

Evaluierung von UML-Modellierungswerkzeugen

Lutz Kirchner
Ulrich Frank
Universität Koblenz-Landau

Der sinnvolle und erfolgreiche Einsatz eines UML-Modellierungswerkzeugs im Kontext der objektorientierten Modellierung und Softwareentwicklung ist abhängig von vielschichtigen Randbedingungen. Um das optimale Werkzeug für das avisierte Einsatzszenario zu ermitteln, ist es notwendig eine fundierte Evaluationsphase durchzuführen. In dem Artikel wird ein Bezugsrahmen, dessen Zielsetzung die Unterstützung eines solchen Evaluationsvorgangs ist, vorgestellt und exemplarisch auf konkrete Werkzeuge angewandt. Dabei handelt es sich um einen Ausschnitt aus einer umfassenden Vergleichsstudie, die am Institut für Wirtschaftsinformatik der Universität Koblenz-Landau durchgeführt wurde.

1 Einführung

In der konzeptionellen Modellierung werden in zunehmendem Umfang objektorientierte Konzepte eingesetzt. Dabei kommt der Unified Modeling Language (UML) seit einigen Jahren eine herausragende Bedeutung zu. Allerdings ist die Anwendung der UML nur dann sinnvoll durchführbar, wenn entsprechende Werkzeuge die Modellierung in allen Phasen eines Softwareentwicklungszyklus unterstützen. Solche Modellierungswerkzeuge sollten einerseits die wichtigsten Sprachkonstrukte gemäß ihrer in der Spezifikation der UML festgelegten Syntax und Semantik unterstützen, andererseits aber auch weiterführend den Systementwurf sowie die Implementierung berücksichtigen. Am Markt wird mittlerweile eine Vielzahl von Werkzeugen dieser Art angeboten. Dabei handelt es sich zumeist um komplexe Software, die sich hinsichtlich ihres Funktionsumfangs wie auch ihrer Ergonomie erheblich unterscheiden. Die Auswahl eines angemessenen Tools ist deshalb für ein Unternehmen mit einem großen Aufwand und dem Risiko verbunden, erhebliche Fehlinvestitionen zu tätigen. Vor diesem Hintergrund zielt das Projekt EvaLUM (Evaluierung von UML-Modellierungswerkzeugen) der Forschungsgruppe Unternehmensmodellierung am Institut für Wirtschaftsinformatik der Universität Koblenz-Landau auf die Unterstützung einer zielgerechten Evaluierung von UML-Modellierungswerkzeugen. Dafür wurde ein komplexer Bezugsrahmen geschaffen, der als Grundlage zur Bewertung von UML-Modellierungswerkzeugen dienen soll. Seine inhaltliche Struktur ergibt sich aus kategorisierten Kriterien, die als Leitfaden durch die Evaluierung führen sollen. Es wird zwischen den folgenden übergeordneten Kategorien unterschieden:

- *Generelle Kriterien für Software*
- *Kriterien für objektorientierte Modellierungswerkzeuge*
- *Kriterien zur UML-Konformität*

Die erste Kategorie behandelt die für die Evaluierung von Software im allgemeinen gültigen Qualitätskriterien. Dazu zählen Kriterien, die die Bereiche *Wirtschaftliche Rahmenbedingungen*, *Ergonomie der Benutzungsoberfläche* und *Hilfestellungen für den Benutzer* abdecken. Als wirtschaftliche Rahmenbedingungen werden Anschaffungs- und Folgekosten verstanden, die sowohl durch den Erwerb der Software und der zum Betrieb nötigen Hardware als auch im Anschluss durch Nutzung des Supports entstehen. Die Kriterien der Ergonomie der Benutzungsoberfläche wurden auf der Grundlage von Gestaltungsrichtlinien für

Dialoge und die Präsentation von Informationen formuliert. Die dem Benutzer gebotenen Hilfestellungen werden durch die Erfassung entsprechender interaktiver Hilfsmittel oder einer Dokumentation gewertet, die Unterstützung bei der Installation und der Einarbeitung in die Software bieten.

Die zweite Kategorie befasst sich mit den spezielleren Anforderungen an ein objektorientiertes Modellierungswerkzeug. Dazu gehören zunächst solche Anforderungen, die unmittelbar mit der *Modellierung* verbunden sind, also Aspekte wie Hilfsmittel zur Erhaltung der Konsistenz eines Modells, die Ergänzung der Sprachmittel der UML um zusätzliche Diagrammartentypen sowie den Im- und Export von Modellen bzw. Teilmodellen. Im Hinblick auf die *Implementierung* werden Kriterien bereitgestellt, die der Bewertung von Codeerzeugung, Codeanalyse, der Unterstützung von Entwurfsmustern, Frameworks sowie Komponententechnologien dienen. Die Berücksichtigung von *Prozess* und *Projektmanagement* trägt dem Umstand Rechnung, dass die Entwicklung von Software im größeren Maßstab in der Regel im Rahmen eines Projekts und unter Berücksichtigung eines Softwareentwicklungsprozesses stattfindet. Somit sind neben der eigentlichen Erstellung der Software weitere Tätigkeiten – wie z.B. Personal- oder Anforderungsmanagement - auszuführen, die ein Tool ebenfalls unterstützen sollte. Eine Betrachtung von Metriken, der projektbegleitenden Dokumentationsmöglichkeiten und Strategien zur Daten- und Versionsverwaltung schließen diesen Themenbereich ab.

In der letzten Kategorie werden Sprachkonzepte für alle Diagrammartentypen sowie Erweiterungsmechanismen der UML, die ein Werkzeug im Rahmen der Modellierung anbieten kann, betrachtet.

2 Anwendung des Bezugsrahmens

Aus der Menge der Tools, die im Rahmen des Projekts EvaLUM getestet wurden, sollen im folgenden zwei exemplarisch vorgestellt und in einigen Aspekten genauer untersucht werden. Dabei bietet es sich an, möglichst unterschiedliche Werkzeuge auszuwählen, da auf diese Weise die Unterschiede der Produkte am Markt am deutlichsten dargestellt werden können. Das erste Tool, das wir an dieser Stelle vorstellen wollen, ist *Together* von *TogetherSoft* in der Version 6.0 (TogetherSoft Homepage s. [Toge]). Präsident und CEO von TogetherSoft ist der im Bereich der Objektorientierung weitreichend bekannte Peter Coad, der Anfang der 90er Jahre zusammen mit Edward Yourdon u.a. eine objektorientierte Methode zur Modellierung und Softwareentwicklung publiziert hat. Für die Topversion *Together ControlCenter* werden vom Hersteller u.a. die folgenden Kernfunktionalitäten genannt:

- Unterstützung aller UML-Diagramme bei der Modellierung
- Simultanes Round Trip Engineering
- Generieren von Dokumentation
- Refactoring
- EJB Unterstützung
- Metriken
- Entwurfsmuster
- Report Designer und Generator

Die oben aufgelisteten Features dokumentieren den Anspruch des Werkzeugs, ein komplettes Hilfsmittel für alle Aspekte der Softwareentwicklung zu sein. Abbildung 1 zeigt die Benutzungsoberfläche von Together.

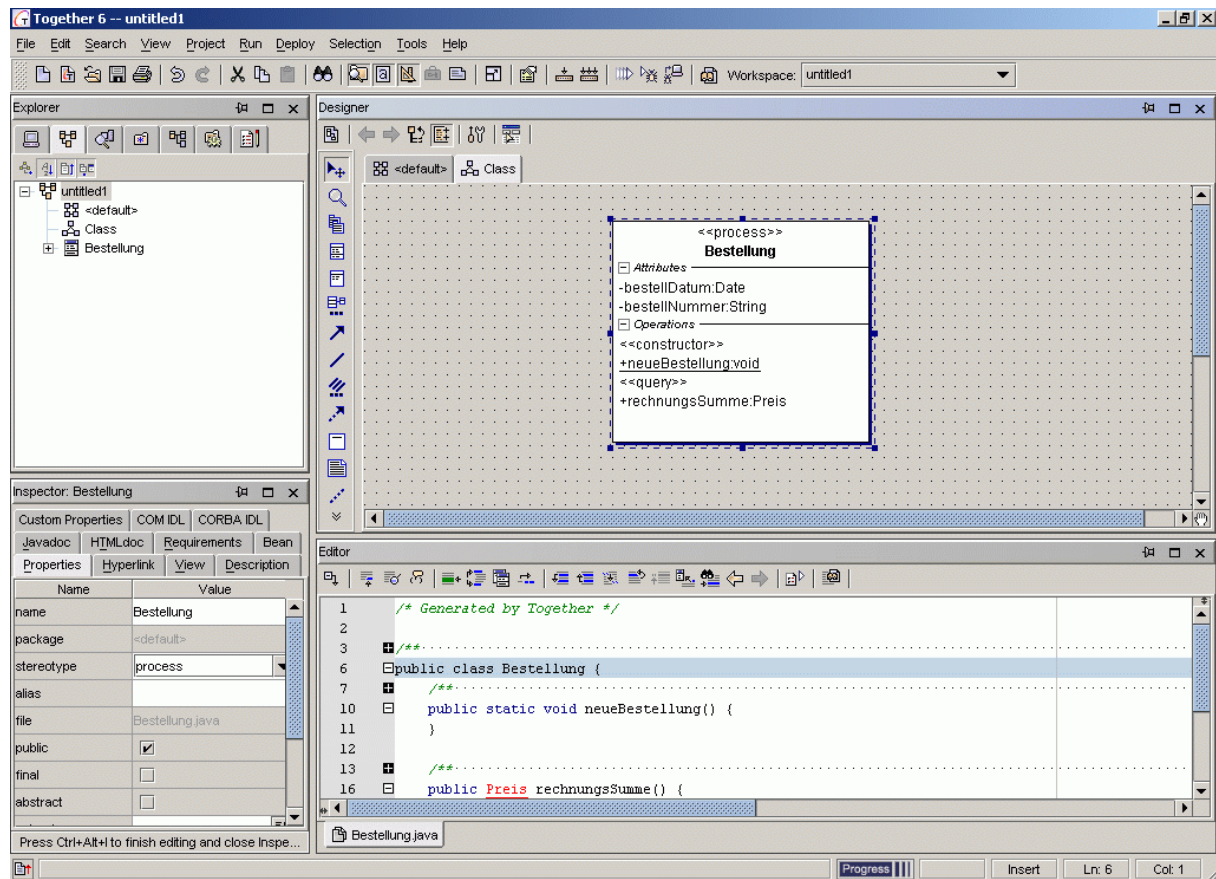


Abbildung 1: Together-Benutzungsoberfläche

Als zweites Tool betrachten wir ArgoUML in der Version 0.10, bei dem es sich im Unterschied zu den meisten anderen Tools nicht um kommerzielle Software handelt, sondern um ein noch im Beta-Stadium befindliches Open Source Projekt der *Tigris-Community* (ArgoUML Projekt Homepage s. [Argo]). ArgoUML ist konzipiert als ein Werkzeug zur Unterstützung des Entwurfs, der Implementierung und Dokumentation von objektorientierter Software. Sein funktionaler Kern kann für kommerzielle Zwecke lizenziert werden. Gegenwärtig ist das einzige kommerzielle Produkt, das auf diesem Grundgerüst aufbaut, *Poseidon für UML* von *Gentleware*. Als Kernfunktionalität von ArgoUML werden von Tigris folgende Punkte angegeben:

- Modellierung in UML
- Unterstützung von OCL
- Ablage des Modells in XMI
- Export von Grafiken in SVG
- Codegenerierung für Java
- Nutzung von Erkenntnissen aus der kognitiven Psychologie durch
 - Checklisten
 - ToDo-Liste
 - Präsentation von Sichten auf das Modell
 - Nutzung von *Design Critics*

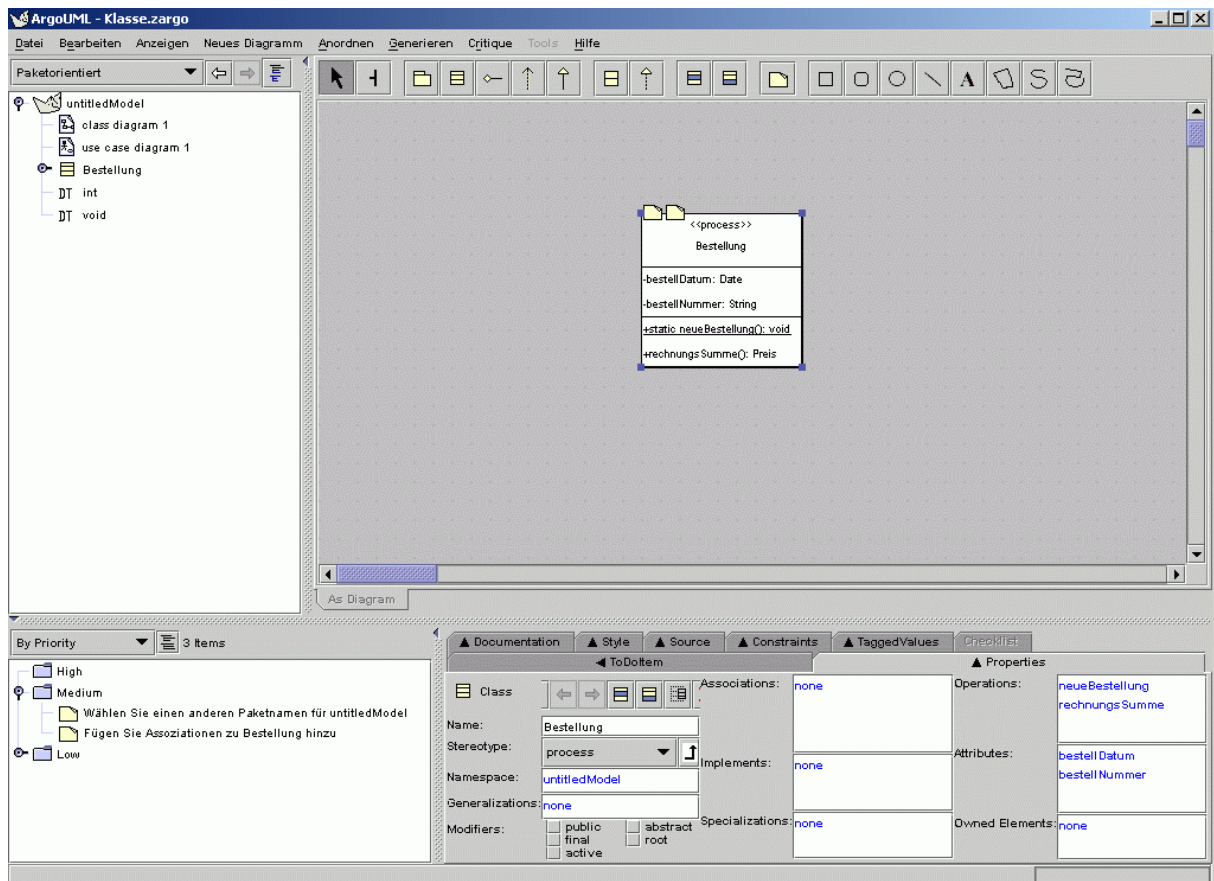


Abbildung 2: ArgoUML-Benutzungsoberfläche

Abbildung 2 zeigt die Benutzungsoberfläche von ArgoUML.

Die Anwendung des Bezugsrahmens auf die beiden Werkzeuge zeigt auf, welche Unterschiede Produkte, die im weitesten Sinne als Modellierungswerkzeuge angeboten werden, aufweisen können. Im folgenden werden drei Punkte des Bezugsrahmens jeweils stellvertretend für einen Aspekt der drei übergeordneten Kategorien etwas ausführlicher diskutiert und die Umsetzung durch die Tools analysiert. Dies betrifft die Programmdokumentation, das Round Trip Engineering sowie die Unterstützung von OCL.

2.1 Programmdokumentation

Die Dokumentation eines Tools ist grundlegend für den Nutzen, den die Software erbringen soll. Bewertet werden Art und Umfang der Dokumentation sowie die Eignung für die verschiedenen Zielgruppen (Administrator, Modellierer, Implementierer). Gängige Präsentationsformen der Programmdokumentation sind die klassische Papierform, sowie HTML- oder PDF-Dateien auf CD oder proprietäre Hilfesysteme. Als Ergänzung dazu sollte eine kontext-sensitive Online-Hilfe angeboten werden, mit der häufiger auftretende Bedienungsprobleme gelöst werden können oder die im Bedarfsfall zumindest auf das entsprechende Kapitel der eigentlichen Dokumentation verweist.

Together

Beim Erwerb von Together erhält man die Programmdokumentation in gebundener Form. Alternativ kann diese auch von der TogetherSoft Website in der jeweils aktuellsten Version im PDF-Format heruntergeladen werden. Ergänzend dazu gibt es eine Online-Dokumentation, die bei der Installation automatisch verfügbar ist. Diese basiert auf JavaHelp - einem auf Java

basierenden Mechanismus zur Erstellung von Online-Hilfesystemen - und bietet Volltextsuche und navigierbare Verknüpfungen (*hyperlinks*). Weiterhin kann über das Hilfemenü die Kurzdokumentation einiger mitgelieferter Module im HTML-Format aufgerufen und mittels Browser betrachtet werden. Eine eigentliche Online-Hilfe, die kontextbasiert ist, existiert nicht. Eine kontextbezogene Erklärung der Einstellungsmöglichkeiten findet man nur im Optionsmenü. Zusammenfassend ist hier festzustellen, dass die Dokumentation alle Aspekte, die die Handhabung des Werkzeugs betreffen, abdeckt. Die Einarbeitung wird auf diese Weise gut unterstützt.

ArgoUML

Die Programmdokumentation von ArgoUML ist auf der Tigris-Homepage online sowie dort zum Download verfügbar und liegt durchgehend im HTML-Format vor. Sie ist in mehreren Teildokumenten wie Quick Guide, User Guide u.a. organisiert. Eine Online-Hilfe innerhalb des Tools steht nicht zur Verfügung. Die Programmdokumentation ist vom Ansatz her relativ umfassend konzipiert, allerdings zum jetzigen Zeitpunkt in ihrer Brauchbarkeit aufgrund der noch sehr großen Lücken stark eingeschränkt. Ein endgültiges Urteil über die Qualität kann somit im Moment noch nicht abgegeben werden.

2.2 Round Trip Engineering

Im Kontext der Softwareentwicklung existieren verschiedene Ansätze des Vorgehens, die jeweils abhängig von der Zielsetzung des Entwicklungsvorgangs sind (vgl. [Balz98] und [Sevo00]). Beim *Forward Engineering* ist das Softwaresystem das Ergebnis des Entwicklungsvorgangs, d.h. bezogen auf ein Modellierungswerkzeug ist Forward Engineering der Vorgang der Quellcodeerzeugung aus dem Modell zum Zwecke der Erstellung einer lauffähigen Applikation. *Reverse Engineering* dagegen bezeichnet den Transfer der Zusammenhänge, die im Quellcode enthalten sind, auf ein höheres Abstraktionsniveau. Ziel ist hier nicht die Erstellung von Software, sondern deren Analyse und Darstellung in einer weniger implementation-sabhängigen Art und Weise - z.B. mittels eines Modells in UML.

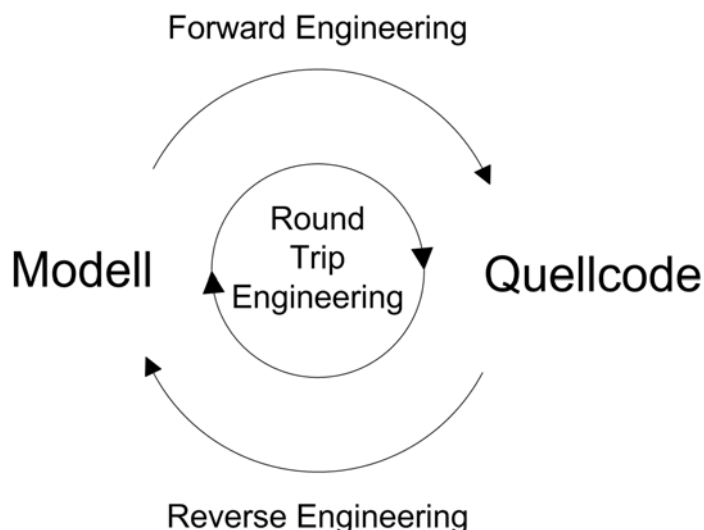


Abbildung 3: Round Trip Engineering, nach [Sevo00], S.159

Eine Modifikation des Quellcodes ist nicht Teil des Reverse Engineering. *Round Trip Engineering* schließlich bezeichnet die Möglichkeit, sowohl Änderungen im Modell mittels Forward Engineering im Quellcode umzusetzen, als auch Modifikationen im Quellcode durch Reverse Engineering ins Modell zu übernehmen (s. auch Abbildung 3). Ein Modellierungs-

werkzeug, das die Möglichkeit des Round Trip Engineering anbietet, muss Quellcode und Modell jederzeit durch Synchronisierungsmechanismen konsistent halten können. Überprüft wird nun für die jeweiligen Tools die Anzahl der im Round Trip Engineering unterstützten Programmiersprachen, die involvierten UML-Diagramme, generelle Strategien (simultanes oder explizites Round Trip Engineering) u.a.

Together

Together bietet eine Einbindung in das ausschließlich simultane Round Trip Engineering für Java, C++, C#, Visual Basic 6, VisualBasic.Net und CORBA IDL. Die Reichweite ist dabei direkt abhängig von der gewählten Zielsprache. Sowohl bei den Programmiersprachen als auch bei CORBA IDL werden Klassendiagramme und Quellcode bzgl. Namen und Deklarationen von Klassen - bzw. deren Entsprechungen in IDL -, Attributen und Methoden sowie Beziehungen bei jeder Änderung synchronisiert.

Zustandsdiagramme sind nicht in das Round Trip Engineering eingebunden. Interaktionsdiagramme dagegen können ausschließlich bei Java als Zielsprache - eine Kopplung mit den korrespondierenden Klassendiagrammen vorausgesetzt - in das Round Trip Engineering einbezogen werden. Somit ist die Änderung von Methodendeklarationen in Form der Modifikation von Nachrichten auch dort möglich und wirkt sich entsprechend auf die Klasse bzw. ihren Quellcode aus. Außerdem wird das Interaktionsdiagramm bei Änderung des Quellcodes neben dem Klassendiagramm bzgl. der verwendeten Bezeichner automatisch synchronisiert.

Zusammenfassend lässt sich feststellen, dass das Round Trip Engineering in den meisten Aspekten zuverlässig funktioniert. Allerdings ist der volle Funktionsumfang nur in Verbindung mit Java als Zielsprache zugänglich. Zu bemängeln ist in diesem Zusammenhang, dass beim Generieren von Quellcode aus Interaktionsdiagrammen Fehler, die während des Vorgangs auftreten und einen Abbruch desselben zur Folge haben, nur durch manuelle Korrekturen im Java-Quellcode wieder rückgängig zu machen sind. Auch wäre eine frühzeitige Überprüfung der Namenkonventionen bzgl. der Restriktionen der Zielsprache wünschenswert. Durch den GUI-Builder und den in der Fülle der Funktionen umfangreichen Quellcodeeditor stellt sich Together als relativ komplette Programmierumgebung – zumindest hinsichtlich Java - dar.

ArgoUML

Die einzige von ArgoUML im Round Trip Engineering unterstützte Programmiersprache ist Java. Sowohl beim Forward als auch beim Reverse Engineering wird nur das Klassendiagramm berücksichtigt.

Bei der Erstellung eines Elements in einem Klassendiagramm wird der Java-Quellcode in einem Ordner im Eigenschaftendialog angezeigt, kann aber nicht editiert werden. Dies ist für spätere Versionen geplant. Die eigentliche Codeerzeugung erfolgt explizit. Dabei können für jedes Diagramm aus einer Liste der Elemente diejenigen Klassen und Schnittstellen herausgesucht werden, welche die Basis zur Codeerzeugung bilden sollen.

Beim Reverse Engineering von JAVA-Dateien können über einen Dialog ausschließlich einzelne JAVA-Dateien ausgewählt werden. In diesem Dialog kann noch angegeben werden, ob Attribute einer Klasse Assoziationen realisieren und dahingehend ins Modell übernommen werden sollen. Ebenso besteht für Arrays die Auswahlmöglichkeit, diese als Datentypen (z.B. *int[]*) oder als Kardinalität eines Attributs zu übernehmen. Nach Abschluss des Vorgangs stehen die Klassen, Schnittstellen, Pakete, Beziehungen und Datentypen sowohl im Elementebrowser als auch in einem Klassendiagramm zur Verfügung. Letzteres allerdings nur in dem Fall, dass keine Fehler während des Prozesses aufgetreten sind.

Das Forward sowie das Reverse Engineering ist lediglich auf Elemente von Klassendiagrammen beschränkt und beinhaltet somit keinerlei dynamische Aspekte. Zudem ist die Implementierung noch fehlerhaft.

2.3 Darstellungsmöglichkeiten von Klassen

Durch die Modellierung von Klassen werden Gemeinsamkeiten der identifizierten Objekte so beschrieben, dass Friktionen zur späteren Implementierung möglichst vermieden werden. Die Berücksichtigung einiger spezieller Klassen findet besondere Beachtung bei der Evaluation der Modellierungswerkzeuge. Dies betrifft abstrakte Klassen, Hilfsmittelklassen und parametrisierbare Klassen. Weiterhin wird die Interpretation von abgeleiteten Attributen untersucht.

Together

Klassen werden mit Stereotyp, Name, Attributen, Methoden und inneren Klassendeklarationen dargestellt (s. Klasse in Abbildung 1 (Dateiname: Together_Screenshot.gif, Bildunterschrift: TogetherSoft Together Benutzungsoberfläche)). Es fehlen ein benutzerdefiniertes Feld, Name/Werte-Paare sowie Kardinalitäten Parametrisierbare Klassen können nicht modelliert werden.

Attribute sind mit Stereotyp, Sichtbarkeit, Name und Typ, allerdings ebenfalls ohne Kardinalitäten annotiert. Abgeleitete Attribute können nicht als solche gekennzeichnet werden, da ein führendes /-Zeichen im Namen eines Attributs nicht zulässig ist.

Operationen verfügen über Stereotype, Sichtbarkeiten, Namen, Parameterlisten und Rückgabewerte. Klassenattribute bzw. -methoden werden unterstrichen dargestellt. Weiterhin wird auf die Verwendung von *Properties* - womit in der Namenskonvention von Together Attribute bezeichnet werden, auf die über eine *get*- und/oder eine *set*-Methode zugegriffen wird - durch einen Eintrag in einem Feld unterhalb der Methoden hingewiesen.

Objekte als Instanzen einer Klasse werden ähnlich diesen, aber mit unterstrichenem Bezeichner und optional einem in eckigen Klammern notierten Zustand angezeigt.

In dem den Klassen zugehörigen Quellcode werden die *import*- und *package*-statements je nach Paketzugehörigkeit der Klassen korrekt ergänzt. Abhängigkeiten, die im Modell auf grafischer Ebene zwischen Paketen definiert werden, besitzen im Kontext des Forward Engineering keine besondere Semantik und werden weder bei Java in *package*- oder *import*-Anweisungen noch bei C++ auf andere Art im Quellcode umgesetzt. Beim Reverse Engineering werden ausschließlich Attributdeklarationen, die als Typ eine Klasse referenzieren, die in einem anderen Paket enthalten ist, als Abhängigkeit interpretiert und als solche im Modell dargestellt. Es wird weiterhin nicht überprüft, ob eine Klasse, die nicht als abstrakt gekennzeichnet ist, eine oder mehrere abstrakte Methoden enthält. Hilfsmittelklassen mit dem Stereotyp <<utility>> werden auf den Quellcode analog zu normalen Klassen abgebildet.

ArgoUML

Die Anzeige einer Klasse erfolgt mit Name, Stereotyp, Attributen und Methoden (s. Abbildung 2 (Dateiname: ArgoUML-Screenshot.gif, Bildunterschrift: Tigris ArgoUML Benutzungsoberfläche)). Im Falle einer abstrakten Klasse wird deren Name kursiv angezeigt. Die Attribute werden mit ihrer Sichtbarkeit, als Instanz- oder Klassenattribut (Schlüsselwort *static* und unterstrichen), als finale Version (*final*) sowie mit Name, Typ und Initialwert dargestellt. Operationen analog mit Sichtbarkeit, Instanz- oder Klassenattribut, *final*, Name sowie Parameter und Typ. Kardinalitäten innerhalb einer Klasse fehlen - ebenso fehlt ein benutzerdefiniertes Feld. Innere Klassen können zwar deklariert aber nicht angezeigt werden. Ihre Deklaration im Quellcode innerhalb der umgebenden Klasse ist korrekt umgesetzt. Die Zuordnung von Stereotypen zu Attributen und Methoden sowie Kardinalitäten zu Attributen ist

möglich, jedoch werden diese im Diagramm nicht angezeigt. Die Anzeige des Schlüsselworts *static* zusätzlich zur Unterstreichung eines Klassenattributs oder einer -operation ist redundant und nicht UML-konform. Abgeleitete Attribute sind als solche nicht zu erstellen, wobei das /-Zeichen aber angegeben werden kann. Da dieses Sonderzeichen in den Code übernommen wird, ist von der Verwendung allerdings abzuraten. Parametrisierbare Klassen werden nicht unterstützt, ebenso wenig die Modellierung von Klasseninstanzen (Objekten).

Zu bemängeln ist zudem, dass bei der Erstellung abstrakter Operationen die Klasse nicht automatisch ebenso als abstrakt gekennzeichnet wird. Auch die Umsetzung von Hilfsmittelklassen (Stereotyp <<utility>>) ist fehlerhaft, da diese entgegen der Spezifikation der UML auch Instanzattribute und -operationen enthalten dürfen. Für Attribute werden keine Zugriffsmethoden erstellt.

2.4 Ergebnisse

Die Evaluierung beider Werkzeuge unter Verwendung des Bezugsrahmens führte jeweils zu einem Gesamteindruck, der im folgenden dargestellt wird.

Together

Das Together ControlCenter stellt in Verbindung mit Java ein umfangreiches Werkzeug zur Unterstützung der Softwareentwicklung dar, wobei die Potentiale der Modellierung in hohem Maße von der für das Projekt gewählten Programmiersprache abhängen: Wenn statt Java eine andere der unterstützten Programmiersprachen verwendet wird, ist die Funktionalität des Tools deutlich eingeschränkt. Der Fokus des Werkzeugs ist hauptsächlich auf die Erzeugung von Java-Quellcode aus Klassendiagrammen und Interaktionsdiagrammen gerichtet. Die Modellierung mit der UML ist dabei ein Hilfsmittel, das in der Art seiner Anwendung und der Interpretation der Semantik der Modellelemente auf die Programmiersprache zugeschnitten und ihr untergeordnet ist. Viele der ermittelten Vorteile resultieren aus diesem Ansatz. Die meisten Nachteile ergeben sich einerseits aus der unzureichenden Unterstützung anderer Programmiersprachen, andererseits aus dem Umstand, dass die Mehrzahl der UML-Diagrammarten nur mit eingeschränkter Semantik abgebildet wird.

Zusammenfassend lässt sich feststellen, dass Together für die Software-Entwicklung mit Java eine umfangreiche Unterstützung anbietet, wobei sich die Modellierung der Semantik der Programmiersprache statt der UML unterordnet. Together ist in erster Linie als ein Werkzeug zur Erstellung von objektorientiertem Quellcode auf der Basis von Klassendiagrammen zu verwenden und weniger als ein Modellierungstool im Sinne einer methoden- resp. notations-treuen Analyse.

ArgoUML

Die Entwicklung von ArgoUML befindet sich noch im Fluss – ein Umstand, der bei der Bewertung des Werkzeugs zu berücksichtigen ist. Prinzipiell bietet das Werkzeug eine der UML-Semantik gut entsprechende Modellierungskomponente, wohingegen die Codegenerierung weniger überzeugen kann. Die relativ große Anzahl an festgestellten Mängeln ist vor allem darauf zurückzuführen, dass die Entwicklung noch nicht abgeschlossen ist. Weiterhin stellt es kein komplettes Softwareentwicklungswerkzeug dar, sondern ein Modellierungswerkzeug mit Codegenerierungskomponente. Die Stärken und somit die eigentliche Ausrichtung von ArgoUML liegen in der Unterstützung des Entwurfs. Dazu tragen zum einen eine detailliert ausgelegte - aber noch nicht ausreichend umgesetzte - grafische Modellierungskomponente sowie die rein syntaktische Überprüfung von OCL-Ausdrücken bei, zum anderen eine leistungsfähige Round Trip Engineering-Funktionalität. Auch die Analysephase wird durch die Modellierungskomponente unterstützt, gleichwohl hier noch einige Funktionen feh-

len, die in späteren Versionen ergänzt werden sollten. Im Hinblick auf die Implementierungsphase kann das Tool aufgrund der rudimentären Codeerzeugung – ausschließlich aus Klassendiagrammen – und der mangelnden Unterstützung von aktuellen Technologien (Datenbanken, Komponenten) weniger überzeugen.

3 Zusammenfassung und Ausblick

Der am Institut für Wirtschaftsinformatik der Universität Koblenz-Landau (s. [IWVI]) entwickelte Bezugsrahmen unterstützt eine detaillierte Bewertung von UML-Modellierungswerkzeugen nach Maßgabe verschiedener Beurteilungsdimensionen. Der Kriterienkatalogs unterstützt auch den Entwurf von Einsatzszenarien für Modellierungswerkzeuge. Der vollständige Bezugsrahmen ist in einer Diplomarbeit [Kirc01] dokumentiert. Angesichts der Komplexität der Modellierungswerkzeuge ist die Anwendung des ebenfalls umfangreichen Bezugsrahmens für viele Entscheider allerdings mit einem nicht tragbaren Aufwand verbunden. Aus diesem Grund wurden im Rahmen des Projekts EvaLUM insgesamt 7 UML-Modellierungswerkzeuge nach Maßgabe des Bezugsrahmens evaluiert. Zusätzlich zu denen in diesem Beitrag betrachteten Werkzeuge handelt es sich dabei um Systeme der Hersteller *Adaptive Arts*, *Elixirtech*, *Embarcadero*, *Rational* und *Telelogic*. Die Ergebnisse wurden multimedial dokumentiert. Entscheidungsträger können so nach ihren individuellen Präferenzen durch die Informationen zu den Werkzeugen und den Beurteilungskriterien navigieren. Da zudem eine interaktive Präsentation ausgewählter Nutzungsszenarien integriert ist, wird gleichzeitig ein gehaltvoller Eindruck vom *Look&Feel* der Werkzeuge vermittelt. Informationen zur vergleichenden multimedialen Evaluation der UML-Werkzeuge sowie ergänzende Publikationen finden sich unter [Eval]. Gegenwärtig wird in der Forschungsgruppe Unternehmensmodellierung auf der Grundlage eines weiteren Bezugsrahmens eine vergleichende Evaluation von Werkzeugen zur Geschäftsprozessmodellierung durchgeführt, deren Ergebnisse in Kürze ebenfalls in multimedialer Form dokumentiert werden.

4 Literatur

- [Argo] ArgoUML Project Home Page, Tigris Community, siehe <http://argouml.tigris.org>
- [Balz98] H. Balzert, Lehrbuch der Softwaretechnik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung, Spektrum Akademischer Verlag, 1998
- [Eval] Universität Koblenz-Landau, Institut für Wirtschafts- und Verwaltungsinformatik, Projekt EvaLUM, siehe <http://evalum.uni-koblenz.de>
- [IWVI] Universität Koblenz-Landau, Institut für Wirtschafts- und Verwaltungsinformatik, siehe <http://iwvi.uni-koblenz.de>
- [Kirc01] Entwicklung und Anwendung eines Bezugsrahmens zur Evaluierung von UML-Modellierungswerkzeugen. Diplomarbeit am Fachbereich Informatik der Universität Koblenz-Landau, 2001
- [Sevo00] J. Seemann, J. W. von Gudenberg, Software-Entwurf mit UML: Objektorientierte Modellierung mit Beispielen in Java, Springer, 2000
- [Toge] TogetherSoft Homepage, siehe <http://www.togethersoft.com>