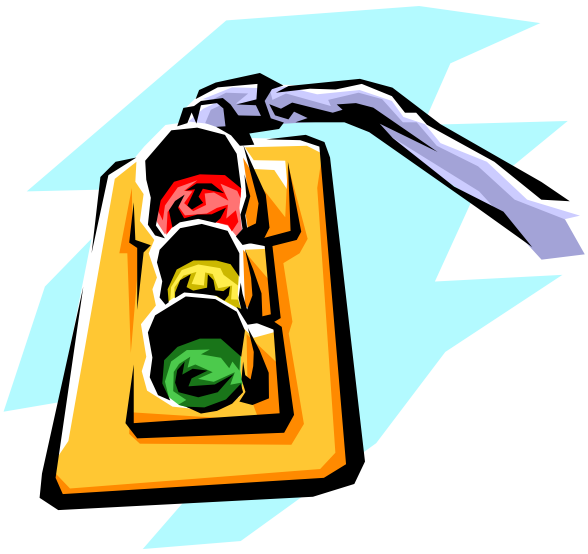
Softwarepraktikum

Teil: Eingebettete Systeme

Sommersemester 2003

Analyse I:

Struktur und Verhalten (Szenarien)



Aufgabe 1

Analyse I: Struktur und Szenarien

Umfang: 2 Wochen
Punkte: 100 P.

In diesem ersten Teil der Analyse sollen auf der Basis der gegebenen Problembeschreibung eine initiale Struktur für das zu entwicklende System erstellt und typische Szenarien erarbeitet werden.

Die **statische Struktur** des Systems wird durch zwei Modelle repräsentiert. Das erste Modell ist das **Klassenmodell**, welches die im System zu realisierenden Klassen und deren Relationen zueinander beschreibt. Dieses Modell dient der eindeutigen Zuordnung von Entwicklerverantwortlichkeiten zu Klassen und bildet im zweiten Teil der Analyse den strukturellen Rahmen für die Verhaltensbeschreibung mittels Zustandsdiagrammen.

Das zweite Modell ist das **Instanzenmodell**, welches die Instanziierung der Klassen und die Beziehung zwischen den einzelnen Instanzen beschreibt.

Um **Testfälle** für den späteren Test des Systems zu erstellen, werden schon während der Analyse **Szenarien** aufgestellt, die ausgewählte Aspekte der gewünschten Systemfunktionalität beschreiben. Diese Szenarien dienen zudem als Ausgangspunkt für die anschließende Verhaltensmodellierung. Daher kann das Aufstellen solcher Szena-

rien auch zu einem weitergehenden Verständnis des Problems und damit eventuell zu einer Änderung der Systemstruktur führen.

1 Struktur: Klassendiagramme

Bei der Erstellung eines Klassendiagramms zur Modellierung der Systemstruktur müssen **Klassen** (oder **Objekttypen**) identifiziert und die Beziehungen zwischen diesen mittels **Relationen** beschrieben werden.

Die Klassen können sich zum Einen aus einer existierenden physikalischen Struktur ergeben – bei der Ampelsteuerung spiegeln sich die Sensoren und Aktuatoren im Klassendiagramm wider. Zum Anderen kommen zusätzliche Klassen hinzu, die für die Realisierung der Funktionalität des Gesamtsystems benötigt werden. Im Beispiel der Ampelsteuerung wären das z. B. die Steuerungen der einzelnen Kreuzungen.

Jede Klasse kann **Attribute** unterschiedlicher Datentypen besitzen, wobei jede Instanz einer Klasse konkrete Attributbelegungen besitzt. Zum Beispiel könnte die Klasse *Zähler* ein Attribut *Zählerstand* vom Typ *Integer* besitzen und eine konkrete Instanz den *Zählerstand* 42 haben.

Typische Relationen sind die **Aggregation**, die **Generalisierung** und die **Assoziation**.

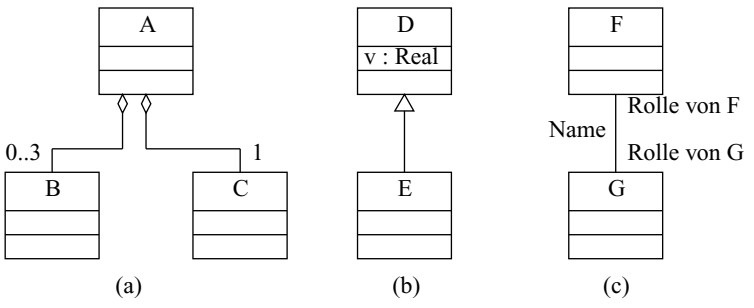


Abbildung 3 Beispiel für Relationen

Die Aggregation beschreibt wieviele Instanzen welcher Klasse von jeder Instanz der anderen Klasse aggregiert werden. In Abbildung

3(a) aggregiert z.B. jede Instanz der Klasse *A* bis zu 3 Instanzen der Klasse *B* und genau eine Instanz der Klasse *C*.

Die Generalisierung ermöglicht die Beschreibung der **Vererbungsbeziehung** zwischen Klassen. In Abbildung 3 (b) erbt Klasse *E* alle Eigenschaften von Klasse *D* (**Oberklasse**), d.h. dass Klasse *D* eine Generalisierung von Klasse *E* ist (daher auch der Pfeil in Richtung von *D*). In dem konrekten Beispiel bedeutet dies, dass Klasse *E* auch das Attribut *v* kennt.

Die Assoziation erlaubt schließlich die Beschreibung der statischen Verbindung zwischen Klassen, wobei zur Verdeutlichung zusätzlich zum **Namen der Assoziation** auch **Rollennamen** angegeben werden können (siehe Abbildung 3 (c)). Auch ohne Rollennamen kann die Assoziation durch Angabe eines Pfeils beim Namen der Assoziation genauer spezifiziert werden. In Abbildung 4 ist ein Beispiel gezeigt.

Relationen sind i.d.R. bidirektional. Es können aber auch gerichtete Relationen modelliert werden. Diese besitzen einen zusätzlichen Pfeil auf der *Linie*, welche die Relation darstellt (dies ist in der Abbildung nicht dargestellt!).

Abbildung 4 Gleichwertige Beschreibung von Assoziationen

Eine mögliche Vorgehensweise zur Aufstellung eines Klassendiagramms wird am folgendem Beispiel veranschaulicht. „Es soll eine Straße *bestehend* aus einer Fahrbahn mit zwei Fahrspuren und zwei Häuserzeilen mit jeweils 3 Häusern modelliert werden. Eine der Häuserzeilen *besitzt* zusätzlich einen Fußweg. Auf den Fahrspuren können Autos *fahren* und die Häuser werden von Menschen *besessen*.“

Die Klassen können aus den Substantiven hergeleitet werden. Die Relationen ergeben sich aus den *Verben*, z.B. beschreibt das Verb „besteht“ die Aggregation während die anderen Verben, z.B. „besit-

zen“, Assoziationen beschreiben. Die zwei Klassen der Häuserzeilen stehen zusätzlich in einer Vererbungsrelation zu der Oberklasse *Häuserzeile*. Damit ergibt sich ein mögliches Klassendiagramm wie in Abbildung 5.

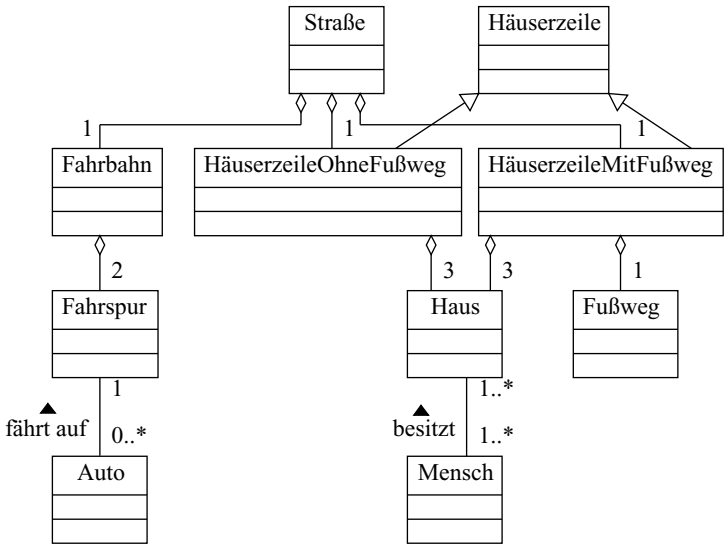


Abbildung 5 Klassendiagramm einer Straße

1.1 Aufgabe (25 Punkte)

Erstellen Sie ein UML-Klassendiagramm, das Ihr zu entwickelndes System beschreibt. Bei der Modellierung sollen Sie annehmen, dass Klassen eigenständige **Prozesse** sind, die ohne Aufforderung aktiv werden, Berechnungen parallel zu den anderen Klassen durchführen und daher **asynchron** Signale versenden können. Diese Annahme ist allgemeiner als die, dass die Kommunikation zwischen Klassen durch synchrone Methodenaufrufe stattfindet. Die Modellierung mit dem allgemeineren Ansatz macht Sinn, da die Abbildung der Prozesse und Signale auf Code in der Analyse noch nicht interessiert.

In UML kennzeichnet man solche sogenannten **aktiven Klassen**[1] durch einen dicken Rahmen (siehe Abbildung 6). Mit Assoziationen

1. Das im Praktikum eingesetzte Werkzeug Telelogic Tau (siehe Abschnitt 4) erlaubt es nicht aktive Klassen mit einem dicken Rahmen zu kennzeichnen. Da für die gesamte Analysephase allerdings von aktiven Klassen ausgegangen wird, kann man die normalen (dünnen) Klassensymbole gleichbedeutend verwenden.

können Interaktionen (also mögliche Signalwege) zwischen solchen aktiven Klassen ausgedrückt werden. Auch hierfür findet sich ein Beispiel in Abbildung 6.

Zu dem zu entwickelnden System gehört auch die Anbindung an die Umgebung, welche im Modell durch die Klassen *Signalgeber* und *Detektor* ausgedrückt wird. Abbildung 6 zeigt in welcher Relation diese Klassen zu einer Klasse *X* stehen. Unter *X* kann man sich z. B. eine einfache Steuerung vorstellen.

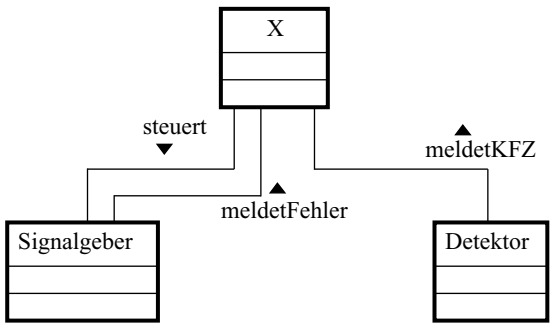


Abbildung 6 Modellierung der Anbindung an die Umgebung

Desweiteren soll auch die Benutzerkonsole als *eine* Klasse in diesem Modell repräsentiert werden, damit die benötigte Kommunikation zum dem restlichen System modelliert werden kann. Denken Sie bei dem Entwurf der Struktur auch daran, dass Konzepte wie z. B. „Grüne Welle“ realisiert werden müssen.

Nach der gemeinsamen Erstellung des Klassendiagramms in der Gruppe soll die Verteilung der Verantwortlichkeiten für die Entwicklung der einzelnen Klassen auf Teilgruppen von jeweils zwei Personen erfolgen. Es steht Ihnen frei wie Sie diese Zuordnung treffen. Allerdings ist diese für die Dauer des gesamten Praktikums verbindlich! Erstellen Sie ein Dokument, in der die Namen der Praktikumsteilnehmer zusammen mit den von ihnen bearbeiteten Klassen zu finden sind.

1.2 Abgabe

Abzugeben sind:

- das aussagekräftig dokumentierte Klassendiagramm
- die Zuteilung der Entwickler zu den einzelnen Klassen

2 Struktur: Instanzendiagramme

Das Klassendiagramm beschreibt das System noch sehr abstrakt. Es ist z. B. noch nicht klar wie (wann) und von wem die Klassen **instanziiert** werden und welche **Instanzen** nun wirklich über die auf Klassen beschriebenen Assoziationen verbunden sind.

Zu dieser eindeutigen Beschreibung dient ein Instanzendiagramm. In diesem werden für eine Instanziierung des Systems dessen Instanzen und alle Relationen zwischen diesen beschrieben.

Zur Veranschaulichung wollen wir hier wieder das Beispiel der Straße aufgreifen. Nun werden wir um Einiges konkreter: „Die Straße Forellenweg besteht aus einer Fahrbahn, welche eine linke und rechte Fahrspur besitzt. Die linke Häuserzeile besteht aus den Häusern mit den Hausnummern 8, 10 und 12, die rechte Häuserzeile aus den Häusern mit den Nummern 9, 11 und 15 mit einem zusätzlichen Fußweg. Haus 8 besitzt Herr Müller, Haus 10 Frau Maier. Häuser 9 und 11 stehen leer und die Häuser Nummer 12 und 15 gehören Herrn und Frau Martin. Es fährt ein einziges Auto auf der rechten Fahrspur.“

Abbildung 7 zeigt das zugehörige Instanzendiagramm.

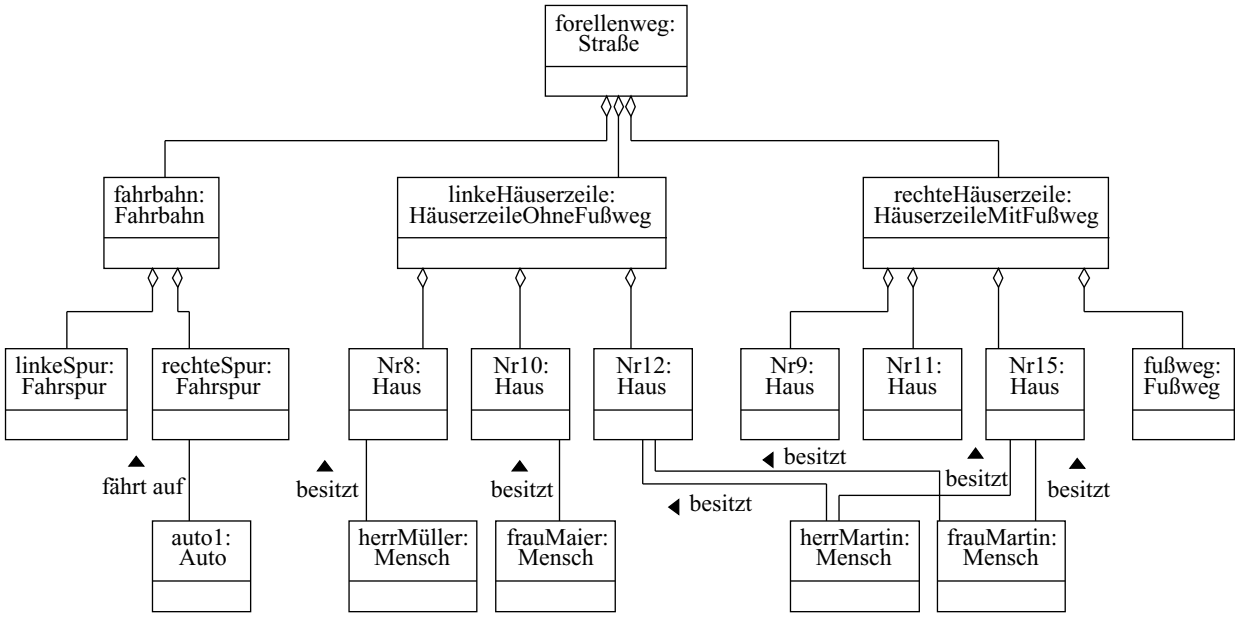


Abbildung 7 Instanzendiagramm der Straße „Forellenweg“

2.1 Aufgabe (25 Punkte)

Erstellen Sie das Instanzendiagramm (**UML-Objektdiagramm**) für Ihr System.

Damit die spätere Anbindung an die Umgebung vereinfacht wird, benutzen Sie für die Namen der Instanzen der Signalgeber und Detektoren das folgende Namensschema:

- **Signalgeber:** lsa<*m*>_sg<*n*>, wobei <*m*> die Nummer der Lichtsignalanlage und <*n*> die Nummer des Signalgebers ist. Beide Angaben entnehmen Sie bitte den Signallageplänen. Als Beispiel soll Signalgeber 1 der Lichtsignalanlage 7 den Instanzennamen *lsa7_sg1* bekommen.
- **Detektoren:** lsa<*m*>_d<*n*>, wobei <*m*> wieder die Nummer der Lichtsignalanlage und <*n*> die Nummer des Detektors ist.

2.2 Abgabe

Abzugeben ist:

- das aussagekräftig dokumentierte Instanzendiagramm

3 Szenarien: Sequenzdiagramme

Die in den beiden vorangehenden Abschnitten erarbeitete statische Struktur des Systems bildet nun die Grundlage für eine erste Beschreibung der **dynamischen** Aspekte des Systems. Dies erfolgt durch Modellierung typischer **Szenarien**, welche mit Hilfe von UML-Sequenzdiagrammen dargestellt werden. **UML-Sequenzdiagramme** beschreiben *mögliche* Interaktionen zwischen Instanzen.

Ein Sequenzdiagramm besitzt immer zwei Dimensionen. Die vertikale **Dimension** repräsentiert die **Zeit** und die horizontale Dimension repräsentiert unterschiedliche Instanzen. Normalerweise sind nur zeitliche *Reihenfolgen* wichtig. In Realzeitsystemen kann man die Zeitachse allerdings auch mit absoluten Zeitwerten versehen. Letzteres wollen wir hier für die Modellierung aber nicht einsetzen, lediglich zur Verdeutlichung einiger Konzepte werden wir davon Gebrauch machen (siehe unten).

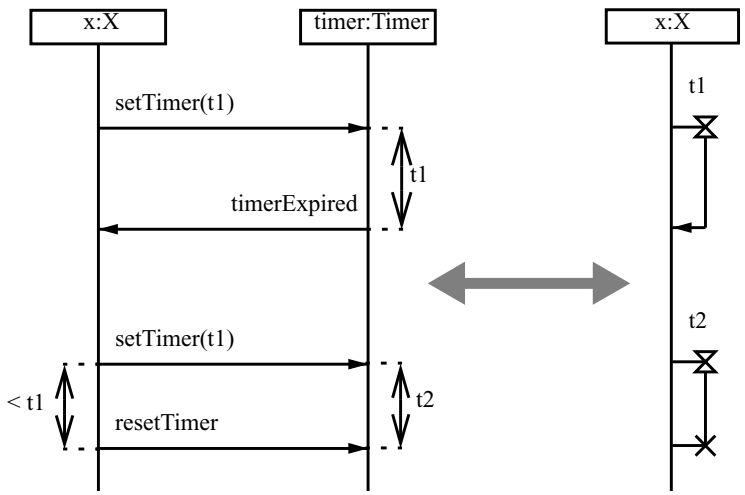


Abbildung 8 Explizite und abkürzende Modellierung eines Timers in einem Sequenzdiagramm

Die **Interaktion** zwischen den Instanzen auf der horizontalen Achse wird durch **Nachrichten** modelliert. Es werden prinzipiell **synchrone** und **asynchrone** Nachrichten unterschieden. Zu den synchronen Nachrichten gehören **Methodenaufrufe** und **-rückgaben**. Asynchrone Nachrichten sind **Ereignisse** oder **Signale**. Wie in Abschnitt 1 schon begründet, sollen allerdings während der Analyse nur asynchrone Nachrichten verwendet werden.

Zeitintervalle lassen sich durch Pfeile (mit Pfeilspitzen an jedem Ende) parallel zur Zeitachse ausdrücken. In Abbildung 8 ist ein Beispiel für die Realisierung eines Timers gezeigt. In der linken Bildhälfte mit Hilfe einer Instanz der Klasse *Timer* ausgedrückt und in der rechten Bildhälfte abkürzend als spezielles Signal beschrieben. Diese abkürzende Schreibweise für Timer soll auch für die Modellierung eingesetzt werden.

Beim Einsatz eines Timers sind zwei Szenarien denkbar. Zum einen kann ein Timer „aufgezogen“ werden und dann nach Ablauf der eingestellten Zeit ein Signal erzeugen (das entspricht einem **Timeout** des Timers). Es kann aber auch während der Laufzeit des Timers ein explizites Rücksetzen (**Reset Timer**) passieren (also quasi ein Stop des Timers)[1].

Um Bedingungen in Sequenzdiagrammen zu beschreiben werden sechseckige Bedingungssymbole verwendet, in die man eine Bedingung in beliebiger Form eintragen kann.

In Abbildung 9 wird zur Veranschaulichung ein Szenario für eine Flurbeleuchtung gezeigt. Abbildung 10 zeigt das zugehörige Klassendiagramm (hier wurde die Klasse *Timer* zur Verdeutlichung nochmals explizit aufgenommen, kann in den zu erstellenden Klassen- und Instanzendiagrammen aber weggelassen werden).

1. Die verwendete Notation für die Beschreibung des Timer-Verhaltens (für die Instanz x auf der rechten Seite von Abbildung 8) entspricht der Notation des im Praktikum eingesetzten Modellierungswerkzeugs (siehe Abschnitt 4). Die UML-Notation anderer Werkzeuge kann davon leicht abweichen.

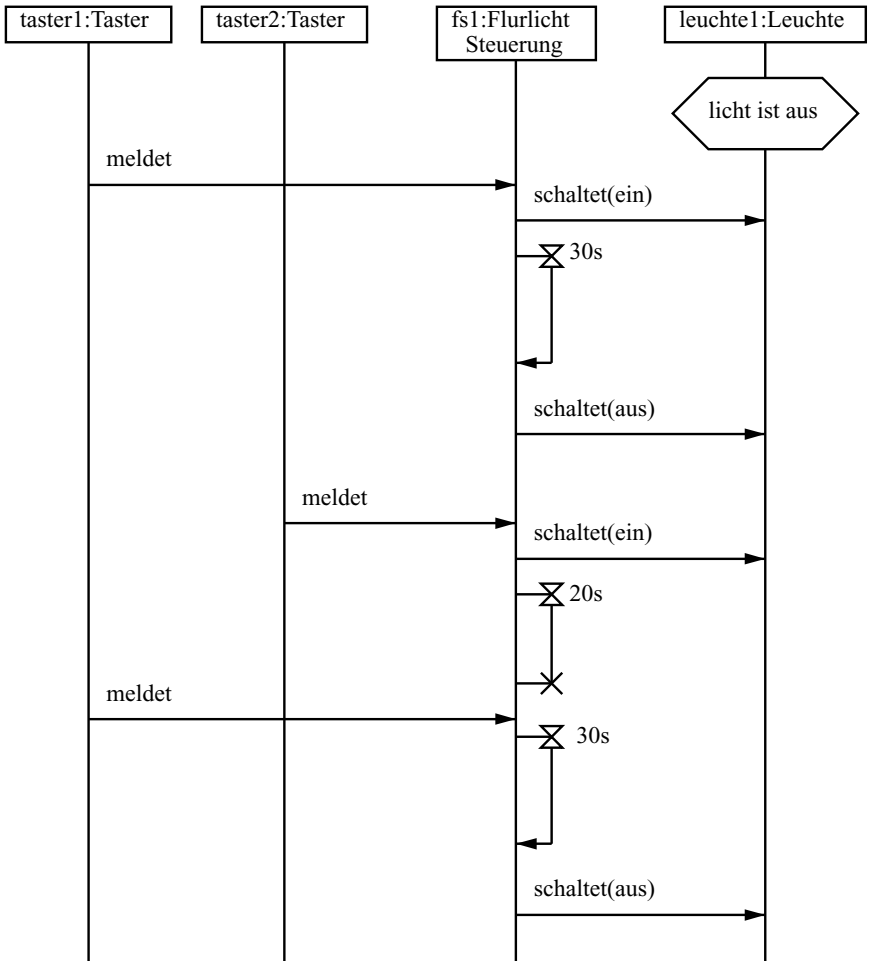


Abbildung 9 Flurbeleuchtung, Sequenzdiagramm

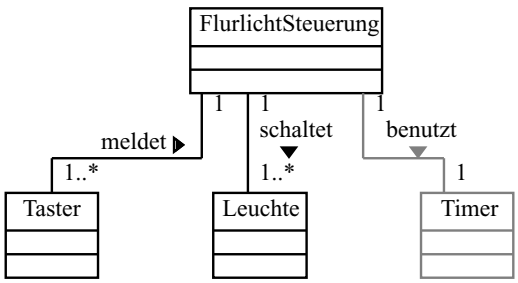


Abbildung 10 Flurbeleuchtung, Klassendiagramm

3.1 Aufgabe (50 Punkte)

Nutzen Sie Ihre stukturellen Modelle aus den Abschnitten 1 und 2 und erstellen Sie mindestens 20 typische Szenarien. Diese Szenarien sollen vornehmlich für spätere Tests geeignet sein. Berücksichtigen Sie dazu, z.B. was bei einem Detektor-Ereignis passiert oder wie die Kommunikation zwischen den Signalanlagen bei der „Grünen Welle“ abläuft.

Beschreiben Sie zu jedem Szenario informell was passiert und stellen Sie diese Szenarien mit Hilfe von UML-Sequenzdiagrammen dar. Dabei sollen sowohl Abläufe *innerhalb* einer Signalanlage (für eine Kreuzung) als auch Abläufe *zwischen* den einzelnen Knotenpunkten (für „Grüne Welle“) dargestellt werden.

Hinweis: Wenn Sie feststellen, dass Sie mit der initialen Objektstruktur nicht weiterkommen, gehen Sie zu den Abschnitt 1 und 2 zurück und modifizieren Sie diese Diagramme!

3.2 Abgabe

Abzugeben sind:

- mind. 20 Sequenzdiagramme mit einer aussagekräftigen textuellen Beschreibung des jeweiligen Szenarios

4 Kurzeinführung in das Modellierungswerkzeug Telelogic Tau

Im Praktikum wird ein kommerzielles Modellierungswerkzeug ‘**Telelogic Tau**’ für die Erstellung der Modelle zum Einsatz kommen. Ursprünglich war das Werkzeug für die Erstellung von **SDL**-Modellen (**Specification and Description Language**) konzipiert, was nach wie vor den größten Teil des Tools ausmacht. Allerdings besitzt Telelogic Tau mittlerweile auch einen Satz sehr guter UML-Editoren.

Dieses Werkzeug ist auf den Ausbildungsrechnern installiert und kann mitdem Befehl **start-sdt** gestartet werden.

Nach dem Start erscheint (nach einem eventuellen Einstiegsfenster) der sogenannte **Organizer**, welcher der Verwaltung der Modell-

Dokumente dient. In Abbildung 11 ist das Fenster des Organizers gezeigt.

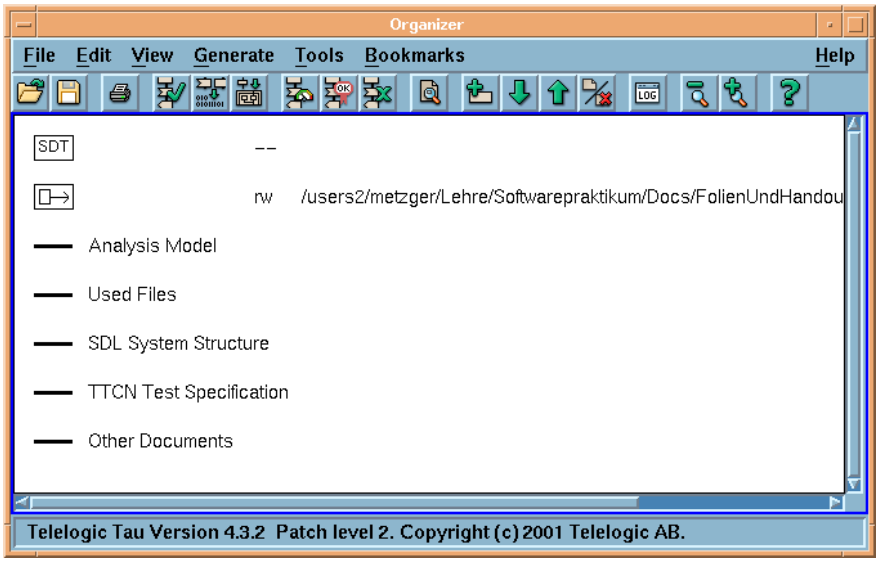


Abbildung 11 Organizer

Um ein neues Modell-Dokument zu erstellen, ruft man im **Edit**-Menü den Punkt **Add New...** auf. Es erscheint dann ein Auswahlfenster, in welchem man den gewünschten Diagrammtyp auswählen kann. Für das Praktikum sind nur die Diagrammtypen **UML/Object Model** für Klassen- und Instanzendiagramme, **UML/State Chart** für die Zustandsdiagramme und **MSC/MSC**[1] für die Sequenzdiagramme wichtig. Im Feld **New Document Name** gibt man den Namen des Dokuments ein. Danach erscheint ein Icon im Organizer und der entsprechende Editor wird geöffnet.

Allen Editoren ist gemeinsam, dass auf der rechten Seite eine Palette von Symbolen erscheint, die man mit einem einfachen Mausklick auswählt und dann auf der Zeichenfläche platzieren kann. Klickt man ein platziertes Symbol an, so erscheinen Punkte, mit denen man Relationen u.ä. per Drag-and-Drop erstellen kann. Mit einem Klick mit

1. Bei den MSCs handelt es sich um sogenannte *Message Sequence Charts*, die prinzipiell den Sequenzdiagramme entsprechen.

der rechten Maustaste auf ein platziertes Symbol erscheinen zusätzliche Bearbeitungsmöglichkeiten.

Nach einer gewissen Einarbeitungszeit lassen sich sehr schnell auch komplexe Diagramme erstellen. Besonders der Export der Dokumente in sehr vielen Formaten (auch HTML) sollte die geforderte Dokumentation der Modelle erleichtern.

Eine sehr ausführliche Hilfe ist unter dem Menü **Help** verfügbar. Sehr sinnvoll ist hierbei das gezielte Suchen nach einem Thema unter dem Punkt **Search...**, da die gesamte Online-Hilfe sehr viele Informationen beinhaltet.

5 Literatur

[BRJ99] G. Booch, J. Rumbaugh, I. Jacobson. *The Unified Modeling Language User Guide*. Reading, Mass.; Harlow, England; Menlo Park, Calif.: Addison-Wesley Longman, Inc. 1999