Skript für STYLE-Langdemo

Ingrid Wetzel, Petra Münnix, Ralf Löst

24. Mai 1995

1 Vorbereitung der Umgebung

1.1 Rechnerwahl

Es ist vorgesehen, die Demo auf dem Rechner `dbis12` vorzuführen. Sie kann zur Zeit aber auch auf den Rechnern `dbis1` und `dbis2` laufen. Vorteilhaft, aber nicht zwingend notwendig, ist die Benutzung eines Farbbildschirms.

1.2 Einloggen in die Testumgebung

Für die Demo sind drei Kommandofenster nötig: zwei für zwei Tycoon Maschinen und das dritte zum Öffnen von Textdateien. Die folgenden Schritte gelten für jedes der Kommandofenster.
Nach dem Einloggen unter dem Benutzernamen `om1`[1] sind folgende Befehle auszuführen, um in die Umgebung `/local/tw1/om1/test/tycoon/bootlib/` zu gelangen:

```
tw test
cd tycoon/bootlib
```

Diese Langdemobeschreibung und die Kurzdemobeschreibung stehen unter `bootlib` in dem Verzeichnis `scripts` als LateX Dateien `longDemoScript.tex` und `shortDemoScript.tex`.

1.3 Rücksetzen der Stores

Da innerhalb der Demo der Store `om1Tracy` mit den Datenbankdaten verändert wird, müssen bei einer Wiederholung die Ausgangssituationen wiederhergestellt werden. Dies geschieht durch Kopieren der Stores (auf tycoon/bootlib):

```
cp saveom1Tracy.ts om1Tracy.ts
cp saveom1Tracy.tsi om1Tracy.tsi
cp saveom1TracyTransIc.ts om1TracyTransIc.ts
cp saveom1TracyTransIc.tsi om1TracyTransIc.tsi
```

(dauert eine Weile)

2 Vorbereitung der Langdemo

Diese Vorbereitung ist in den Abschnitten 2.2 bis 2.3 nahezu identisch zur Vorbereitung der Kurzdemo. (Lediglich die Datei `longDemoInteractive.tm` wird zusätzlich geöffnet).

[1] Das `password` kann bei Andreas Rudloff, Petra Münnix oder Ralf Löst erfragt werden.

2.1 Starten von Tycoon und von STYLE

Im ersten Kommandofenster ist unter dem Verzeichnis `/local/tw1/om1/test/tycoon/bootlib/` einzugeben:

```
../bin/sunos4/tycoon -store om1edit
session.init();
initGRD.loadPostScript();
styleTop.create("");
```

(Dauer ca. fünf bis sieben Minuten)
Es wird eine Tycoon-Maschine gestartet und (nach einigen Initialisierungen) der STYLETop Editor aufgerufen. Er erscheint als Fenster auf dem Bildschirm (mit blauer Kopfleiste). Das Fenster des STYLETop Editors sowie das Kommandofenster mit der gestarteten Tycoon Maschine werden wie in Abb. 1 dargestellt positioniert.

Im zweiten Kommandofenster ist unter demselben Verzeichnis einzugeben:

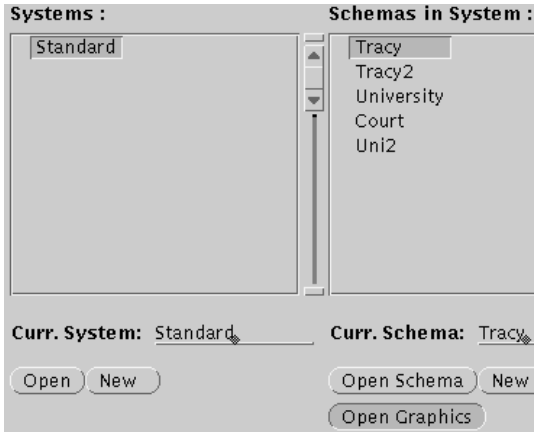
```
../bin/sunos4/tycoon -store om1TracyTransIc
```

(Dauer ca. fünf bis sieben Minuten)
Es wird eine zweite Tycoon Maschine gestartet, die für interaktiv einzugebenden TL Code (unter geänderten ICs – vgl. Abschnitt 4) dient.

2.2 Auswahl der TRACY-Anwendung

Im STYLETop Editor sind folgende Eingaben zu tätigen:

- Im linken Dateiauswahlfenster: *Standard* auswählen;
- links unten im Editor: den Knopf *Open* drücken und warten; nach einigen Sekunden erscheinen im rechten Dateiauswahlfenster Schemanamen;
- im rechten Dateiauswahlfenster: *Tracy* auswählen.



2.3 Starten der Editoren und Browser der TRACY Anwendung

Im folgenden werden die Fenster der Editoren/Browser der TRACY-Anwendung gestartet, als geöffnete Fenster auf dem Bildschirm positioniert und danach jeweils geschlossen und als Icons wiederum auf dem Bildschirm positioniert. Die Reihenfolge ist wie folgt.

2.3.1 Graphikeditor

1. **Öffnen:** vom Fenster des STYLETop Editors aus:
   Aktion: den Knopf *OPEN Graphics* drücken und warten, bis das Diagramm vollständig angezeigt wird (Dauer ca. 15 Sekunden);

Anzeige: Graphikeditorfenster mit Referenzdiagramm (Klassenknoten, Referenzkanten, Rautensymbole für versteckte Werteknoten) und grüner Kopfleiste;

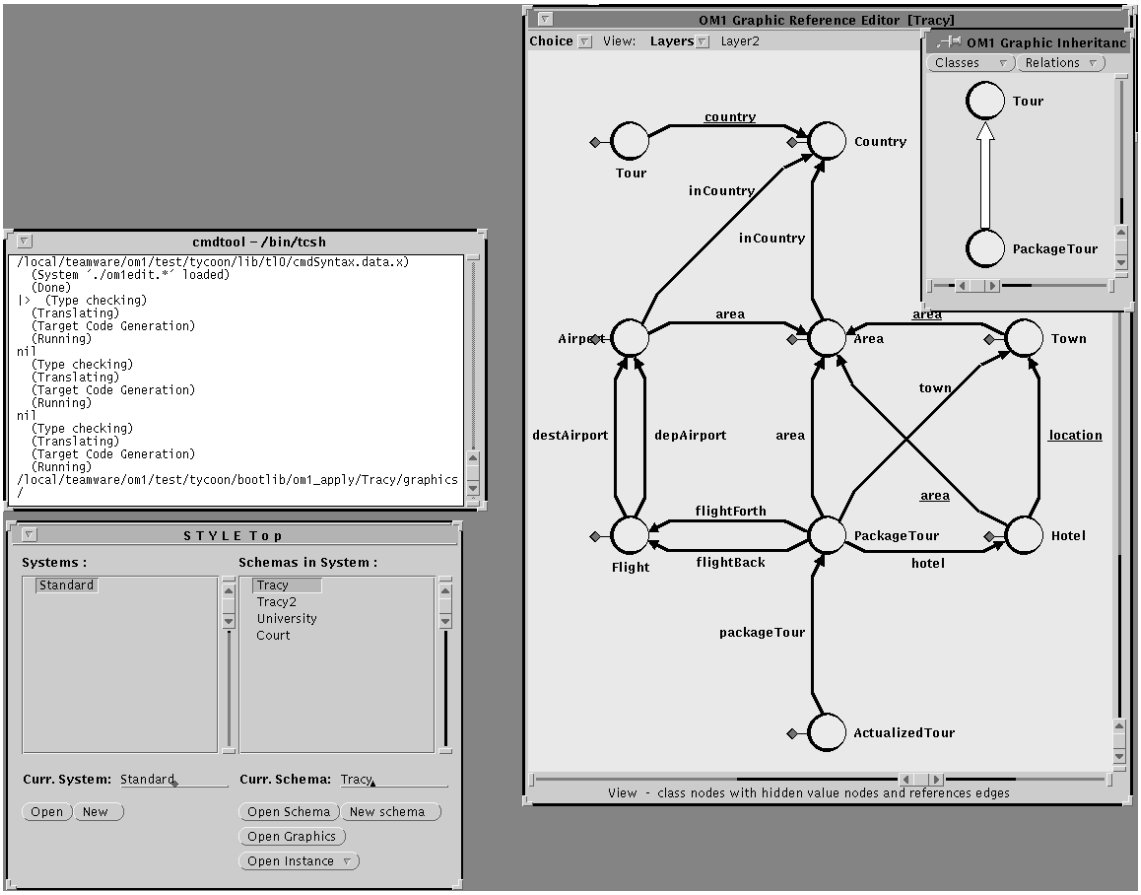
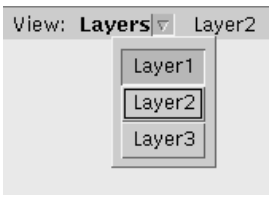


Abbildung 1: Fensteranordnung: links unten: *StyleTopEditor*; rechts oben: *Graphikeditor*; links mitte: *Kommandofenster*.

2. im Fenster des Graphikeditors:

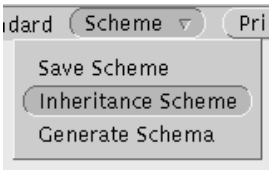
   - **Ebene 1 auswählen:**

     Aktion: im Menü *Layers* den Menüpunkt *Layer1* auswählen;



   - **Hierarchiefenster öffnen:**

     Aktion: im Menü *Scheme* den Menüpunkt *Inheritance Scheme* auswählen;



3. **Positionieren** des Editorfensters und des Hierarchiefensters:
   Aktion: Verkleinern der beiden Editorfenster, beharrliches (!) Betätigen der Scrollbars zum mittigen Positionieren der Diagramme in den Fenstern und Verschieben der Fenster wie in Abb. 1 dargestellt.

2.3.2 Instanzbrowser

Achtung! Unbedingt vor dem Klasseneditor öffnen! Sonst stürzt der Klasseneditor ab!

1. **Öffnen:** vom Fenster des STYLETop Editors aus:

   Aktion: im Menü *Open Instance* den Menüpunkt *Start* auswählen;
   Anzeige: Es wird in einer zweiten Tycoon-Maschine der Store `om1Tracy` geladen. Dieser Vorgang wird im Kommandofenster kommentiert. Dauer ca. fünf (!!) Minuten. Danach erscheint der Instanzbrowser, der die Instanzeditoren und Operationen anzeigt (mit roter Kopfleiste);

2. **Positionieren**:
   Aktion: Verschieben wie in Abb. 2 angezeigt.

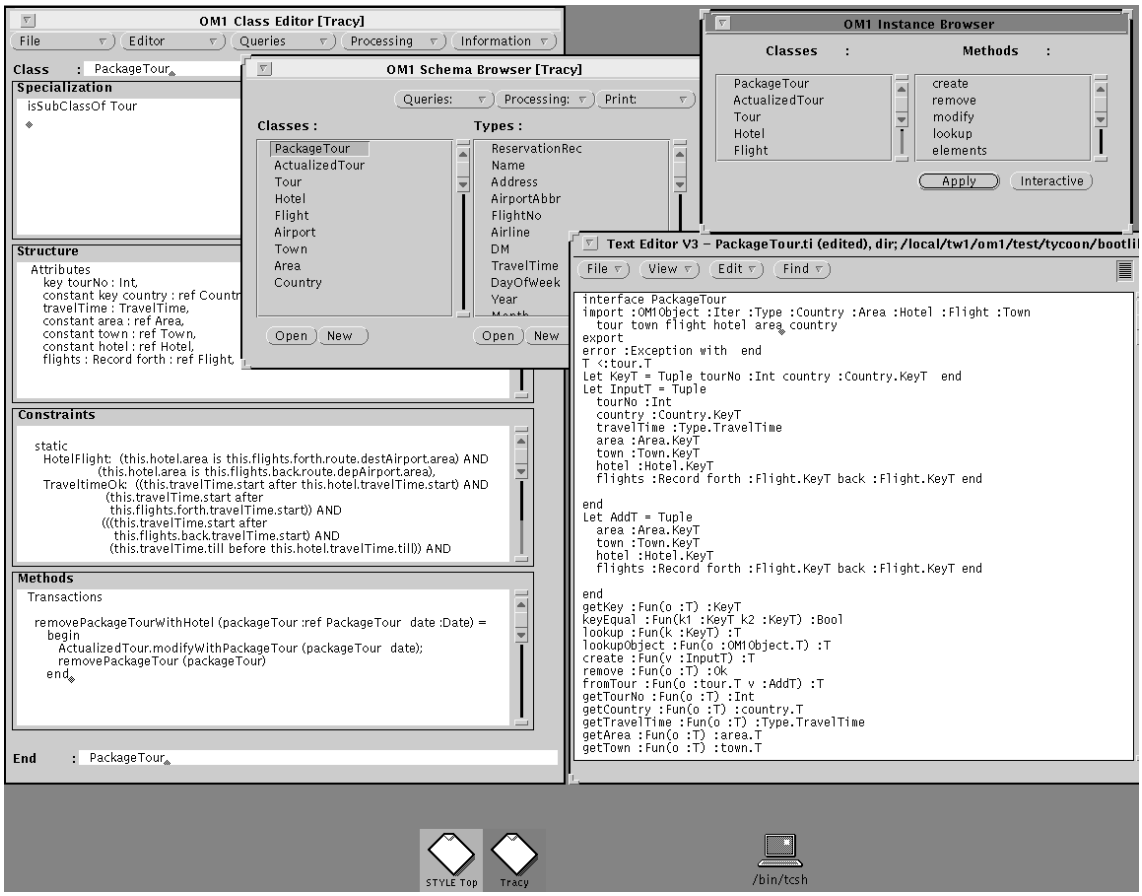


Abbildung 2: Fensteranordnung: oben links: *Klasseneditor*; oben mitte: *Schemabrowser*; oben rechts: *Instanzbrowser*; rechts: *generierte Klassenschnittstelle*.

2.3.3 Schemabrowser

1. **Öffnen:** vom Fenster des STYLETop Editors aus:
   Aktion: den Knopf *OPEN Schema* drücken; Nach einigen Sekunden erscheint der Schemabrowser (mit gelber Kopfleiste), der die TRACY Klassen und Typen anzeigt;

2. **Positionieren**:
   Aktion: Verschieben des Schemabrowsers wie in Abb. 3 angezeigt.

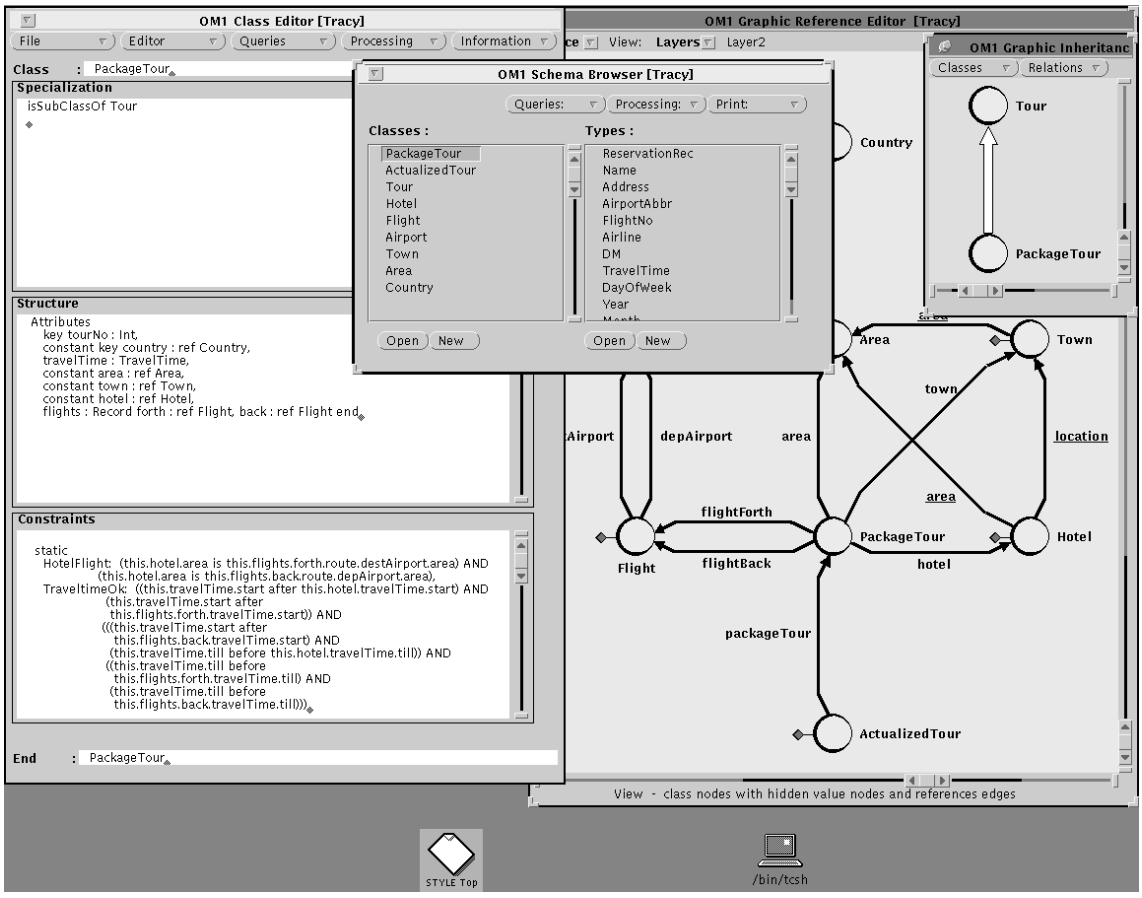
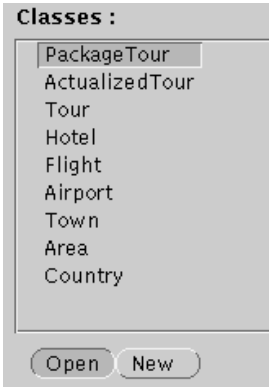


Abbildung 3: Fensteranordnung: links: *Klasseneditor*; rechts: *Graphikeditor*; oben mitte: *Schemabrowser*.

2.3.4 Klasseneditor für die Klasse *PackageTour*

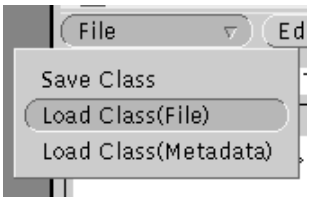
1. **Öffnen:** vom Fenster des Schemabrowsers aus:



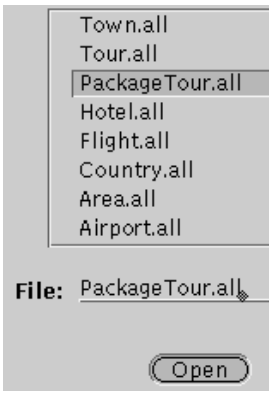
   Aktion: im linken Auswahlfenster: *PackageTour* auswählen, danach links unten den Knopf *Open* drücken;
   Anzeige: Nach ca. zehn Sekunden erscheint der Klasseneditor (Fenster mit dem Titel *PackageTour* und mit gelber Kopfleiste);

2. ***Methods*-Komponente aus Textdatei hinzuladen**:

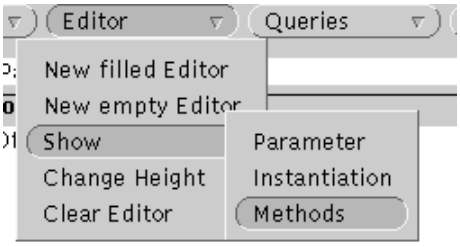
- **Aktion**: im Klasseneditor:
  im Menü *File* den Menüpunkt *Load Class (File)* auswählen; es erscheint ein Auswahlfenster;



- **Aktion**: im Auswahlfenster:
  PackageTour.all auswählen und den Knopf *Open* drücken;



- **Aktion**: im Klasseneditor:
  im Menü *Editor* den Menüpunkt *Show* und darin den Menüpunkt *Methods* wählen;
  Anzeige:
  Es wird die *Methods*-Komponente im Klasseneditor angezeigt.



3. **Positionieren**:
   Aktion: Verschieben wie in Abb. 3 angezeigt.

2.4 Öffnen von Textdateien

Für das Öffnen von Textdateien wird das dritte vorbereitete Kommandofenster (mit dem Verzeichnispfad `/local/tw1/om1/test/tycoon/bootlib/`) verwendet.

- Klasseninterface für *PackageTour*
  Öffnen von `PackageTour.ti` im Verzeichnis `tycoon/bootlib/om1Tracy`:
  ```
  te om1Tracy/PackageTour.ti
  ```

- Dateien mit Funktionen auf Schnittstellen für Interaktivmodus
  Öffnen von `shortDemoInteractive.tm` und `longDemoInteractive.tm` im Verzeichnis `tycoon/bootlib/scripts`:
  ```
  te scripts/shortDemoInteractive.tm
  te scripts/longDemoInteractive.tm
  ```

Das Fenster mit dem Tycoon Klasseninterface für *PackageTour* wird entsprechend Abb. 2 positioniert, die beiden anderen Fenster untereinander davor.
Danach werden alle Fenster geschlossen und ihre Icons entsprechend Abb. 4 auf dem Bildschirm positioniert. Zur Unterscheidung der Icons sind sie verschieden gefärbt und beschriftet:

- STYLETop Editor: blau, Name: *StyleTop*;
- Graphikeditor: grün, Name: *Tracy*;
- Schemabrowser: gelb, Name: *Tracy*;
- Instanzbrowser: rot, Name: *Tracy*;
- Klasseneditor: gelb, Name: *PackageTour*;
- weitere Icons für Kommando-/Textfenster.

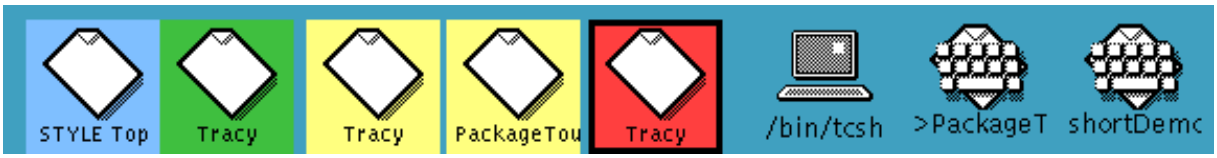
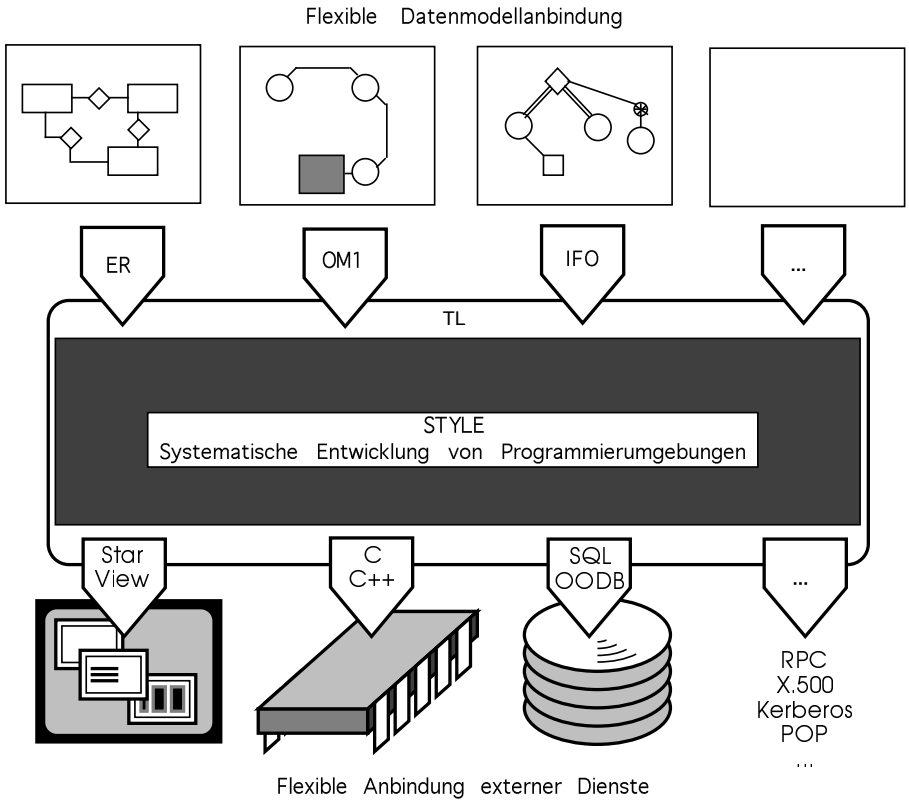


Abbildung 4: Anordnung von Icons der STYLE Werkzeuge und weiterer Kommando- und Editierfenster

3 Start der Demo

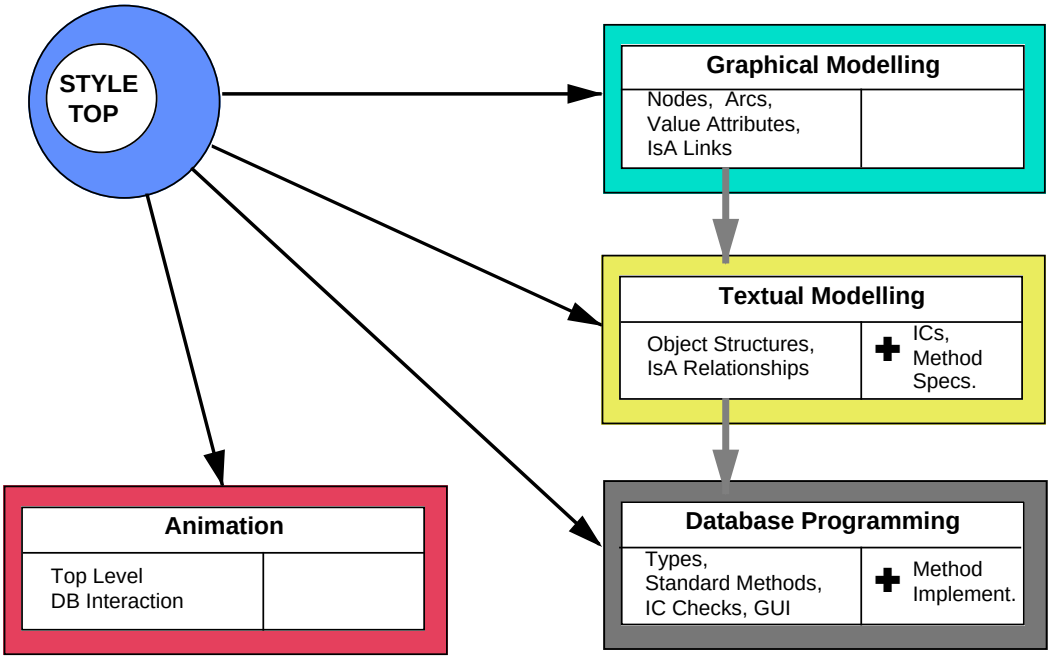
Der in diesem Abschnitt vorgestellte Ablauf entspricht genau dem der Kurzdemo. In Abschnitt 4 folgen die zusätzlichen Schritte der Langdemo.

3.1 Folie I



Willkommen zur STYLE Demo. Das STYLE Projekt untersucht die flexible Unterstützung unterschiedlicher Datenmodelle in einer persistenten, uniformen Sprachumgebung mit flexibler Anbindung externer Dienste durch systematische Konstruktion von Enwicklungsumgebungen. Die STYLE Demo präsentiert eine datenmodellspezifische Entwickungsumgebung für das objektorientiere Datenmodell OM1, deren Realisierung der STYLE Systematik folgt und damit auf der systematischen Kombination generischer Dienste und Generatoren basiert.

3.2 Folie II

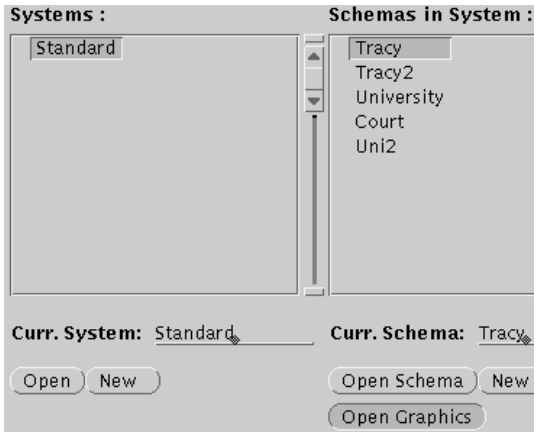


Die Demo beginnt mit dem STYLE-Top Editor, von dem aus Werkzeuge auf der Modellierungs-, der Programmier- und der Animationsebene wählbar sind. Eine typische Entwurfsunterstützung durch die STYLE Umgebung sieht folgende Aufgaben vor.

Im Graphikeditor modelliert der Benutzer Anwendungsschemata durch Klassenknoten, Werteknoten, Referenz- und Vererbungskanten. Aus diesen strukturellen Informationen werden durch Generatorunterstützung textuelle Schemarepräsentationen erzeugt, die im Texteditor um benutzerdefinierte Integritätsbedingungen und Methoden ergänzt werden. Aus dieser vervollständigten Modellierung werden TL Schnittstellen und Module generiert, die Objekt- und Klassenstrukturen sowie Standardmethoden bereitstellen, die die modellinhärenten und benutzerdefinierten Integritätsbedingungen erfüllen. Auf diesen Schnittstellen erfolgt die Programmierung der benutzerdefinierten Methoden in der TL-Sprachumgebung. Die Animationskomponente stellt einen Datenbankprototyp mit Benutzerschnittstelle zum Erzeugen, Löschen und Verändern von Anwendungsobjekten bereit. Dieser ruft die generierten Standardmethoden auf.

3.3 STYLETop Editor

Aktion: vorbereiteten STYLETop Editor aufklappen Wir beginnen mit dem STYLETop Editor, in dem vorhandene Systeme mit den in ihn enthaltenen Schemata angezeigt werden. Wir wählen in dem Standardsystem das Schema *TRACY* aus.
Zu diesem Schema können wir über den Knopf *Open Graphics* den Graphikeditor öffnen, über den Knopf *Open Schema* den Schemabrowser, der eine Liste von Klassen- und Typnamen des Schema anzeigt, aus der heraus die jeweiligen Klasseneditoren geöffnet werden können. *Open Instance* ruft die Animationskomponente auf.
Wir rufen zunächst den Graphikeditor auf (*Open Graphics*).

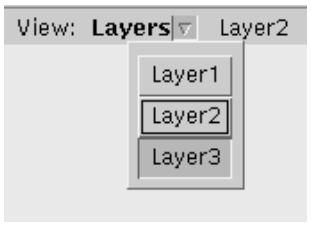


3.4 Graphikeditor

(allgemein bei Bedienung zu beachten: Maushandling, Zusammenspiel von linker und rechter Maustaste)

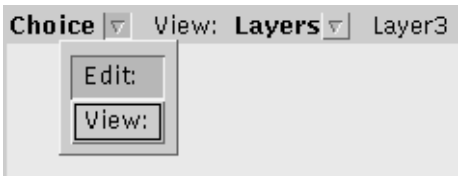
Aktion: vorbereiteten Graphikeditor aufklappen Der Graphikeditor zeigt unser Demobeispiel, die Modellierung einer Reiseagentur. Wir betrachten hauptsächlich die Klassen *PackageTour*, *Hotel* und *Flight*. Die graphische Anzeige erfolgt in unterschiedlichen Granularitätsstufen, die aktuelle Anzeige gibt einen Überblick über die definierten Klassen eines Schemas sowie deren Beziehungen. Vererbungsbeziehungen werden der Übersicht halber in einem eigenen Fenster angezeigt.

Aktion: in Menü *Layers* den Menüpunkt *Layer3* auswählen
Eine weitere Stufe ergänzt die Überblicksanzeige um die Werteattribute von Klassen.
Aktion: in Menü *Layers* den Menüpunkt *Layer1* auswählen



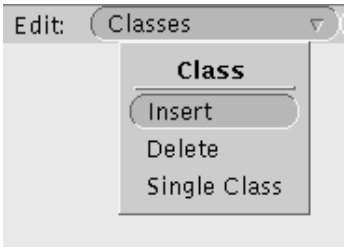
Aktion: im Menü *Scheme* den Menüpunkt *Generate Schema* auswählen und Scrollen über die generierte textuelle Ausgabe im Kommandofenster der Tycoon-Maschine.
Schemaweit lassen sich aus diesen strukturellen Informationen textuelle Repräsentationen generieren.

Neben Schemaanzeige lassen sich im Editor Schemata graphisch erstellen. Wir wechseln hierzu in den Editmodus.
Aktion: in Menü *Choice* den Menüpunkt *Edit* auswählen
Neben Schemaanzeige lassen sich im Editor Schemata graphisch erstellen. Wir wechseln hierzu in den Editmodus.



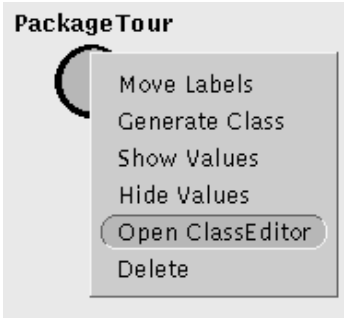
Falls das Einfügen von Klassenknoten (oder Kanten) gezeigt werden soll, dann

Aktion: in (im Edit-Modus erscheinendem) Menü *Classes* den Menüpunkt *Insert* auswählen



Gridraster erscheint. Gridpunkt mit linker Maustaske selektieren, Label eintippen und *Return*-Taste drücken. (Bei Löschen des Knotens, erst in Menü *Classes* den Menüpunkt *Delete* auswählen, dann den zu löschenden Klassenknoten selektieren).

Aktion: in Menü *Classes* den Menüpunkt *Single Class* auswählen, dann mit der linken Maustaste den Klassenknoten *PackageTour* selektieren und durch Drücken und Halten der rechten Maustaste auf dem Klassenknoten das klassenbezogene Menü öffnen



Über den Menüpunkt *Generate Class* kann die Textgenerierung klassenweise durchgeführt werden und über den Menüpunkt *Open ClassEditor* der zu einem Klassenknoten gehörige Klasseneditor geöffnet werden.

3.5 Schemabrowser

Aktion: vorbereiteten Schemabrowser aufklappen Alternativ läßt sich der Klasseneditor zu einer Klasse über den Schemabrowser öffnen. Diesen Weg gehen wir für die Klasse *PackageTour*.

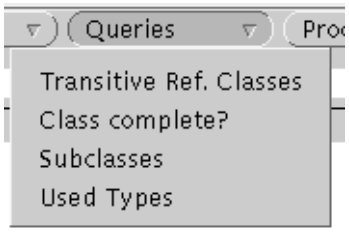
3.6 Klasseneditor

Aktion: vorbereiteten Klasseneditor aufklappen Der Klasseneditor zeigt die vier Komponenten einer OM1 Klassendefinition *Specialization*, *Structure*, *Constraints* und *Methods*. In der Spezialisierungskomponente ist die *isSubClassOf*-Beziehung zwischen *Tour* und *PackageTour* angezeigt. Die Strukturkomponente besteht (in der Vollanzeige) aus den geerbten (z.B. *tourNo*, *travelTime* und *country*) und den zusätzlich definierten (weiteren) Attributen der Klasse *PackageTour*, wobei Referenzbeziehungen durch das Schlüsselwort **ref** gekennzeichnet sind.

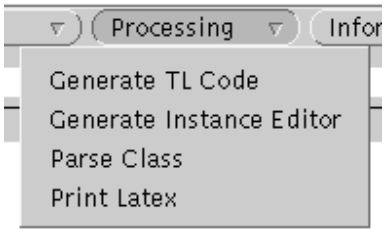
Die Komponente *Constraints* enthält benutzerdefinierte Integritätsbedingungen, in Prädikatenlogik erster Ordnung formuliert. Die erste Bedingung *HotelFlight* legt fest, daß die angebotenen Flüge einer Pauschalreise in die gleiche Region führen müssen, in der das Hotel liegt.

Die Komponente *Methods* enthält eine benutzerdefinierte Methode *removePackageTourWithHotel*, die im Falle eines Hotelausfalls alle gebuchten Reisen zu diesem Hotel auf entsprechende andere Hotels (und andere Pauschalreisen) umbucht und erst danach ein Löschen der Pauschalreise vornimmt.

Aktion: Menü *Queries* aufgeklappt halten
Im Klasseneditor werden Modellierungshilfen in Form von Anfragen an das gesamte Schema angeboten, z.B. nach den transitiv referenzierten Klassen (Menüpunkt: *Transitive Ref. Classes*), benutzten Typen (Menüpunkt: *Used Types*) oder Subklassen (Menüpunkt: *Subclasses*) einer Klasse. Über deren Auflistung kann man erneut Klassen- oder Typeditoren aufrufen. Hierdurch wird z.B. die Formulierung von klassenübergreifenden Integritätsbedingungen unterstützt.



Aktion: Menü *Processing* aufgeklappt halten
Aus dem Klasseneditor heraus kann die Codegenerierung sowie die Erzeugung von Instanzeditoren klassenweise aufgerufen werden. Darüberhinaus werden Funktionen zur Syntaxüberprüfung und Erzeugung eines LateX-Dokumentes angeboten.



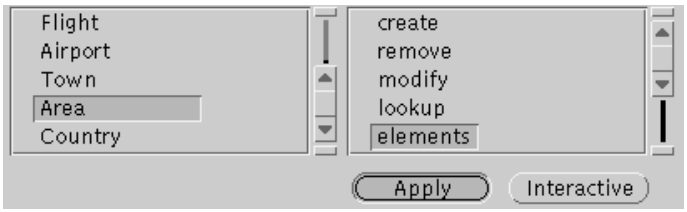
3.7 Programmierebene

Aktion: Aufklappen der generierten Schnittstelle PackageTour.ti Das Fenster zeigt die aus der OM1 Klassendefinition generierte Schnittstelle `PackageTour.ti`. Sie exportiert einen abstrakten Objekttyp T zusammen mit Standardmethoden zum Einfügen, Löschen, Lesen und Ändern von Objekten. Die Änderungsmethoden erfüllen die modellinhärenten Integritätbedingungen referentielle und *isa* (Subklassen-) Integrität sowie die benutzerdefinierten. Intern wird die Klassenextension durch geeignete Strukturen verwaltet.
Auf diesen Schnittstellen kann die Anwendungsprogrammierung aufsetzen und z.B. die spezifizierten benutzerdefinierten Methoden implementiert werden.

3.8 Animationskomponente und TL Interaktionsmodus

Aktion: Aufklappen des Instanzbrowsers und Zeigen auf den Knopf *interactive*
Die Animationskomponente stellt eine interaktive Datenbankbenutzeroberfläche zum Bearbeitung konkreter Anwendungsobjekte zur Verfügung. Die bereitgestellte Funktionalität benutzt die generierten Klassenschnittstellen sowie generierte Datenbankeditoren.

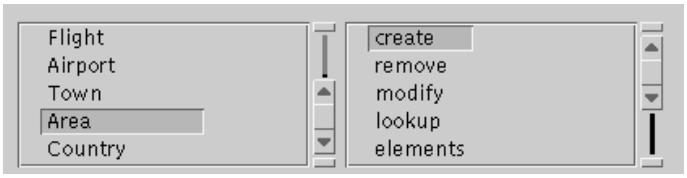
Aktion: Auswahl der Klasse *Area* und der Methode *elements* im Instanzbrowser, Drücken des *Apply* Knopfes



Durch Auswahl der Klasse *Area*, der Methode *elements* und Drücken des *Apply* Knopfes wird der Datenbankeditor für die Anzeige von *Areas* geöffnet, durch Drücken auf den Rollbalken kann durch die vorhandenen *Areas* gescrollt werden (fünf an der Zahl).

Aktion: im Fenstermenü des Areabrowsers Auswahl des Menüpunkts *Quit* Im dem TL Fenster werden darüberhinaus die hinter der Datenbankbrowserbenutzung stehenden Aufrufe der entsprechenden Editoren, die ihrerseits die generierten Klassenmethoden aufrufen, angezeigt.

Aktion: Auswahl der Klasse *Area* und der Methode *create* im Instanzbrowser, Drücken des *Apply* Knopfes (Aufklappen eines neuen *area-Input*-Editors).
Beim Erzeugen einer *Area* wollen wir die Verletzung der referenziellen Integrität zeigen, wir fügen hierzu die *Area Toscany* im Country *Italy* ein, das noch nicht als Land existiert.
Nach Drücken des Knopfes *Apply* im Area(!!) Editor erscheint die Fehlermeldung im Kommandofenster der interaktiven Tycoonmaschine.



Aktion: im Fenstermenü des Areaeditors Auswahl des Menüpunkts *Quit*

Aktion:
Drücken des *Interactive* Knopfes im Instanzbrowser und Öffnen der Datei `shortDemoInteractive.tm`; *copy* und *paste* der Dateidaten ins Kommandofenster der zweiten Tycoonmaschine.

Über den *Interactive* Knopf kann in eine interaktive TL Programmierumgebung gewechselt werden, von der aus die generierten Klassenmethoden (und Editoren, sofern importiert, in der Demoumgebung nicht der Fall) und alle weiteren in der Umgebung bereitgestellten Bibliotheksdienste zur Anwendungsprogrammierung und interaktiven Datenbankanfrage/-Verände-

rung herangezogen werden können.
Die in `shortDemoInteractive.tm` stehende Transaktion fängt die Verletzung der referentiellen Integrität ab (selbes Beispiel wie bei Editorbenutzung) und führt statt dessen (im `else` Zweig) erst die Erzeugung des *Country* und dann der *Area* erfolgreich durch.

```
try  area.create(tuple "Toscana" tuple "Italy" end end)
else country.create(tuple "Italy"  "mediterrian climate, good wine" end)
     area.create(tuple "Toscana" tuple "Italy" end end)
end;
```

4 Erweiterungen in der Langdemo

4.1 Einsatz eines generischen Dienstes zur IC-Überwachung mit Vorteil von IC-Änderungsunterstützung

- Intention:
  Anknüpfend an die Folie (der Fide-Demo) über Aufbau der generierten Schichten und dem Umgang mit ICs soll gezeigt werden, daß die Veränderung von IC-Überprüfungen allein durch die Neugenerierung und der Änderung der IC-Kollektionen in der IC-Überwachungskomponente möglich ist.

- Dazu das Beispiel:
  Aktion: Öffnen der Datei `longDemoInteractive.tm` und *copy* und *paste* des *create* Aufrufs Im *TL Interaktionsmodus* (der bereits oben durch Drücken des Knopfes *interactive* im Fenster des Instance Browsers eingeschaltet wurde) wird durch Aufruf der Funktion *create* versucht, eine neue *PackageTour* einzufügen, bei der das Hotel in „St. Andrews“ in der *Area* „Fife“ liegt, aber der Flug über „Glasgow“ geht, das in einer anderen *Area* liegt. Aufgrund der verschiedenen *Areas* scheitert das Einfügen.
  Aktion: Im zweiten Kommandofenster (in dem Tycoon mit dem Store `om1TracyTransIc` gestartet wurde) denselben *create* Aufruf (s.o.) tätigen Die Store Umgebung einer weiteren Tycoon Maschine in **diesem** Fenster enthält eine Lockerung der Bedingungen: Nicht die Area-Gleichheit, sondern nur noch Land-Gleichheit muß gelten. Das Einfügen der *PackageTour* mit derselben *create* Funktion wie oben ist in dieser Tycoon Maschine, die die neu generierten IC-Kollektionen besitzt, erfolgreich: „Glasgow“ (*Airport*) und „St. Andrews“ (*Hotel*) liegen im selben Land „Scotland“ (*Country*).

- Bemerkung:
  Lockerung der Bedingung kann sogar realitätsnah begründet werden. Wenn Reisegesellschaften ihre Buchungen nicht voll bekommen, lockern sie Bedingungen, um vor allem die von ihnen gebuchten Flüge auszulasten (da die der kostenkritische Faktor sind). Allerdings Vorsicht: wir haben das ganze etwas vereinfacht, da es sich hier um den Eintrag einer *PackageTour* und nicht einer Buchung (*ActualizedTour*) handelt. Falls Fragen kommen, sagen, daß ist nur zur Vereinfachung passiert, um das wesentliche zu zeigen. Ansonsten ist durch die Unterscheidung *PackageTour*, *ActualizedTour* in unserer Modellierung die Realität genau an diesem Punkt anwendungsgetreu nachgebildet (Verweis auf Ingrids Diss und Beeri-Paper).

4.2 Anwendungsprogrammierung auf generierten Schnittstellen

- Intention:
  Ausführung einer benutzerdefinierten und „per Hand“ programmierten Transaktion, die die generischen Klassenschnittstellen benutzt. Die Transaktion implementiert eine benutzerdefinierte Methode zum Löschen eines Hotels, die die in *PackageTour* definierte Transaktion *removePackageTourWithHotel* für alle Pauschalreisen, die dieses Hotel anbieten, aufruft. *RemovePackageTourWithHotel* bucht alle auf eine Pauschalreise (*PackageTour*) gebuchten Reisen (*ActualizedTours*) auf „vergleichbare“ Pauschalreisen (gleiche Ausstattung/ in derselben Region gelegen) um und löscht danach die *PackageTour* über die Standardlöschmethode. Das Löschen kann erst nach dem Umbuchen erfolgen, andernfalls würde es wegen Verletzung modellinhärenter Integritätsbedingungen (auf die *PackageTour* zeigen noch Referenzen) zurückgewiesen. Dies wird durch nachfolgende Anfragen, siehe Kommentare im Code, nachgewiesen.
  Aktion: Öffnen der Datei `longDemoInteractive.tm` und *copy* und *paste* des *create* Aufrufs

- **Aktion: Durch *copy* und *paste* die in der Datei `longDemoInteractive.tm` nach `Transaktionsdemo` aufgeführten TL Aufrufe der generierten Klassenschnittstellen in der Tycoonmaschine ausführen**. Erklärung der Aufrufe anhand des Transaktionscodes.

```
tracyTrans.deleteHotel(tuple "Calypso" tuple "WestCrete" end tuple "Chania"
  tuple "WestCrete" end end end dateDef.new(10 4 1993));

actualizedTour.lookup(tuple 1302 end);
(* geloescht, da Reisebeginn vor dem 10.4. *)

let actualizedTour1101 = actualizedTour.lookup(tuple 1101 end);

packageTour.getTourNo(actualizedTour.getPackageTour(actualizedTour1101));
(* Ergebnis 14, da ActualizedTour 1101 jetzt auf PackageTour 14 zeigt *)

let actualizedTour1303 = actualizedTour.lookup(tuple 1303 end);

hotel.getName(packageTour.getHotel(actualizedTour.getPackageTour(actualizedTour1303)));
(* Ergebnis: Atrion, da ActualizedTour 1303 jetzt auf PackageTour 16 zeigt,
   die das Hotel Atrion enthaelt *)


packageTour.lookup(tuple 11 tuple "Greece" end end);
(* nicht mehr vorhanden, geloescht, da Hotel Calypso nicht mehr vorhanden *)

tour.lookup(tuple 11 tuple "Greece" end end);
(* vorhanden, da nur PackageTour, nicht aber Tour geloescht wird *)


hotel.lookup(tuple "Calypso" tuple "WestCrete" end tuple "Chania"
```

```
            tuple "WestCrete" end end end);
(* Hotel ist nicht mehr in Klassenextension, wurde geloescht *)
```

Genauerer Kommentar zur Transaktion (über vorhandene und zu löschende Daten):

Folgende drei Hotels befinden sich bereits im Store `om1TracyTransIc`:

- *Hotel* "Calypso" in der *Town* "Chania" in der *Area* "WestCrete" im *Country* "Greece"
- *Hotel* "Atrion" in der *Town* "Chania" in der *Area* "WestCrete" im *Country* "Greece"
- *Hotel* "Rethymnon" in der *Town* "Rethymnon" in der *Area* "WestCrete" im Country "Greece"

Zu jedem Hotel gibt es zwei *PackageTours*, wobei die erste am Samstag von Hamburg startet und die zweite am Mittwoch von Muenchen. Die *PackageTours* 11 und 13 gehen ins Hotel "Calypso", 14 und 16 ins "Atrion" und 26 und 28 ins *Hotel* "Rethymnon".

Zu jeder *PackageTour* gibt es drei *ActualizedTours*, die durch eine vierstellige Zahl gekennzeichnet werden, wobei die ersten beiden Ziffern der Nummer der *PackageTour* entsprechen, die letzten beiden Ziffern 01, 02 oder 03 sind.

Beispiel: Löschen eines Hotels mit HotelKey und mit Datumsangabe, ab wann das Hotel gelöscht werden soll. Hier: Löschen des Hotels "Calypso" ab dem 10.4.1993.

Die Transaktion bewirkt folgendes:
Es werden zunächst alle für dieses Hotel gebuchten Reisen (*ActualizedTours*) herausgesucht. Diejenigen *ActualizedTours*, deren Reisebeginn vor dem Löschdatum des Hotels liegt, werden gelöscht. Für die restlichen wird alternativ eine *PackageTour* mit Flügen (*Flights*) am gleichen Tag ausgesucht, die nach Möglichkeit ein Hotel im gleichen Ort, ansonsten in der gleichen Region (*Area*) hat. Ist eine solche *PackageTour* nicht vorhanden, so wird die *ActualizedTour* gelöscht. Gibt es eine alternative *PackageTour*, so werden die Buchungen für das neue Hotel vorgenommen. Die alten Flüge werden storniert und die neuen Flüge gebucht. Wenn alle Umbuchungen erfolgt sind, werden die *PackageTours* mit dem zu löschenden Hotel gelöscht. Abschließend erfolgt das Löschen des Hotels.
Das konkrete Beispiel enthält folgende Daten und Datenänderungen:

- Die *ActualizedTours*, die das Hotel "Calypso" referenzieren, sind 1101,1102,1103,1301,1302 und 1303.
- Folgende von diesen werden gelöscht, da ihr Reisebeginn bereits vor dem 10.4. liegt: 1103 und 1302.
- Die *ActualizedTours* 1101 und 1102 erhalten als neue *PackageTour* die mit der Nummer 14, die das Hotel Atrion, welches im gleichen Ort wie das Hotel "Calypso" liegt, enthält.
- Die *ActualizedTours* 1301 und 1303 erhalten als neue *PackageTour* die mit der Numer 16, die ebenfalls Hotel "Atrion" hat.

4.3 Erweiterte Graphikdarstellung anhand eines weiteren Anwendungsbeispiels

4.3.1 Vorbereitung: Auswahl der *Court*-Anwendung

Aktionen im STYLETop Editor:

- im rechten Dateiauswahlfenster: *Court* auswählen.
- rechts unten den Knopf *OPEN Graphics* drücken und warten, bis das Diagramm vollständig angezeigt wird (Dauer ca. 15 Sekunden);
  Anzeige: Graphikeditorfenster mit Referenzdiagramm des *Court*-Beispiels (Klassenknoten, Referenzkanten, Rautensymbole für versteckte Werteknoten).

4.3.2 Kommentar

Der Graphikeditor zeigt ein zweites Anwendungsbeispiel, die Modellierung eines Gerichts mit angezeigten Klassen für Untersuchungen (*Investigation*), Verhandlungen (*Trial*), Fällen (*Case*), Richtern (*Judge*) Anklage (*Accuser*) etc.

- **Aktion: im Menü *Grid* den Menüpunkt *Extended* auswählen und darauf zeigen**;
  Eine engere Rasterung im erweiterten Rastermodus ermöglicht die Darstellung von noch mehr Beziehungen zwischen Klassen. Unterstützt wird vor allem die Darstellbarkeit von kaskadierenden Beziehungen einer Klasse – im Beispiel u.a. *Case* (Fall) – zu n weiteren Klassen – im Beispiel *PoliceDept, Accuser, Accusation* (Polizeibehörde, Ankläger, Angeklagter), zwischen denen keine weiteren Beziehungen bestehen.
- **Aktion: im Menü *Layers* den Menüpunkt *Layer3* auswählen**;
  Werteattribute und Vererbungsbeziehungen lassen sich ebenso wie im *Tracy*-Beispiel anzeigen und ausblenden.
  Aktion: im Menü *Scheme* den Menüpunkt *Inheritance Scheme* auswählen.