49. Internationales Wissenschaftliches Kolloquium
Technische Universität Ilmenau
27.-30. September 2004

Tommy Baumann / Prof. Horst Salzwedel

Verwendung von UML-Modellen in einem integrierten Mission Level Design Flow zur Entwicklung von Hard- und Software

EINFÜHRUNG

Um komplizierte Sachverhalte besser zu verstehen und untersuchen zu können, ist es von großem Nutzen diese zu modellieren und zu systematisieren. Vor allem erspart man sich dadurch teure Tests in der realen Welt. Mit der fortschreitenden Entwicklung werden solche Systeme immer komplexer. Zum Beispiel verdoppelt sich die Komplexität elektronischer Systeme alle 1,5 Jahre. Daraus folgt, dass man zum Erhalt der Produktivität immer mehr „Human Resources“ benötigt, oder man findet einen Weg um diese effizienter einzusetzen. Mit Hilfe von Electronic Design Tools ist dies möglich, da Problemstellungen virtuell modelliert und danach simuliert werden können. Die Entwicklung solcher Software vollzog sich in mehreren Schritten, parallel zum Anwachsen der Komplexität in der realen Welt. Angefangen in den Siebziger Jahren mit dem „Physical Design“ (CAD), in den Achtzigern dem „Logic Level Design“ (Verilog, VHDL), in die Neunziger dem „Functional Level Design“ bis in die heutige Zeit zum „Mission Level Design“ [6]. Einhergehend mit der weiteren Abstraktionsebenenerhöhung ist es notwendig standardisierte Modelle und einen leistungsfähigen zyklischen Design-Flow zu verwenden. Das Electronic-Design-Tool „MLDesigner“ [3] bietet einen solchen Design-Flow zur Entwicklung von Hard- und Software an. Dabei wird auf bestimmte Diagramme der UML [4] zurückgegriffen. Des Weiteren wird die UML hauptsächlich von den vielfach verfügbaren CASE-Tools zur Entwicklung von Software benutzt. Eines der bekanntesten darunter ist „Rational Rose Realtime“ [5]. Da beide Softwaresysteme ähnliche Zielstellungen mit Hilfe der standardisierten UML verfolgen, ist es sinnvoll eine Schnittstelle zwischen ihnen zu entwickeln [2]. Damit können die Vorteile der beiden Softwaresysteme und deren unterschiedlichen Design-Flows miteinander kombiniert werden. Bei Electronic-Design-Tools liegt der Schwerpunkt auf der Modellierung und Simulation von Systemen zur Hard- und Softwareentwicklung, bei CASE-Tools beim Entwurf und Design von Software. Dieser Vortrag behandelt die Konzeption und Umsetzung einer Schnittstelle zwischen den Softwaresystemen „MLDesigner“ und „Rational Rose

Realtime“. Dabei wird auf die relevanten zu konvertierenden Informationen eingegangen, welche hauptsächlich in Form von UML-Diagrammen vorliegen.

MLDESIGNER

„MLDesigner“ [3] ist ein Simulations- und Modellierwerkzeug auf der Systemebene, welches beide Hauptbereiche (Architektur und Funktion) und die hauptsächlichsten Simulationsdomänen integriert. Die Modellierdomänen umfassen Discrete Event, Finite State Machine, drei Arten von Data Flow (Dynamic, Synchronous, Boolean) und Continuous Time/ Discrete Event. Diese Domänen können einzeln oder gekoppelt (für Mehrdomänen-Modelle) verwendet werden. Modelle werden graphisch, durch hierarchische Block Diagramme definiert. Alle Blöcke können parametrisiert werden, für einfache „what-if-Analyse“ um die Wiederverwendbarkeit zu maximieren. „MLDesigner“ bietet eine vollständige Entwurfsumgebung zum Modellieren von komplexen Systemen von einem Missions-Szenario [7], über Netzwerke, bis hin zur Signalform. Außerdem ist „MLDesigner“ einfach erweiterbar. Anwender können Entwurfsdomänen hinzufügen und Schnittstellen zu existierenden Werkzeugen bilden. Somit können eigene neue Werkzeuge entwickeln werden. Es werden hierarische Datenstrukturen zur Verfügung gestellt. Sockets bieten die Möglichkeit, dynamisch über Schnittstellen mit anderen Entwurfswerkzeugen mit internet-residenten Anwendungen, oder direkt über die Hardware zu kommunizieren. Eine Vielzahl von Konvertern ermöglicht den Datenaustausch mit „First-Generation-Products“, wie „Ptolemy“, „BONeS“ und „COSSAP“. Die mitgelieferte Anwendungsbibliothek, welche in der Grundausstattung aus über 2000 Design-Blöcken besteht, kann beliebig erweitert werden.

RATIONAL ROSE REALTIME

Das CASE-Tool „Rational Rose Realtime“ [5] wird zur Modellierung, Generierung und Simulation von Software verwendet. Alle Lebenszyklusphasen einer solchen Software werden durch den an die UML-Spezifikation angelehnten Design-Flow umgesetzt. In jeder Lebenszyklusphase werden bestimme UML-Diagramme verwendet, welche als Artefakte bezeichnet werden. Folgende vier Kategorien werden von „Rational Rose Realtime“ in Bezug auf die Lebenszyklusphasen unterschieden und nacheinander verwendet. Erstens der Use-Case-View. Dieser sagt aus was ein System tut, aber nicht wie dies intern umgesetzt wird. Zweitens der Logical-View. Hier wird der architektonische Prozess von Software-Modellierung, -Design und -Entwicklung dargestellt. Drittens der Component-

View, welcher die konkrete Ausprägung des Systems bzw. der Software beinhaltet. Dabei realisieren die Komponenten aktive Klassen und Datenklassen, um aus ihnen ein ausführbares Modell zu generieren. Viertens der Deployment-View. Dieser legt die Verteilung der Software bzw. des Systems auf verschiedene Prozessoren fest.

KONZEPT

Nach einer gründlichen Analyse der Informationsträger [1] und der an die Schnittelle gestellten Anforderungen, wurde ein Konzept zur Informationsübertragung entwickelt. „Rational Rose Realtime" verwendet zur Informationsspeicherung das durch „rtmdl"-Dateien repräsentierte „Petal"-Format. Die dort genutzten Datenstrukturen folgen einer bestimmten Grammatik. Aus diesem Grund kommen im ersten Schritt der Konvertierung ein Scanner und ein Parser zum Einsatz. Die dabei ausgelesenen Informationen werden in Form von Objekten abgelegt, dem „Rose Parser Object Model" (RPOM). Resultierend durch die zusätzliche Anforderung später XMI anbinden zu können, ist es außerdem notwendig eine gemeinsame Schnittstelle für das gewählte Petal-Format und XMI zu schaffen. Dies wird durch das „Rose Object Model" (ROM) erreicht, welches größtenteils ein Nachbau der Objekt-Struktur von „Rational Rose Realtime" ist. Im zweiten Schritt der Konvertierung werden die Informationen vom RPOM in das ROM transformiert. Die jetzt vorliegenden Informationen müssen wiederum transformiert werden. Im dritten Schritt der Konvertierung wird das ROM in die XML-Objekte des „MLDesigner" übertragen, mit deren Hilfe man danach „mml"-Dateien generieren kann (Abbildung 1). Ergänzend muss noch hinzugefügt werden, dass die Bestimmung der Top-Kapsel, welche im Gegensatz zu den anderen Kapseln zu einem System mit eingebetteten FSM-Primitive, anstatt zu einem Modul mit eingebetteten FSM-Primitive wird, nicht über den eben beschriebenen Weg beschafft werden kann. Um an diese Information zu gelangen muss die dazugehörige „rtusr"-Datei geöffnet und gelesen werden, was allerdings nur notwendig ist, wenn mehrere Komponenten mit einer gesetzter Top-Kapsel existieren. Ist das der Fall, so wird mittels der recht einfach aufgebauten Textdatei die aktive Komponente detektiert, indem man nach dem Schlüsselwort „Component" sucht. Direkt dahinter befindet sich der benötigte Identifikator.

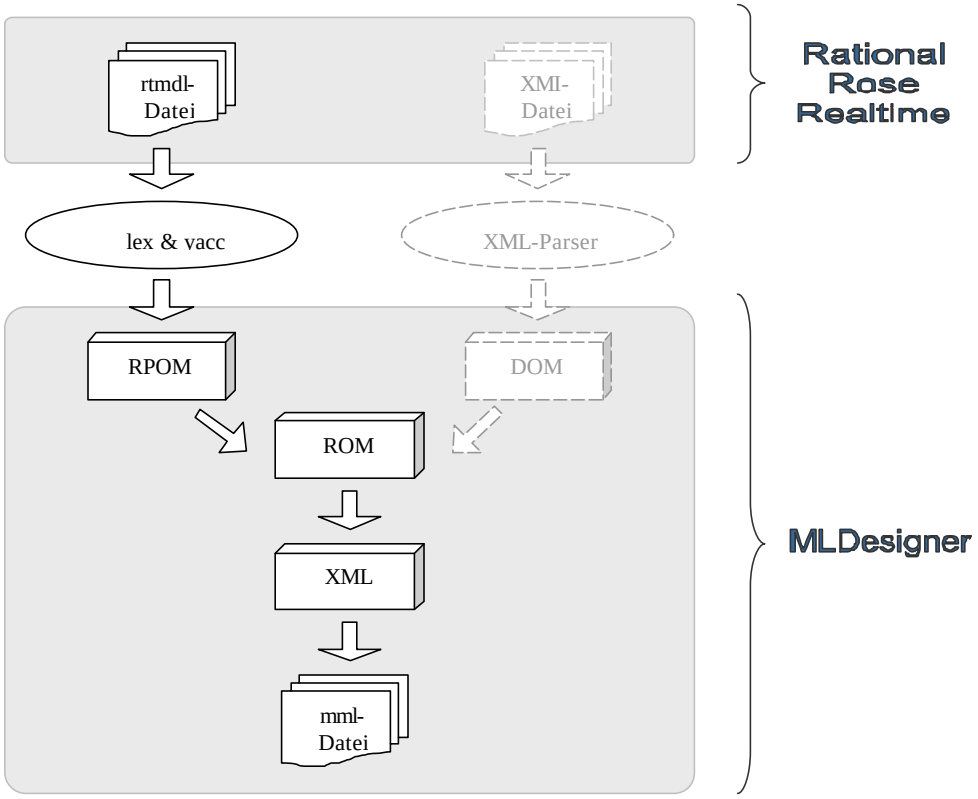

Abbildung 1: Konzept

ANWENDUNG

Durch den Konverter werden von „Rational Rose Realtime“ her hauptsächlich die Struktur- und Zustandsdiagramme der Kapseln, sowie die Signale der Protokolle übertragen. Außerdem werden optional vorhandene Vererbungshierarchien berücksichtigt. Im Einzelnen handelt es sich um Kapselrollen, Konnektoren, Ports, Signale, Zustände, Initialzustände, Choicepoints, Transitionen, Ereignisse, Aktionen, Guards und Dokumentationen. Bild (Abbildung 2) zeigt an einem Beispiel, wie ein Modell vor und nach der Konvertierung aussieht. Man kann gut erkennen, dass Struktur- und Zustandsdiagramm einer Kapsel aus „Rational Rose Realtime“, zu einem in ein Modul eingebettetes FSM-Primitive im „MLDesigner“ wird. Dabei beinhaltet das FSM-Primitive die Informationen aus dem Zustandsdiagramm, wie Zustände und Transitionen, und das Modul die Informationen aus dem Strukturdiagramm, wie Kapselrollen und Konnektoren. Ports werden im FSM-Primitive und im Modul angelegt, da sie einerseits durch das Modul die Schnittstelle nach außen repräsentieren und andererseits auch für das eingebettete FSM-Primitive notwendig sind, um Signale dorthin und von dort weiterzuleiten. Jeder Port in „Rational Rose Realtime“ verwendet ein Protokoll. Dort sind Eingangs- und Ausgangssignale definiert. Signale verwenden einen bestimmten Datentyp. Demzufolge

muss ein Port aus „Rational Rose Realtime“ zu einer Menge von Ports im „MLDesigner“ konvertiert werden, wobei die Mächtigkeit der Menge die Anzahl der Signale ist. Wenn beispielsweise ein Port in „Rational Rose Realtime“ ein Protokoll verwendet, in dem ein Eingangs- und zwei Ausgangssignale definiert sind, dann werden daraus im „MLDesigner“ ein Eingangsport und zwei Ausgangsports. Diese besitzen den Datentyp des entsprechenden Signals. Die Zustandshierarchie, welche in „Rational Rose Realtime“ mittels mehrerer Modeleditoren realisiert ist, wird durch den Konverter in das Single-Model-Editor-Konzept des „MLDesigner“ übertragen (Abbildung 3). Weiterführende Informationen kann man in der Diplomarbeit [2] finden. Durch die Anwendung des Konverters kann man ein Modell, welches mit dem „MLDesigner“ und dem dort forcierte Mission Level Design-Flow [7] modelliert wurde, durch zusätzliche Modelle aus „Rational Rose Realtime“ ergänzen. Die Vorteile beider Softwaresysteme werden dadurch miteinander verbunden, was eine effektivere Systemmodellierung zur Entwicklung von Hardware und Software ermöglicht.

Rational Rose Realtime MLDesigner

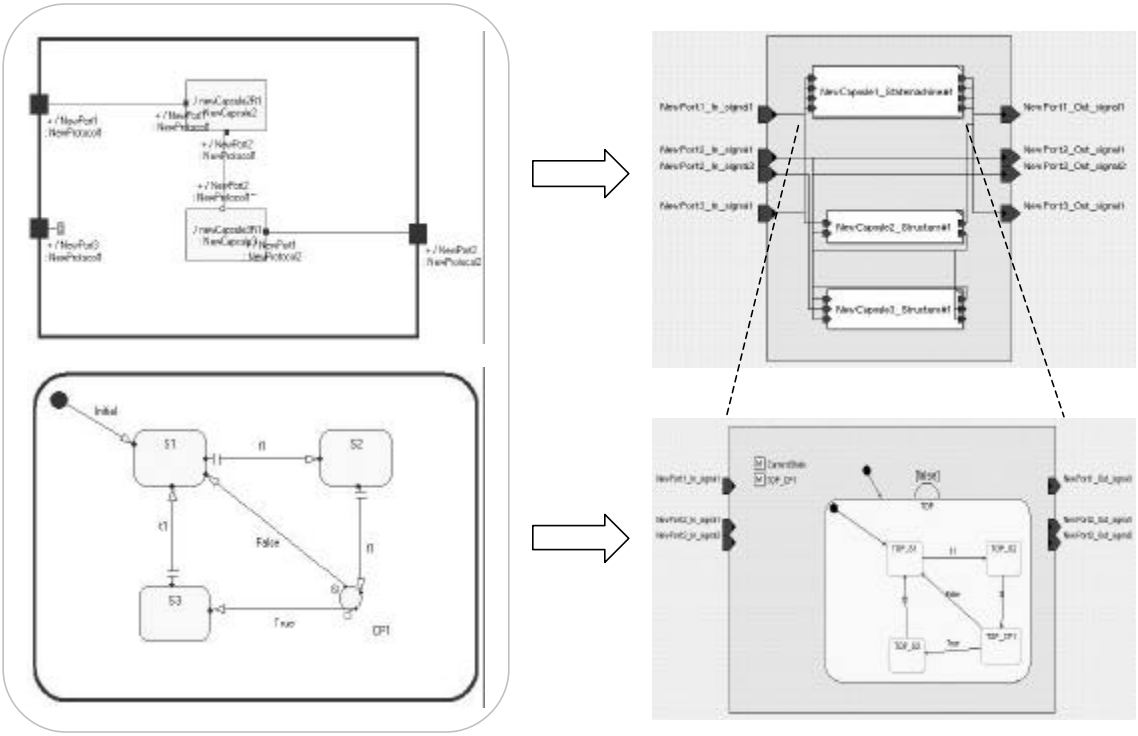

Abbildung 2: Kapselkonvertierung

Rational Rose Realtime MLDesigner

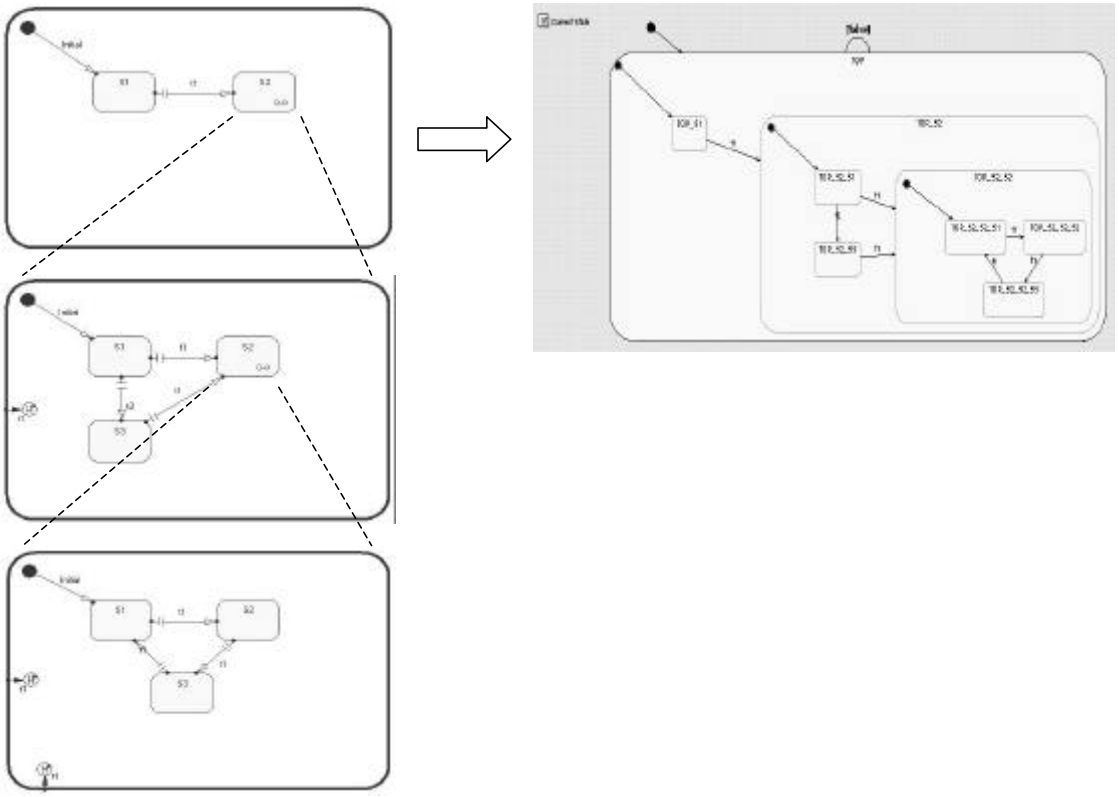

Abbildung 3: Zustandshierarchie

Literatur- bzw. Quellenhinweise:

[1] Tommy Baumann. *Methoden zur Aufwandsminimierung beim Generationswechsel von Softwaresystemen*. Studienarbeit, TU-Ilmenau, 2003.
[2] Tommy Baumann. *Integrierte Hard- und Softwareentwicklung mit dem MLDesigner – Entwicklung einer UML-Schnittstelle*. Diplomarbeit, TU-Ilmenau, 2004.
[3] *MLDesigner Dokumentation*, Version 2.5, 2004. http://www.mldesigner.com.
[4] Object Management Group. *Unified Modeling Language Specification*. 2001. http://www.omg.org.
[5] *Rational Rose Realtime Online Hilfe* , Version 2003.06.00.436.000, 2003. http://www.rational.com.
[6] Gunar Schorcht. *Entwurf integrierter Mobilkommunikationssysteme auf Missionsebene.* Dissertation, TU-Ilmenau, 2000.
[7] Horst Salzwedel. *Design Technology Development Towards Mission Level Design*. 49th International Scientific ColloquiumTU-Ilmenau, 2004.

Autorenangabe(n):

Tommy Baumann
Prof. Horst Salzwedel
Technische Universität Ilmenau, Max-Planck-Ring 14, PF 100565
98684, Ilmenau
Tel.: (+49)3677/69-2767
Fax: (+49)3677/69-1285
Email: tommy.baumann@gmx.de