Zur Entwicklung zuverlässiger Informationssysteme in KorSo*

S. Conrad, G. Denker, M. Gogolla, R. Herzig, N. Vlachantonis, H.-D. Ehrich

Technische Universität Braunschweig, Informatik, Abt. Datenbanken
Postfach 3329, D-38023 Braunschweig, Germany
e-mail: `conrad@idb.cs.tu-bs.de`

Zusammenfassung

Innerhalb des KorSo-Projektes verfolgt unsere Gruppe das Ziel, die Entwicklung zuverlässiger Informationssysteme auf der Basis formaler Spezifikationen zu unterstützen. Dazu konzentriert sich unsere Arbeit auf die Spezifikationssprache TROLL *light*, die es erlaubt, Teile der zu modellierenden Welt als eine Gemeinschaft nebeneinander bestehender und miteinander kommunizierender Objekte zu beschreiben. Auf diese Weise bestimmen wir sowohl die Struktur als auch das Verhalten der konzeptionellen Objekte. Unsere Spezifikationsumgebung für TROLL *light* erlaubt die Animation von Spezifikationen ebenso wie das Beweisen von Eigenschaften der Spezifikationen unter Verwendung von Theorembeweisern.

1 Einleitung

In den letzten Jahren wurden unterschiedlichste Ansätze zur Spezifikation komplexer Softwaresysteme vorgeschlagen, z.B. Spezifikation von Funktionen (VDM, Z), abstrakte Datentypen, Prädikatenlogik und ihre Erweiterungen wie temporale und modale Logik, semantische Datenmodelle und Ansätze zur Prozeß–Spezifikation (CCS, CSP, Petri–Netze).

Jede dieser Spezifikationsmethoden scheint aber für sich alleine betrachtet nicht für die konzeptionelle Modellierung von Informationssystemen angemessen zu sein. Denn das Ziel der konzeptionellen Modellierung ist es, eine erste präzise Beschreibung des zu modellierenden Weltausschnitts — die sogenannte Diskurswelt (engl. Universe of Discourse) — zu liefern. Betrachtet man aber eine solche Diskurswelt näher, wird klar, daß sie aus komplex strukturierten Entitäten mit zeitabhängigem Verhalten besteht. Die Entitäten existieren nebeneinander und kommunizieren miteinander. Die Idee der objektorientierten Spezifikation ist es nun, Entitäten der realen Welt als formale Objekte in einer Spezifikationssprache darzustellen. Erste Vorschläge für Sprachen zur Spezifikation von Objekten sind beispielsweise OBLOG und CMSL. In unserem Projekt arbeiten wir mit der Spezifikationssprache TROLL *light* [CGH92], ein Dialekt der Sprache TROLL [JSHS91]. In TROLL *light* kann man sowohl die Struktur als auch das Verhalten von Entitäten der realen Welt charakterisieren.

Das Ziel des KorSo–Projektes, in dem wir als Teilvorhaben eingebunden sind, ist es, formale Methoden und Techniken für den Entwurf von Softwaresystemen hoher Qualität zu entwickeln, zu verbessern und zu konsolidieren. Daher beschäftigt sich das Gesamtprojekt mit Fragestellungen wie der angemessenen Darstellung der informellen Anforderungen, der schrittweisen Entwicklung und Verfeinerung des Entwurfs sowie der Validation und

*Die hier vorgestellte Arbeit wird vom Bundesministerium für Forschung und Technologie unter der Förderungsnummer 01 IS 203 D (KorSo = Korrekte Software) unterstützt.

Verifikation von Resultaten. Unser Teilvorhaben behandelt die Entwicklung zuverlässiger Informationssysteme auf der Grundlage von TROLL *light*–Spezifikationen. Das Ziel der vorliegenden Arbeit ist es, das Spektrum unserer Aktivitäten zu skizzieren, die vom Sprachentwurf über Implementierungsaspekte bis hin zu Fragen der Logik und Semantik reichen. Die Semantik unserer Sprache basiert auf theoretischen Grundlagen, die in einer Reihe von Artikeln (z.B. [EGS92, SE91]) entwickelt wurden. Daneben haben wir auch andere semantische Ansätze betrachtet. So skizzieren wir in [VHG+93] eine Transformation von TROLL *light*–Spezifikationen in die logische Programmiersprache Maude, die Nebenläufigkeit mit Objektorientierung verbindet.

Es gibt eine Reihe anderer Forschungsprojekte, die sich mit ähnlichen Fragestellungen befassen. Das Ziel des Verbundprojektes STONE ist die Entwicklung einer Software–Entwicklungsumgebung für Schulungszwecke. Eine Grundidee in STONE ist, daß das objektorientierte Paradigma auf alle Teile des entwickelten Systems Anwendung findet. Ein weiteres Resultat des Projektes ist das objektorientierte Datenbanksystem OBST. Das Ziel des DAIDA–Projekts besteht darin, eine Software–Entwicklungsumgebung für den Entwurf und die Implementierung von Informationssystemen zu entwickeln. Dieser Ansatz wurde vom IRIS–System fortgesetzt, das Entwürfe für Informationssysteme aus Anforderungsspezifikationen generiert. Außerdem gibt es einen Ansatz zur Übersetzung eines semantischen Datenmodells in objektorientierte Datenbanksysteme. CADDY ist ein CASE–Tool, das auf einem erweiterten Entity–Relationship Modell basiert. Es integriert Werkzeuge zum konzeptionellen Entwurf von Datenbanken und zum Prototyping von Datenbank–Schemata.

Der Rest der vorliegenden Arbeit gliedert sich wie folgt. In Abschnitt 2 werden die Konzepte unserer Spezifikationssprache TROLL *light* kurz skizziert. In Abschnitt 3 wird die Bedeutung von Zertifikation, also Validation und Verifikation von Spezifikationen, diskutiert. Abschnitt 4 stellt kurz die TROLL *light*–Entwicklungsumgebung für den Entwurf von Informationssystemen vor. Im letzten Abschnitt wird der Stand der Implementierung der TROLL *light*–Entwicklungsumgebung dargestellt.

2 Konzepte von TROLL *light*

TROLL *light* ist eine Sprache zur Beschreibung von strukturellen und dynamischen Eigenschaften von Objekten. Die Beschreibung von strukturellen Eigenschaften orientiert sich in weiten Teilen an semantischen Datenmodellen.

- Die in einem Zustand beobachtbaren Eigenschaften eines Objektes werden durch Attribute beschrieben. Einfache Attribute sind daten– oder objektwertig. Darüber hinaus erlauben vordefinierte Sortenkonstruktoren wie *set* oder *tuple* auch die Spezifikation komplexer Attributbereiche.

- TROLL *light*–Objekte sind in Objekthierarchien, die sich aus Unterobjektbeziehungen ergeben, organisiert. Eine Unterobjektbeziehung ist hierbei als eine exklusive „Teil von“– bzw. Komponentenbeziehung zu verstehen.

- Normalerweise werden Attributwerte direkt durch das Eintreten gewisser Ereignisse bestimmt. Daneben ist es auch möglich, abgeleitete Attribute zu spezifizieren, d.h. Attribute, deren Inhalte sich aus anderer gespeicherter oder abgeleiteter Information bestimmen. Zur Formulierung von Ableitungsregeln stellt TROLL *light* einen SQL–ähnlichen Anfragekalkül zur Verfügung.

- Der Anfragekalkül von TROLL *light* unterstützt auch die Spezifikation statischer Integritätsbedingungen.

Die Beschreibung von dynamischen Eigenschaften basiert auf der Spezifikation von Ereignissen. Ereignisse sind Abstraktionen von zustandsverändernden Operationen auf Objekten.

- Objektereignisse werden durch eine endliche Menge von Ereignisgeneratoren beschrieben. Jeder Ereignisgenerator kann mit einer Liste von Parametersorten versehen sein. Ein Ereignisgenerator mit aktuellen Parameterwerten liefert ein Ereignis.
- Die Auswirkung von Ereignissen auf Attribute werden durch Auswertungsregeln beschrieben.
- Ereignisse in verschiedenen Objekten können durch Interaktionsregeln synchronisiert werden.
- Die möglichen Ereignisfolgen können mit Hilfe CSP-ähnlicher Prozeßbeschreibungen auf zulässige Folgen eingeschränkt werden.

Den Rahmen zur Beschreibung all dieser Objekteigenschaften bilden *Templates*. Templates besitzen folgenden Aufbau:

```
TEMPLATE
  DATA TYPES    Deklaration von verwendeten Datentypen
  TEMPLATES     Deklaration von anderen verwendeten Objektbeschreibungen
  SUBOBJECTS    Unterobjektbeziehungen
  ATTRIBUTES    Attribute
  EVENTS        Ereignisgeneratoren
  CONSTRAINTS   statische Integritätsbedingungen
  VALUATION     Auswirkung von Ereignissen auf Attribute
  DERIVATION    Ableitungsregeln für abgeleitete Attribute
  INTERACTION   Synchronisierung von Ereignissen in verschiedenen Objekten
  BEHAVIOR      Beschreibung zulässiger Lebensläufe
END TEMPLATE
```

Neben den Bemühungen, allgemeine theoretische Grundlagen für objektorientierte Sprachkonzepte zu entwickeln, sind auch speziell für TROLL *light* semantische Untersuchungen gemacht worden. Vielversprechend erscheinen die Arbeiten, die auf dem Gebiet der Nebenläufigkeit im Zusammenhang mit Objektorientierung von Meseguer erstellt wurden. Auf dieser Basis wurde von Meseguer die logische Programmiersprache Maude entwickelt, die auf einer fundierten Modelltheorie basiert. Einer unserer Ansätze zur Semantik von TROLL *light* beschäftigt sich deshalb mit der Übersetzung von TROLL *light*–Konzepten nach Maude, um die Vorteile der zugrundeliegenden Theorie auszunutzen.

3 Zertifikation

Ein Schwerpunkt unseres Projektes ist Zertifikation. Unter diesem Begriff werden verschiedene Aspekte zusammengefaßt: Validation, Konsistenz und Verifikation. Mit diesen Aspekten beschäftigen wir uns, um die Entwicklung zuverlässiger Informationssysteme innerhalb der oben vorgestellten Umgebung zu gewährleisten. Im folgenden erläutern wir kurz unsere Vorstellungen zu Validation und Verifikation. Konsistenzprüfungen sind natürlich für alle Entwicklungsschritte notwendig, um frühzeitig Hinweise auf einfache Fehler zu bekommen.

Validation bedeutet, die entworfene Spezifikation zu testen und dabei festzustellen, ob sie eine angemessene Abbildung des modellierten Weltausschnittes darstellt. Dies geschieht mit Hilfe eines Animationssystems.

Unter Verifikation verstehen wir das Beweisen von Eigenschaften spezifizierter Objekte. Dies ist nötig, da man in der Regel nicht alle beabsichtigten Eigenschaften von Objekten direkt in ihre Spezifikation schreibt. Dies würde zu sehr umfangreichen Spezifikationen führen, die letztendlich nicht mehr lesbar wären. Außerdem wären dadurch viele Eigenschaften redundant spezifiziert. Deshalb wird es oft nötig sein, zusätzliche Eigenschaften, die nicht explizit in der Spezifikation stehen, für die spezifizierten Objekte zu beweisen. Dafür wird ein Verifikationskalkül zusammen mit einem Ableitungssystem benötigt, um diese Eigenschaften ausdrücken und Beweise über sie führen zu können.

Um das Rad nicht neu zu erfinden, haben wir uns an in der Literatur vorgeschlagenen Kalkülen für Objekte orientiert und daraus einen auf unsere Spezifikationssprache angepaßten entwickelt. Die Betrachtung einzelner Objekte stellt in allen uns bekannten Ansätzen kein Problem dar. Hingegen gibt es für die Behandlung mehrere Objekte, die miteinander kommunizieren können, bis heute nur wenige, zumeist rudimentäre Ansätze.

Zur Unterstützung des Beweisens von Objekteigenschaften wird schließlich Werkzeugunterstützung benötigt. Dazu werden wir bestehende Beweissysteme in unsere Entwicklungsumgebung integrieren. Dabei muß allerdings betont werden, daß wir davon ausgehen, daß in einer konkreten Entwicklung eines Informationssystems nur kritische Eigenschaften der Objekte formal bewiesen werden. Dies liegt vor allem darin begründet, daß ein vollständig automatisches Beweisen in der Regel nicht möglich ist und daß interaktives Beweisen am Rechner sehr langwierig werden kann und damit vermutlich auf geringe Akzeptanz stoßen wird.

4 Die TROLL *light*–Entwicklungsumgebung

Für die Unterstützung der Spezifikation von Informationssystemen mit TROLL *light* ist eine integrierte CASE–Umgebung wünschenswert. Einer der wichtigsten Zwecke der TROLL *light*–Entwicklungsumgebung ist die Zertifikation, die Validierung und Verifikation beinhaltet. Ein anderer Zweck, der in Zukunft betrachtet werden soll, ist die Transformation von TROLL *light*–Spezifikationen in ausführbaren Programm–Code.

Im ersten Schritt des Software–Entwicklungsprozesses, der Anforderungsanalyse, wird im wesentlichen informal gearbeitet. Dieser Schritt führt zu einer Entwurfsphase, in der ein realer Weltausschnitt in einer abstrakten TROLL *light*–Spezifikation modelliert wird. Von nun an kann die TROLL *light*–Entwicklungsumgebung für die folgenden Schritte der Entwicklung verwendet werden.

1. **Validierung/Animation:** Die TROLL *light*–Spezifikation wird mit einem Animator prototypisch ausgeführt. Der Animator stellt Objektfenster zur Verfügung, die es den Benutzern ermöglichen, Objekte zu beobachten, neue Objektfenster durch Verfolgen der Unterobjektbeziehungen oder der objektwertigen Attribute zu öffnen, Zustandsübergänge durch Anklicken von Ereignissen zu initiieren, Anfragen an Objektzustände zu stellen, etc. Dies soll den Entwicklern helfen, die informale Sicht auf den realen Weltausschnitt mit der aktuellen Spezifikation zu vergleichen.

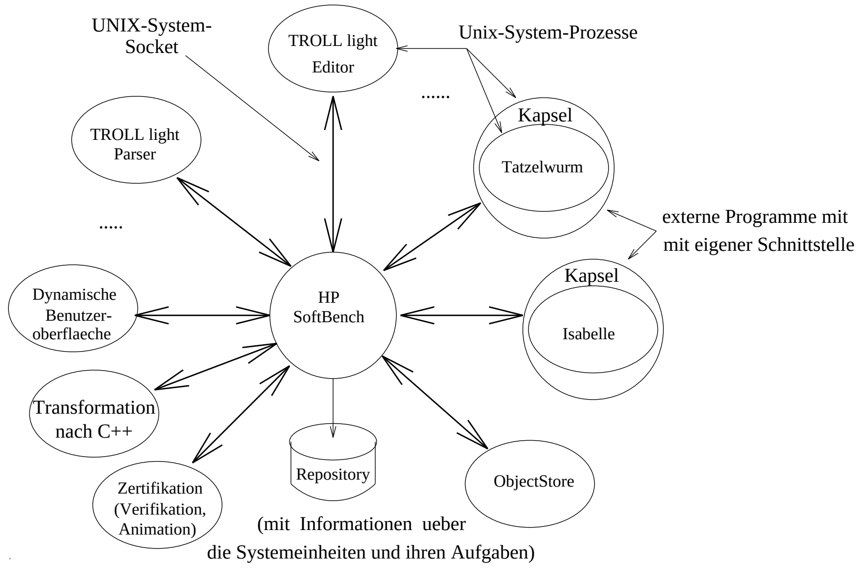

Abbildung 1: Architektur der TROLL *light*–Entwicklungsumgebung

2. **Verifikation:** Die TROLL *light*–Spezifikation wird bzgl. Inkonsistenzen, Mängeln und Zusicherungen geprüft.
3. **Modifikation:** Validierung und Verifikation können Ergebnisse liefern, die es nötig machen, die TROLL *light*–Spezifikation zu verändern.
4. **Transformation:** Die Spezifikation wird in möglicherweise mehreren Schritten in Richtung eines ausführbaren Programm–Codes transformiert.

Technische und konzeptionelle Aspekte. Die Architektur der TROLL *light*–Entwicklungsumgebung, wie sie in Abbildung 1 skizziert ist, stellt eine offene Struktur dar, die in der Lage ist, Programme und Tools (Systemeinheiten) jeder Art im Arbeitsbereich der Benutzer zu integrieren. Alle Systemeinheiten laufen innerhalb ihrer eigenen Systemprozesse fast unabhängig voneinander und können darüber hinaus auf mehrere Maschinen verteilt werden. Diese verteilte Architektur wurde gegenüber einer monolithischen bevorzugt, da sie in der Lage ist, existierende Tools und Programme, die in verschiedenen Programmiersprachen geschrieben sind, zu integrieren.

Der Systemkern der TROLL *light*–Entwicklungsumgebung ist der Kommunikationsmanager *HP SoftBench*, der alle Systemeinheiten, die zur Umgebung gehören, miteinander verbindet. Dadurch wird eine lose aber funktionsintegrierende Kopplung der Systemeinheiten erreicht. Der Kommunikationsmanager verwaltet darüber hinaus Informationen über alle Systemeinheiten und deren Dienste in einem lokalen Repository.

Das dynamische Einbinden von neuen Systemeinheiten in die Umgebung ist ein essentieller Aspekt des Systems. Diese Systemarchitektur läßt sich als eine weitere Schicht über der Betriebssystemebene auffassen, die einen einheitlichen Rahmen für die Integration von Tools schafft.

Die Basisversion der TROLL *light*–Entwicklungsumgebung wird im wesentlichen einen Parser für TROLL *light*, ein Repository für das strukturierte Abspeichern von Spezifikationen, ein Animationssystem, ein Zertifikationssystem und daran angeschlossen Theorembeweiser enthalten.

5 Stand des Projektes

Teile der TROLL *light*–Entwicklungsumgebung sind bereits implementiert. Die Arbeiten am TROLL *light*–Parser sind abgeschlossen, und Spezifikationen können mit Hilfe von ObjectStore in strukturierter Form abgelegt werden. Die erste Komponente eines Zertifikations–Tools, das die Überprüfung der Konsistenz von Spezifikationen gestattet, ist auf der Basis des Theorembeweisers Tatzelwurm implementiert.

Zur Zeit haben ca. 15 Studentinnen und Studenten ihre Studien– oder Diplomarbeiten im Projekt erstellt oder sind bei der Anfertigung ihrer Arbeit. Die gegenwärtigen Implementierungsarbeiten betreffen im wesentlichen den TROLL *light*–Animator, die Benutzerschnittstelle und die Kommunikationskomponente. Bis zum Ende des Projektes im März 1994 bleibt noch viel Arbeit zu tun, aber unsere bisherigen Ergebnisse lassen uns hoffen, daß dann die wichtigsten Teile der TROLL *light*-Entwicklungsumgebung realisiert sind.

Danksagung: Kolleginnen und Kollegen der Abt. Datenbanken an der TU Braunschweig und KorSo–Projektpartner haben durch Fragen und Anregungen zu dieser Arbeit beigetragen. Eine Vorversion des ursprünglichen Textes ist von Karl Neumann kritisch durchgesehen worden.

Anmerkung: Leider kann aus Platzgründen nur diese stark verkürzte Fassung abgedruckt werden. TROLL *light* wird in [CGH92] detailiert beschrieben. Eine ausführlichere Darstellung der Entwicklungsumgebung ist in [VHG+93] zu finden, insbesondere ist dort eine ausführliche Literaturliste zu finden, die alle hier erwähnten Ansätze und Arbeiten berücksichtigt.

Literatur

[CGH92] S. Conrad, M. Gogolla und R. Herzig. TROLL light: A Core Language for Specifying Objects. Informatik-Bericht 92–02, Technische Universität Braunschweig, 1992.

[EGS92] H.-D. Ehrich, M. Gogolla und A. Sernadas. Objects and their Specification. In M. Bidoit und C. Choppy, Herausgeber, *Proc. 8th Workshop on Abstract Data Types*, S. 40–66. LNCS 655, Springer, Berlin, 1992.

[JSHS91] R. Jungclaus, G. Saake, T. Hartmann und C. Sernadas. Object-Oriented Specification of Information Systems: The TROLL Language. Informatik-Bericht 91–04, Technische Universität Braunschweig, 1991.

[SE91] A. Sernadas und H.-D. Ehrich. What Is an Object, After All? In R. Meersman, W. Kent und S. Khosla, Herausgeber, *Object-Oriented Databases: Analysis, Design and Construction (Proc. 4th IFIP WG 2.6 Working Conference DS-4, Windermere (UK))*, S. 39–70, Amsterdam, 1991. North-Holland.

[VHG+93] N. Vlachantonis, R. Herzig, M. Gogolla, G. Denker, S. Conrad und H.-D. Ehrich. Towards Reliable Information Systems: The KORSO Approach. In C. Rolland, Herausgeber, *Proc. 5th Int. Conf. Advanced Information Systems Engineering*. Springer, LNCS Series, 1993. *To appear.*