Boris Böhlen: **Ein parametrisierbares Graph-Datenbanksystem für Entwicklungswerkzeuge**

Promotion: RWTH Aachen
Erstgutachter: Prof. Dr.-Ing. Manfred Nagl, RWTH Aachen
Zweitgutachter: Prof. Dr. Andy Schürr, TU Darmstadt
Datum der Prüfung: 26. Mai 2006
Veröffentlichung: 223 S., Shaker-Verlag, 2006

Kurzfassung:

Klassische Datenbanksysteme sind für die Verwaltung komplexer Dokumente als Ergebnis von Entwicklungstätigkeiten nur bedingt geeignet. Dies gilt insbesondere, wenn die Dokumente untereinander auf einer sehr detaillierten Ebene miteinander vernetzt sind. Im IPSEN-Projekt und anderen Projekten des Lehrstuhls für Informatik 3 werden daher Graphen für die Modellierung der Dokumente verwendet. Die Speicherung der Dokumente erfolgt in der Graphdatenbank GRAS. Aktuelle Projekte definieren Anforderungen, die von GRAS nicht unterstützt werden. Die Konsequenz ist die Entwicklung des im Folgenden vorgestellten Datenbanksystems DRAGOS.

Die Probleme von GRAS liegen im Bereich Erweiterbarkeit und Anpassbarkeit. Auf das einfache Graphmodell von GRAS können komplexe Graphmodelle — wie Hypergraphen — nur mit hohem Aufwand abgebildet werden. Graphersetzungssysteme mit anderen Graphmodellen können GRAS nicht einsetzen. Auch die Funktionalität kann nur schwer an neue Anforderungen angepasst werden, insbesondere wenn zentrale Komponenten betroffen sind. Die mangelnde Anpassbarkeit erschwert zum Beispiel die Realisierung verteilter Werkzeuge. GRAS wird daher nur von Werkzeugen verwendet, die mit dem Graphersetzungssystem PROGRES spezifiziert werden.

Basierend auf den Schwächen von GRAS werden zunächst die Anforderungen an DRAGOS definiert. DRAGOS definiert ein allgemeines Graphmodell, auf das andere Graphmodelle mit geringem Aufwand abgebildet werden können. Durch seine serviceorientierte Architektur lässt sich DRAGOS an verschiedene Anwendungsszenarien anpassen. Der DRAGOS-Kernel stellt nur die Funktionalitäten als Dienste zur Verfügung, die von einem graphorientierten Datenbanksystem bereitgestellt werden müssen: Ereignisse, Persistenz, Regeln und Transaktionen. Jede zusätzliche Funktionalität — zum Beispiel die Versionierung von Graphen — wird von Erweiterungsmodulen zur Verfügung gestellt. Für die Einbindung der Erweiterungsmodule ist der Kernel verantwortlich. Die Anbindung der verschiedenen Werkzeuge und Graphersetzungssysteme — PROGRES, Fujaba, etc. — erfolgt durch spezialisierte Graphmodelle.

Das Graphmodell von DRAGOS wurde mit der Maßgabe entwickelt, eine Vielzahl unterschiedlicher Graphmodelle zu unterstützen. Im Gegensatz zu seinen Vorgängern werden daher nicht nur Knoten und Kanten bereitgestellt, sondern auch graphübergreifende und n-äre Relationen sowie hierarchische Graphen. Diese Graphelemente können mit Attributen versehen werden. Zusätzlich muss jedes Graphelement typisiert sein. Die zugehörigen Graphelementklassen werden im DRAGOS-Graphschema definiert, das auch Mehrfachvererbung und abstrakte Klassen unterstützt. Für die persistente Speicherung der Graphen werden austauschbare Graphenspeicher verwendet, die einen Graphen unter anderem in Datenbanksystemen wie PostgreSQL speichern.

Im Rahmen der vorliegenden Arbeit sind neben DRAGOS zwei verschiedene Erweiterungsmodule für DRAGOS realisiert worden. Die erste Erweiterung stellt inkrementell berechnete Attribute zur Verfügung. Von der zweiten Erweiterung wird die Versionierung von Graphen angeboten. Änderungen an einem Graphen lassen sich hiermit verfolgen. Zustände mehrerer unterschiedlicher Graphen können zu einer Konfiguration zusammengefasst werden. Die Erweiterungen können beliebig miteinander kombiniert werden.

Oberhalb des DRAGOS-Graphmodells werden spezialisierte Graphmodelle realisiert. Dies wird durch SUMAGRAM unterstützt, das die Spezifikation von Graphmodellen und deren Abbildung auf das DRAGOS-Graphmodell durch UML-Klassendiagramme erlaubt. An den Beispielen GXL und DIAPLAN wird die Funktionalität von SUMAGRAM demonstriert. Die spezialisierten Graphmodelle für die Graphersetzungssysteme

PROGRES und Fujaba wurden ebenfalls im Rahmen dieser Arbeit realisiert. Bei PROGRES wurden die Codegenerierung und das UPGRADE-Rahmenwerk angepasst. DRAGOS ersetzt nun das bisher verwendete GRAS vollständig. Seine Verwendung ist transparent und hat keinen Einfluss auf die Spezifikation oder die Werkzeugentwicklung. Die durch GRAS bedingten Einschränkungen sind weggefallen. Für Fujaba wurde ein spezielles Plugin entwickelt, das dessen Codegenerierung anpasst. Bis auf einige wenige Einschränkungen hat die Verwendung von DRAGOS keinen Einfluss auf die Spezifikation. Basierend auf den DRAGOS-Erweiterungen kann zukünftig die Funktionalität von Fujaba um komplexe Pfade, Nichtdeterminismus, etc. erweitert werden.

Die vorgestellten Konzepte wurden implementiert und validiert. DRAGOS stellt einen vollständigen Ersatz für seine Vorgänger dar. Durch seine offene Architektur und Erweiterbarkeit ist DRAGOS für zukünftige Anwendungen gewappnet. Die Probleme seiner Vorgänger werden somit langfristig vermieden.

Felix H. Gatzemeier: **CHASID:A semantics-oriented authoring environment**

Promotion: RWTH Aachen
Erstgutachter: Prof. Dr.-Ing. Manfred Nagl, RWTH Aachen
Zweitgutachter: Prof. Dr. Jörg Haake, FernUniversität Hagen
Datum der Prüfung: 10. Mai 2004
Veröffentlichung: 270 S., Shaker-Verlag, 2005

Kurzfassung:

In writing an informational document (such as a scientific article or a textbook), an author faces a number of complicated problems: not just creating the actual media (text, images, and so forth) with its intricacies of formulation, but also selecting the content to be presented and structuring it so that the document in its entirety is consistent and readable. The latter is complicated by re-iterations over the document, where the author reviews and edits it from different perspectives.

CHASID, the project described in this book, addresses these tasks of the author: (a) Planning a document, (b) Upholding plans, and (c) Avoiding common structural problems. To do so, it blends itself into the existing conventional authoring environment and maintains a semantical model of the document, which is connected to its hierarchical structure of chapters, sections etc. The functionality is organized according to a cognitive model of the authoring process.

The semantical model contains the topics of the document, together with their relations. It is thus also called the topic map here. The connection to the document's hierarchical structure is kept in exports and imports, indicating which divisions explain a topic, and which ones expect it to be known.

The additional functionality requires additional information which is not usually captured during authoring, namely the topic map and its connection. The author generally cannot be assumed to be willing to invest the effort of providing this information without seeing some direct benefit. So, the relationship between additional effort and kinds of possible functionality has also been considered.

Beginning with the least additional effort, re-