Nettoverdienst

Together für Visual Studio .NET

von Karsten Strobel

Durch die Akquisition der Together Software Corporation, mit der Anfang 2003 ein Multi-Millionen-Dollar-Deal abgeschlossen wurde, hat Borland das eigene Produktportfolio um die unter dem Namen „Together“ schon im Tool-Markt eingeführten Softwaremodellierungs-Produkte angereichert. Wie der ganze Softwaremarkt musste auch diese neue Unternehmenssparte ihre Hausaufgaben machen und sich bemühen, mit der Evolution von .NET mitzuhalten. Und so brachte Borland gegen Ende 2003 eine gesonderte Edition für Microsofts Visual Studio auf den Markt, welche die Entwicklungsumgebung der Redmonder um UML-Funktionalitäten ergänzt, wobei man in der ersten Version den Schwerpunkt auf die Unterstützung von C# gelegt hat.

Um eine Sache völlig zu begreifen, muss man zunächst das Vokabular verstehen. Fachbegriffe und Abkürzungen, deren Bedeutungen unbekannt oder unklar sind, trüben den Durchblick. Bei einem Softwareprodukt fängt man logischerweise mit der Produktbezeichnung an. Gerade amerikanische Hersteller wählen gerne prägnante Namen, deren deutsche Übersetzung in unserem Sprachraum bisweilen einen Aha-Effekt auslöst. Man denke nur an Produkte wie „Fenster mit Ausblick“. Also habe ich im Internet auf einer Online-Übersetzungsseite die Worte „Together for Visual Studio .NET“ eingetippt und per Mausklick um eine Übertragung ins Deutsche gebeten. Das bemerkenswerte Resultat dieses Vorgangs lautete wie folgt: „Gemeinsam für sehendes Künstleratelier. Nettoverdienst“. Dem habe ich nichts hinzuzufügen. Der Name ist Programm!

Getting started

Wer wie ich gewohnheitsmäßig vor jeder Softwareinstallation den Inhalt der Installations-CD kurz begutachtet, findet bei der Together Edition for Visual Studio .NET [1] (die ich im Folgenden nur noch Together nennen werde) einen sehr übersichtlichen Vorrat an Dateien vor, nämlich das Setup-Programm und ein Getting Started-PDF-Dokument, das einem anhand eines neu zu erstellenden C#-Projekts den Umgang mit der Modellierungssoftware nahe bringen will. Nach der Installation sind verschiedene Menüs in VS.NET (das für diese Together-Edition natürlich Voraussetzung ist) um neue Einträge erweitert, die jedoch erst verfügbar werden, nachdem man die Registrierung vorgenommen und die Lizenz aktiviert hat.

Da die Dokumentation nur aus einer nicht allzu ergiebigen Kontext-Hilfe besteht, die sich in das Hilfesystem von VS.NET einnistet, ist das erwähnte Getting Started-Dokument der beste Einstiegspunkt, um sich mit dem Produkt vertraut zu machen. Hier lernt man zunächst, dass Together sich wie ein Plug-in in Visual Studio integriert und das Bedienkonzept der Entwicklungsumgebung mitbenutzt. Am auffälligsten ist dies beim Bearbeiten von UML-Diagrammen, wenn man die Diagrammelemente aus der Toolbox auf die Zeichenfläche zieht und die Elementeigenschaften mit Hilfe des Eigenschaftenfensters von VS.NET einstellt. Ebenfalls sehr eng ist die Bindung zwischen Together und VS.NET bei der Speicherung des Klassenmodells eines Projekts, denn Together greift hier fast ausschließlich auf den Quellcode zurück und verzichtet weitgehend darauf, ein eigenes Abbild des Modells in irgendeiner Form persistent zu speichern. Borland hebt diese als LifeSource bezeichnete Eigenschaft des Produkts besonders hervor und sieht darin den Königsweg zur ständigen Synchronisierung von Modell und Code.

Gegenständliche Schaubilder?

Schaltzentrale des Together Plug-ins ist das *Model View*-Fenster, das über das ANSICHT-Menü geöffnet wird und eine Baumansicht des aktuellen Projekts, der darin definierten Namensräume und der darin enthaltenen Klassen anzeigt. Dieser Baum ist der Darstellung des Objektbrowsers von VS.NET nicht unähnlich, enthält aber noch weitere, modellierungsspezifische Elemente. Dies sind vor allem die Diagramme, allen voran die Klassendiagramme, von denen es für jeden Namensraum zumindest eines gibt, in dem alle Klassen mitsamt ihrer Felder, Methoden und Eigenschaften symbolisch dargestellt werden und die hier auch modelliert werden können. Die etwas ulkige Bezeichnung Physical Class Diagram, die der Hersteller für diese Art von Diagramm gefunden hat, kann als Hinweis darauf verstanden werden, dass alle Klassen des Namensraumes hier zwangsweise dargestellt werden; entfernt man hier ein Klassensymbol, wird tatsächlich auch der entsprechende Quellcode aus dem Projekt gelöscht.

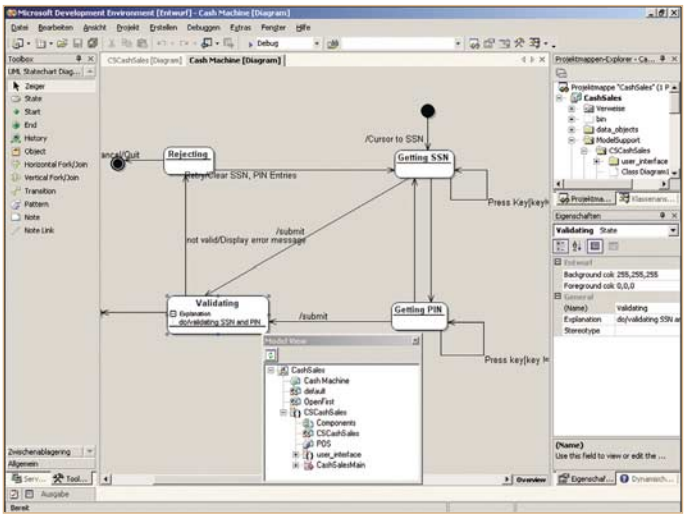

Abb. 1: State Chart-Diagramm

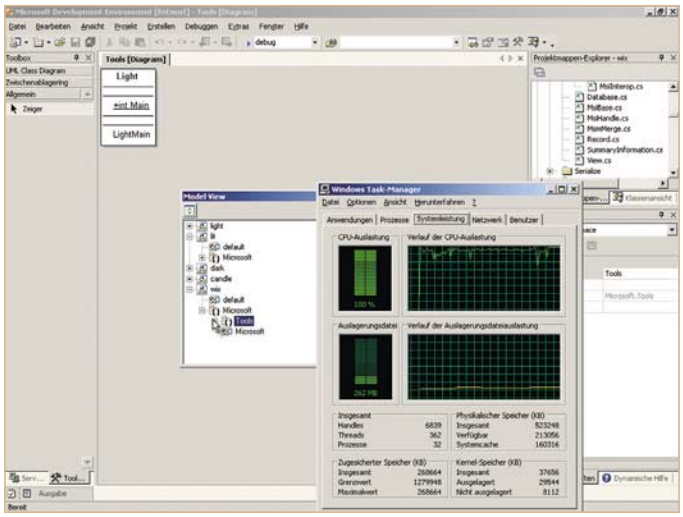

Abb. 2: Analyse umfangreicher Namespaces

Per Voreinstellung ist die Together-Unterstützung für alle neuen und auch für alle vorhandenen, nach der Installation geöffneten Projekte eingeschaltet und Together legt in jedem mit VS.NET bearbeiteten Projekt das Unterverzeichnis *ModelSupport* an. Hier werden aber nur die Layoutinformationen für die Diagramme in XML-Dateien gespeichert, denn die eigentliche Informationsbasis ist ja der Quellcode. Neben dem Klassendiagramm werden noch sieben weitere in UML gängige Diagrammarten angeboten, darunter Use Case-, Sequenz- und Aktivitätsdiagramme. Auch diese werden über das *Model View*-Fenster erstellt und lassen sich in die Baumhierarchie eingliedern. Das Bearbeiten der Diagramme gelingt – dank der nahtlosen Integration in VS.NET – fast ohne Einarbeitungsaufwand und ist dabei zwar komfortabel, aber das Fehlen einiger Funktionen wie des freien Positionierens von Text an Verbindern oder die nicht unterstützte Undo-Funktion macht einem doch zuweilen die Arbeit sehr schwer. Abbildung 1 zeigt ein State Chart-Diagramm, das den Vorgang des Geldabhebens an einem Bankautomaten illustriert.

Nur die Klassendiagramme sind unmittelbar mit Modell und Quellcode verbunden. Änderungen wirken sich direkt auf den Quellcode des Projekts aus. Auch wenn den übrigen Diagrammarten somit nur die Aufgabe der Illustration zufällt, kann man ihnen durch das Setzen von Hyperlinks eine gewisse Dynamik verleihen. Zum Beispiel lässt sich dadurch in einem Sequenzdiagramm ein Message-Symbol, das einen Methodenaufruf darstellt, direkt mit der aufgerufenen Methode verknüpfen, sodass per Mausklick zu dieser Methode in dem zugehörigen Klassendiagramm und mit einem weiteren Klick in den Quellcode gewechselt werden kann. Nicht unterstützt wird leider eine automatische Erzeugung von Sequenzdiagrammen, sodass man diese nützlichen Darstellungen per Handarbeit erstellen muss.

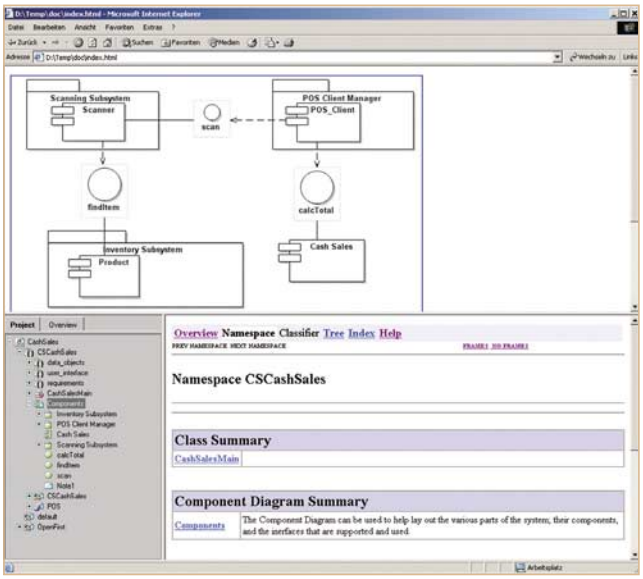

Abb. 3: Generierung von HTML-Dokumentation

Bitte warten…

Während die Software bei kleineren Projekten schnell genug ist, verursacht ein größeres Codevolumen durchaus einigen Rechenaufwand, der wahrscheinlich mit der oben erwähnten Strategie, die Modellinformationen nicht separat zu verwalten, sondern aus dem Quellcode zu entnehmen, in Zusammenhang steht. Zum Experimentieren habe ich unter anderem Microsofts Open Source-Software WiX (Windows Installer XML Tools [2]) verwendet. Abbildung 2 zeigt das Verhalten meines immerhin mit 2,8 GHz getakteten Notebooks beim Aufschlagen des Namensraumes *Microsoft.Tools* in Togethers *Model View*-Fenster. Die CPU wird deutlich über eine Minute lang weitgehend ausgelastet. Zwar ist dies ein einmaliger Aufwand, der jedoch zu wiederholen ist, sobald man das Projekt erneut lädt. Fraglich ist, ob durch solche Wartezeiten nicht die mit dem Werkzeug vielleicht zu erreichende Beschleunigung der Entwicklung wieder aufgezehrt wird.

Ein interessantes Feature sind die so genannten *Patterns*. Darunter versteht man Muster oder Vorlagen von Klassenarchitekturen und Implementierungen, die aus

einem mitgelieferten Fundus gewählt werden können. Im Lieferumfang enthalten sind sowohl einige von Together definierte Muster als auch eine Portierung von Musterlösungen der GoF (Gang of Four [3]). Mit ein paar Mausklicks lassen sich damit z.B. Lösungen für Class Factories, Iteratoren oder Proxy-Objekte über ein Klassendiagramm in das Modell einfügen und weiter ausgestalten, wobei sogleich der entsprechende Quellcode generiert und die Anwendung um einige bereits bewährte Objekte angereichert wird. Auch eigene Code-Kreationen kann man zur späteren Wiederverwendung in dieser Vorratskammer deponieren.

In dem wichtigen Bereich Code-Produktivität überzeugt die Software nur teilweise. In einem Klassendiagramm lassen sich zwar Objekte und deren Felder, Methoden und Eigenschaften recht einfach visuell ergänzen und ändern, was sich auch unmittelbar auf den Quellcode auswirkt. Typen, Modifizierer (*private*, *public* etc.) und Attribute von Objekten, Methoden usw. werden ebenso wie Kommentierungen über das Eigenschaftenfenster eingestellt oder können jederzeit auch direkt im Quellcode verändert werden, wobei die Synchronität mit den Diagrammen automatisch bewahrt wird.

Viele Wünsche bleiben jedoch offen. Zum Beispiel erhält man beim Ergänzen von Methoden keine Auswahl der überschreibbaren Methoden der Vorgängerklassen oder der zu implementierenden Interfaces. Ferner wird für jede neue Klasse automatisch eine neue Datei angelegt. Wünscht man eine andere Aufteilung, muss man den Quellcode in Handarbeit ändern. Und auch die veröffentlichte Liste der bekannten Einschränkungen ist in der aktuellen Version noch unerfreulich lang.

Ein Highlight ist die Generierungsfunktion für die HTML-Dokumentation. Für ein gewähltes Diagramm, einen Namensraum oder ein ganzes Projekt erzeugt Together auf Knopfdruck eine im Browser darstellbare, interaktive Dokumentation mitsamt Navigationsbaum und inklusive grafischer Wiedergabe der Diagramme. Abbildung 3 zeigt dies am Beispiel des Demoprojekts.

Fazit

Together for Visual Studio .NET verfolgt einen viel versprechenden Architekturansatz. Das Werkzeug ist eine nützliche Erweiterung von Visual Studio .NET, die konsequent die Integration in die Entwicklungsumgebung betreibt und daher relativ einfach anzuwenden ist. Auch der Preis von knapp 230 Euro ist angemessen. Leider weist die Version 1.0 noch einige Schwächen auf. In Kürze wird mit der Veröffentlichung der Version 2.0 gerechnet, die eine höhere Produktreife aufweisen sollte, sodass ein individueller Test dieser Software durchaus empfehlenswert ist.

Links & Literatur

[1] www.borland.com/together/msvs/index.html

[2] sourceforge.net/projects/wix/; siehe auch den Artikel „Das Unaussprechliche“ von Michael Seeboerger-Weichselbaum in *dot.net magazin* 6.2004

[3] c2.com/cgi/wiki?GangOfFour