachrichten. Achten Sie bei der Modellierung auch auf die korrekte Verwendung von Rückgabemessages. Die Modellierung der notwendigen Set-Nachrichten ist optional, um das Sequenzdiagramm im Umfang zu begrenzen.



Die Abnahme des Design-Modell umfasst folgende Abnahmepunkte:

Nr.	Abnahmepunkt
1	Das UML-Analysediagramm wurde verwendet, um das UML-Designklassendiagramm zu spezifizieren. (Es soll gezeigt werden, welche Klassen übernommen wurden.)
2	<p>Das UML-Designklassendiagramm ist vollständig und korrekt. Es berücksichtigt alle geforderten Funktionen. Wichtige Attribute und Methoden sind spezifiziert. Das Klassendiagramm erfüllt die Regeln guten Designs. Es kann als Implementations-Spezifikation (= präskriptives Modell) für eine C++-Anwendung verwendet werden.</p> <p>Das Klassendiagramm kann von beiden Praktikumssteilnehmern unter Verwendung der Fachsprache erläutert werden. Getroffene Entscheidungen bei der Modellierung sollen begründet werden.</p>
3	<p>Die Sequenzdiagramme zeigen den Ablauf für die Funktionen „Initialisierung der Anwendung“ und „Besprechung anlegen“.</p> <p>Die Sequenzdiagramme können von beiden Praktikumssteilnehmern unter Verwendung der Fachsprache erläutert werden. Getroffene Entscheidungen bei der Modellierung sollen begründet werden.</p>

VIII) Praktikumsaufgabe für OOP - 6. Praktikumstermin (Implementierung in C++ v11)

Ziele und Aufgaben

Nachdem Sie nun ein Design-Modell erstellt haben, geht es an die Realisierung der Anwendung. Hierzu soll die NetBeans-Plattform für die Implementierung des Programms verwendet. Aus Zeitgründen kann allerdings nur ein kleiner Ausschnitt der Funktionalität gemäß dem Designmodell realisiert werden.

Folgende Funktionen sollen in C++ realisiert werden:

- Anmelden eines Nutzers
- Menü zur Anzeige und Auswahl von Funktionen
- Anzeige der vorhandenen Räume (inkl. initiales Einlesen über Datei)
- Registrieren von weiteren Nutzern (potentielle Teilnehmer einer Besprechung)
- Erstellen einer Besprechung (inkl. Zuordnung eines freien Raumes und Zuordnung von Teilnehmern sowie eines Beamers) zu der Besprechung
- Terminkonflikt für eine Besprechung prüfen (optional)
- Abmelden eines Nutzers
- Anwendung schließen

Nehmen Sie als Basis für die Implementierung das von Ihnen erstellte Design-Klassendiagramm sowie die Sequenzdiagramme (-> Implementations-Spezifikation). Führen Sie zunächst einen **Code-Engineering-Schritt**. Entwickeln Sie anschließend ein gut strukturiertes und ausführlich kommentiertes C++-Programm. Passen Sie ihr Design-Modell an die Implementierung an. Führen Sie hierzu am Ende ein **Reverse Engineering-Schritt** durch. Am Ende der Entwicklung soll das Design-Klassendiagramm sowie die Sequenzdiagramme ihre Implementierung korrekt widerspiegeln.

Um den dritten Teil des Praktikums erfolgreich zu bestehen und somit das Testat für das OOAD-Praktikum zu erhalten, sind folgende Bedingungen zu erfüllen:

Nr.	Abnahmepunkt
1	Funktionsfähige Implementierung der geforderten Funktionen (keine Compiler- und Runtime-Fehler). Das C++-Programm muss auf einem Laborrechner unter NetBeans compilierbar und ausführbar sein und realisiert alle geforderten Funktionen. Des Weiteren muss der Source-Code ausreichend kommentiert sein. Die Diagramme müssen der Implementierung entsprechen. Hinweis: Beachten Sie, dass auf den Laborrechnern C++ in der Version 11 vorhanden ist.
2	Die UML-Diagramme aus Aufgabe 1 (OOA) und 2 (OOD) sind unverändert im UML-Modell vorhanden.

Nach erfolgreicher Abnahme erfolgt die Testierung des gesamten OOAD-Praktikums in OBS zum Ende der Vorlesungszeit.