Dipl.-Inform. Thomas Ziemer
www.tfh-berlin.de/~ziemer
Stand: 5. September 2006

7 Anhang

WEITERE HINWEISE ZU BORLAND TOGETHER CONTROLCENTER

Ich danke Frau Ilse Schmiedecke für ihre nette Erlaubnis, die generelle Struktur ihres Skripts *Hinweise zur Modellierung mit Together* für diese Lehrunterlage verwenden zu dürfen.

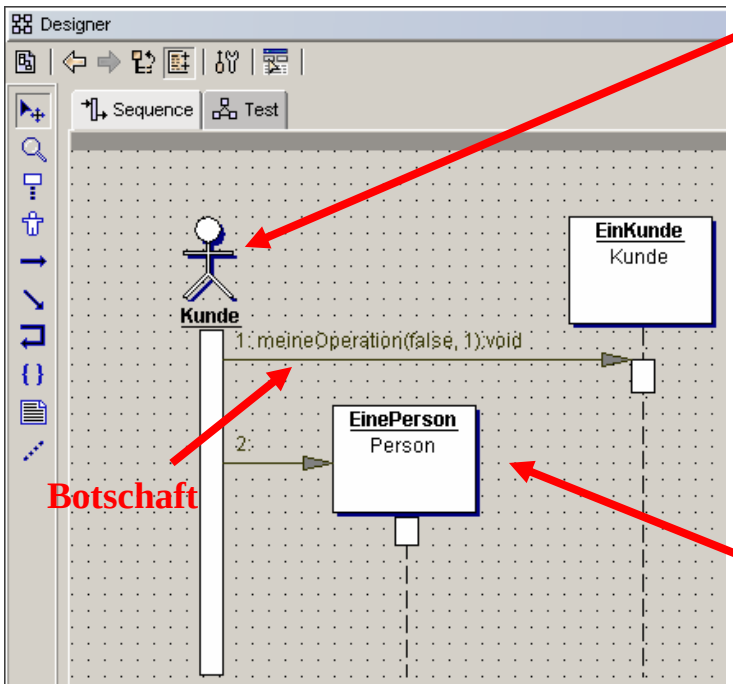
7.7 Einführung

Dieses Manuskript liefert weitere wichtige Hinweise zum Umgang mit dem Borland Together ControlCenter. Es setzt den ersten Manuskriptteil [SE I, 4. VL: Einführung in Borland Together ControlCenter] als verstanden voraus.

7.8 Sequenzdiagramm

Bisher enthalten die mit Together generierten Modelle nur diejenigen Operationssignaturen, die in den Klassendiagrammen spezifiziert wurden. Methodenrümpfe gibt es natürlich nicht, da keine Algorithmen modelliert wurden.

Erstellen Sie im Paket *DynamicModels* ein neues Diagramm des Typs *Sequence*. Aus der Werkzeugleiste am linken Rand des Designer-Fensters können nun Elemente wie Akteure (*Actor*), Objekte (*Object*), Botschaften (*Message*) und weiteres ausgewählt und im Sequenzdiagramm platziert werden.



Akteur

Akteure werden manuell modelliert und mit einem sprechenden Namen versehen (hier *Kunde*). Jedes Sequenzdiagramm benötigt mindestens einen Akteur, der den Einsprung in die Sequenz ermöglicht. Ein Akteur muss nicht immer eine Person bzw. funktionelle Rolle sein. Auch ein Fremdsystem (beispielsweise eine andere Software) kann einen solchen repräsentieren.

Objekt

Jedes im Sequenzdiagramm modellierte Objekt ist von einem bestimmten Typ. In der Abbildung ist *EinePerson* vom Typ *Person* und *EinKunde* vom Typ *Kunde*. Um den Typ auszuwählen, dient einmal mehr das Properties-Fenster des Inspektors. Sollte der Inspektor nicht sichtbar sein, so kann er über das Hauptmenü *View→Main Panes→Properties…* aktiviert werden.

Für die Eigenschaft *instantiates* ist die gewünschte Klasse im Pfad *Model→Analyse→ClassModels→…* auszuwählen. Somit ist der Objekttyp festgelegt. Together legt für jede im Sequenzdiagramm neu definierte Botschaft sogleich die entsprechende Operation im Klassendiagramm an. Für die Eigenschaft *name* ist ein sprechender Name für das Objekt zu wählen.

Botschaft und Operation

Eine Botschaft verbindet einen Akteur mit einem Objekt, oder auch zwei Objekte miteinander.

Die Benennung der Botschaft – die letztlich einen Operationsaufruf repräsentiert – erfolgt immer aus dem Kontextmenü des Diagrammpfeils heraus und wird niemals manuell am Pfeil eingetippt!

Falls in der mit dem Objekt korrespondierenden Klasse bereits eine entsprechende Operation definiert wurde, kann diese über *Choose Operation* ausgewählt werden. Gibt es die gewünschte Operation noch nicht, wird sie mittels *New→Operation* erstellt, und sollte empfohlenerweise sogleich mit einem sprechenden Namen versehen werden. Die Operation wird automatisch in die Klasse eingetragen. Überprüfen Sie es im Klassendiagramm und im Programmcode der Klasse.

Alle Aufrufparameter der Operation können im Operation-Fenster des Inspektors definiert werden. Die Eigenschaft *parameters* nimmt den Namen (*name*) und den Typ (*type*) aller Parameter auf, sowie die Eigenschaft *final*, die bestimmt, ob eine Operation in Unterklassen dieser Klasse überschrieben werden darf oder nicht.

Wurden alle Aufrufparameter definiert, ist es sinnvoll, den Inhalt anzugeben, den diese Parameter beim Versenden der Botschaft enthalten sollen. Hierzu dient die Eigenschaft *arguments* im Link-Fenster des Inspektors.

Soll ein neues Objekt mittels einer Botschaft erzeugt werden (Aufruf des Konstruktors), wird im Link-Fenster des Inspektors der Haken der Eigenschaft *creation* gesetzt. Ein Haken hinter *destruction* führt zur Zerstörung des Objekts – zur späteren Laufzeit der modellierten Anwendung, nicht im Modell.

7.9 Sequenzdiagramm und Kollaborationsdiagramm

Ein Sequenzdiagramm ist automatisch in ein Kollaborationsdiagramm überführbar und umgekehrt. Hierzu dienen die Menüeinträge *Show as Collaboration* bzw. *Show as Sequence* im Kontextmenü des jeweils anderen Diagrammtyps.

7.10 Codegenerierung aus dem Sequenzdiagramm

Nachdem das Sequenzdiagramm fertiggestellt ist, führt ein Klick auf den Eintrag *Generate Implementation* des Diagramm-Kontextmenüs zur Codegenerierung. Ist der Code nicht sichtbar, kann das Editor-Fenster über das Hauptmenü *View→Main Panes→Editor Pane* geöffnet werden. Der generierte Code entspricht genau den Eigenschaften des Sequenzdiagramms.

7.11 Reverse Engineering

Moderne CASE-Tools sind nicht nur komfortable Modellierungswerkzeuge, sondern stellen die Verbindung zwischen Modell und Programmcode in beiden Richtungen her.

Codegenerierung

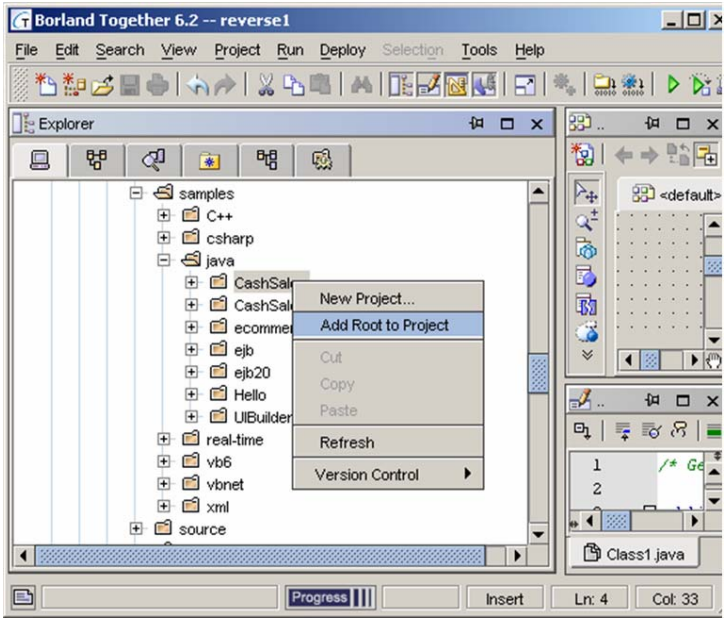
Aus einem Modell wird Programmcode generiert. Der Code ist typischerweise ausführbar, aber in dem Sinne unvollständig, dass Details, die durch das Modell nicht erfasst oder festgelegt sind, von Hand implementiert werden müssen.

Aus einem Klassendiagramm lassen sich z.B. abstrakte Klassen generieren, d.h. Klassen mit all ihren Vererbungsstrukturen und Attributen, aber nur mit Operationssignaturen. Üblicherweise werden stattdessen Klassenstümpfe (*class stubs*) mit leeren Methodenrümpfen generiert, in welche die Algorithmen später hineinprogrammiert werden.

Reverse Engineering

Aus vorhandenem Programmcode kann jedoch auch rückwärts ein Modell abgeleitet werden – typischerweise ein Klassenmodell.

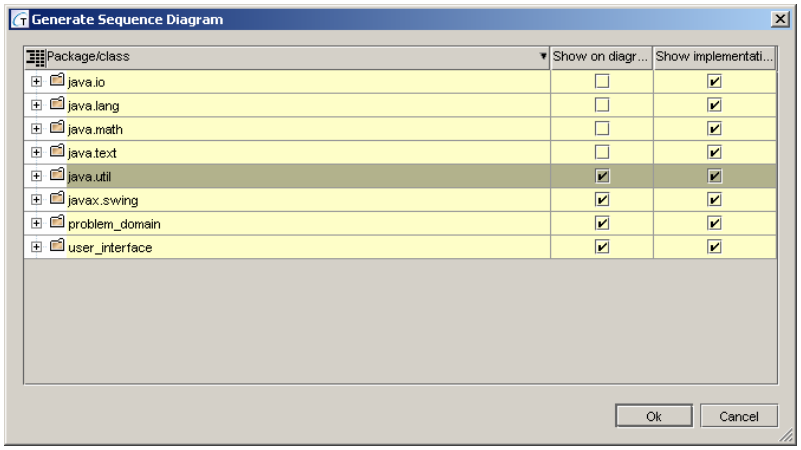
Der Zweck des Reverse Engineerings ist es, Änderungen struktureller Art nicht direkt auf Codeebene zu formulieren, sondern auf der abstrakteren Ebene des Modells zu planen. Außerdem stellt es eine gute Möglichkeit der Code-Dokumentation dar. So verschafft man sich durch Reverse Engineering leicht einen Überblick über ein komplexes Stück Software, das man aus irgendeinem Grund verstehen muss, etwa deshalb, um darin Änderungen vorzunehmen oder Fehler zu beseitigen.



Um die Klassenstruktur eines beliebigen Java-Programms zu importieren, erstellen Sie ein neues Projekt. Wählen Sie im Explorer-Fenster die Ansicht *Directory* (die Registerkarte mit dem Computersymbol) und navigieren Sie zu dem Verzeichnis, welches das darzustellende Java-Programm enthält. Im Kontextmenü dieses Verzeichnisses wählen Sie *Add Root to Project*.

Sie erhalten ein Paketdiagramm des entsprechenden Verzeichnisses. Ein Doppelklick auf ein Paket zeigt das dazugehörige Klassendiagramm.

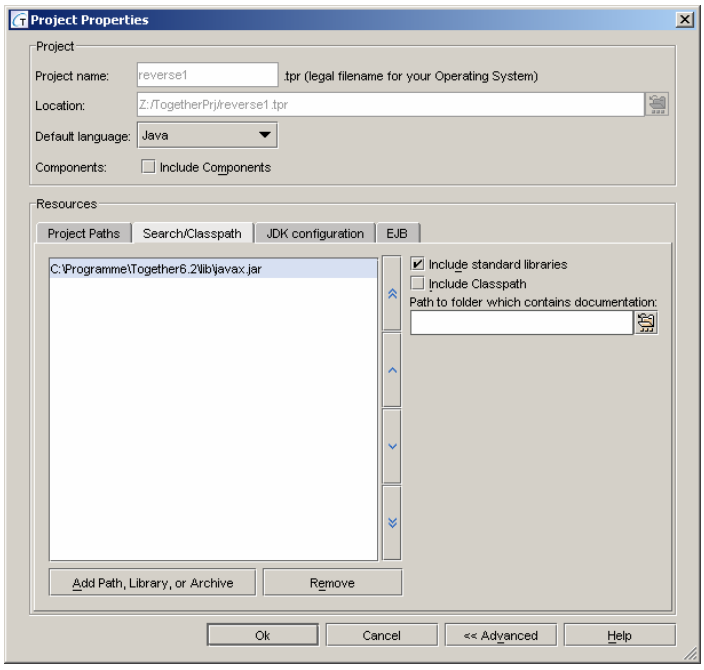
7.12 Generierung eines Sequenzdiagramms aus dem Programmcode



Um ein Sequenzdiagramm für eine Operation zu erzeugen, klicken Sie im Kontextmenü der Operation den Menüeintrag *Generate Sequence Diagram…* und wählen Sie in dem sich öffnenden Fenster *Generate Sequence Diagram*, aus welchen Paketen die Objekte im Sequenzdiagramm dargestellt werden sollen.

Bitte beachten Sie, dass das Sequenzdiagramm im gleichen Paket erstellt wird, in dem sich auch das Klassendiagramm befindet, und nicht im gewünschten Paket *DynamicModels*.

7.13 Codeausführung



Das Thema automatische Codegenerierung aus Klassendiagrammen und Ausführung des Codes wurde bereits im ersten Teil des Manuskripts beschrieben.

Together installiert zur Codeausführung zwar eine eigene Java-Umgebung. Im Sinne der Aktualität dieser Umgebung empfiehlt es sich jedoch, zusätzlich eine eigene Java-Umgebung zu installieren. In den Einstellungen *Project→Project Properties…* wird die Verzeichnisstruktur (*Project Paths*) und der Classpath der einzubindenden Bibliotheken festgelegt.

Die Wahl eines *Project Path* für die Java-Quelltextdateien (dazu die Schaltfläche *Add Path, Library, or Archive* anklicken, einen Pfad wählen und das Häkchen setzen bei *Java source files*) führt automatisch auch zur Anlage eines *classes*-Verzeichnisses, allerdings im Together-Installationsverzeichnis.

Auf der Registerkarte *Search/Classpath* werden darüber hinaus die gewünschten Java-Bibliotheken angegeben, beispielsweise *rt.jar* aus der aktuellen Java-Laufzeitumgebung.

7.14 Weitere Together-Einstellungen

In Klassendiagrammen der UML wird nicht die vollständige Signatur von Operationen angezeigt, sondern nur ihr Name. In den Einstellungen *Tools→Options→Default Level→View Management* kann die Eigenschaft *Diagram detail level* auf *Implementation* geändert werden. Folge davon ist jedoch auch, dass die einzelnen Klassen im Diagramm deutlich „breiter werden“.