Ein kooperativer XML-Editor für Workgroups

Francis Gropengießer, Katja Hose, Kai-Uwe Sattler
{francis.gropengiesser|katja.hose|kus}@tu-ilmenau.de

Abstract: In vielen Anwendungsgebieten, z.B. im Design oder in Medienproduktionsprozessen, hat sich XML als Format für den Datenaustausch etabliert. Zur Verarbeitung von XML-Daten in Mehrbenutzerumgebungen sind spezielle Werkzeuge nötig. Allerdings berücksichtigen die derzeit auf dem Markt verfügbaren Editoren nur unzureichend die Anforderungen kooperativer Arbeitsumgebungen. In dieser Arbeit wird daher ein transaktionsbasierter XML-Editor vorgestellt, der diese Anforderungen erfüllt. Er ist für den Einsatz in eng gekoppelten Systemumgebungen, auch Workgroups genannt, konzipiert. Der Editor erlaubt ein intuitives Bearbeiten von XML-Daten und macht Änderungen eines Nutzers an den Daten anderen Nutzern sehr früh zugänglich. Weiterhin ermöglicht er durch dein Einsatz eines intelligenten Sperrprotokolls einen hochgradig parallelen Arbeitsprozess.

1 Einleitung

In vielen Anwendungsbereichen hat sich XML als Format für den Datenaustausch etabliert. Ein Beispiel ist der Postproduktionsprozess von Filmton mit dem räumlichen Soundsystem IOSONO (*http://www.iosono-sound.com/*). Die von den Autoren erstellten Soundszenen werden als XML-Dateien gespeichert. Das Ziel unserer Arbeit ist es, ein kooperatives Arbeiten von mehreren Autoren an den selben Soundszenen bzw. den selben XML-Dateien zu ermöglichen. Kooperativität bedeutet dabei, dass alle Autoren gleichberechtigt sind und jeder von ihnen Kenntnis über den aktuellen Stand des Projekts besitzt. Jeder Autor ist berechtigt, Lösungsansätze zum Projekt beizutragen, die wiederum von anderen Autoren auf Korrektheit überprüft werden. Es findet daher ein Informationsaustausch in beliebiger Richtung zwischen den Autoren statt.

Die aus der Forderung nach Kooperativität resultierenden Anforderungen, z.B. multidirektionaler Informationsfluss und frühe Sichtbarkeit von Änderungen am Datenbestand, sind nur schwer zu erfüllen. Beispielsweise bedeutet die Forderung nach einem beliebig gerichteten Informationsaustausch einen Verzicht auf Serialisierbarkeit. Diese dient jedoch den traditionellen Transaktionsmodellen als Korrektheitskriterium. Um alle Autoren immer auf dem aktuellen Stand des Projekts zu halten, sind Versionierungslösungen wie SVN oder CVS unzureichend, da in diesen Systemen Änderungen nur durch den expliziten Aufruf von Check-in und Update propagiert werden. In [GS08] ist eine detaillierte Beschreibung aller Anforderungen kooperativer Umgebungen sowie eine Einsatztauglichkeitsanalyse bekannter Transaktions- und Workflowmodelle zu finden.

In dieser Arbeit wird ein neuartiger XML-Editor präsentiert, der den Anforderungen kooperativer Medienproduktionsprozesse in eng gekoppelten Systemumgebungen (Workgroups) gerecht wird. Er stellt ein Konglomerat aus (i) klassischen Datenbanktechniken, wie z.B. Transaktionen, Sperrprotokollen und Transaktionsrecoverymodellen, (ii) einem grafischen XML-Editor, wie z.B. XMLSpy (*http://www.altova.com/de/*), und (iii) einem Kollaborationsmechanismus, ähnlich dem in SubEthaEdit (*http://www.codingmonkeys.de/subethaedit/index.de.html*), dar.

2 Systemarchitektur

Für die Modellierung einer Workgroup wurde eine *Client-Server-Architektur* gewählt, in der mehrere Nutzer (*Clients*) permanent mit einem zentralen Repository für die XML-Daten (*Server*) verbunden sind (Abbildung 1).

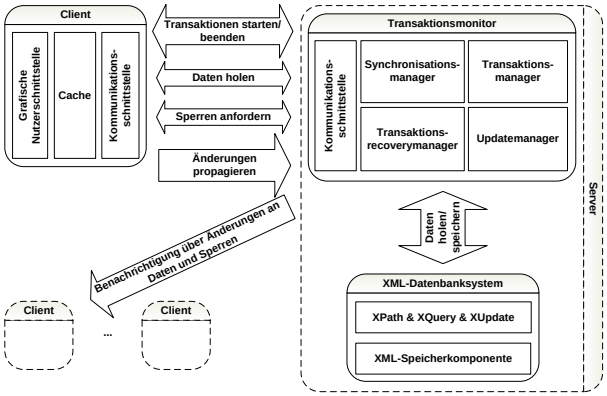

Abbildung 1: Systemarchitektur

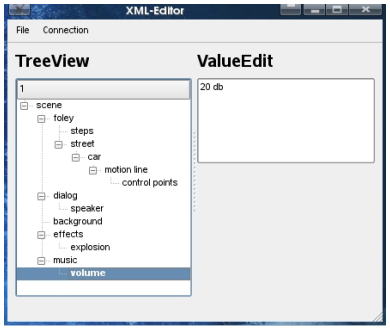

Abbildung 2: Graphische Nutzerschnittstelle

Der *Server* besteht aus 2 Hauptkomponenten – dem *Transaktionsmonitor* und dem *XML-Datenbanksystem*. Der Transaktionsmonitor enthält vier Hauptkomponenten. Der *Transaktionsmanager* implementiert unser kooperatives Transaktionsmodel [GS08], basierend auf geschachtelten dynamischen Aktionen [NW94]. Weiterhin verwaltet er die Transaktionen aller verbundenen Clients. Der *Synchronisationsmanager* realisiert unser kooperatives Sperrprotokoll [GS08], das die Semantik der Operationen auf der Baumrepräsentation von XML-Dokumenten ausnutzt. Der *Transaktionsrecoverymanager* implementiert eine Transaktionsrecoverystrategie, basierend auf einer Logging-Technik. Er wird benötigt um Transaktionsfehler und Transaktionsabbrüche zu behandeln. Der *Updatemanager* realisiert das Publisher-Subscriber-Pattern. Jeder Client registriert sich beim Updatemanager, um über Änderungen jeglicher Art, z.B. an den Daten oder Sperren, benachrichtigt zu werden. Die Registrierung erfolgt automatisch, sobald der Nutzer einen Teil eines XML-Dokuments herunterlädt, den er bearbeiten möchte. Treten Änderungen bezüglich dieses Dokumentteils auf, so wird der Nutzer darüber sofort in Kenntnis gesetzt.

Um XML-Daten dauerhaft zu speichern, macht der Transaktionsmonitor Gebrauch von einem *XML-Datenbanksystem*. Um Daten vom Server zu erhalten / anzufragen wird *XPath* und *XQuery* genutzt. Änderungen werden mit *XUpdate* geschrieben. Da der Transaktionsmonitor lediglich die Behandlung von Transaktionsfehlern und Transaktionsabbrüchen übernimmt, bietet die *XML-Speicherkomponente* Möglichkeiten für ein Systemrecovery. Weiterhin übernimmt das XML-Datenbanksystem eine Transformation der XML-Daten in eine spezielle Baumstruktur [HH03, GS08], die die Grundlage für alle Verarbeitungsschritte bildet.

Die Clients besitzen einen lokalen *Cache*, in dem sie Kopien von den Teilen der XML-Dokumente speichern, die sie bearbeiten wollen. Alle Operationen werden zunächst auf diesen lokalen Kopien ausgeführt. Dies hat den Vorteil, dass im Falle eines Transakti-

onsabbruchs noch keine Daten zum Server gesendet wurden. Um ein komfortables Arbeiten zu ermöglichen, bietet der Client eine *grafische Nutzeroberfläche* (Abbildung 2). Diese stellt die zu bearbeitenden XML-Daten als Baumstruktur dar. Attributwerte oder Text können sehr einfach editiert werden, indem der betroffene Knoten selektiert wird und die Änderungen vorgenommen werden. Einzelne Knoten oder Teilbäume können ganz leicht per Drag'n'Drop verschoben werden. Weiterhin bietet diese Nutzerschnittstelle alle nötigen Funktionen an, um eine Verbindung zum Server herzustellen und Kopien der zu bearbeitenden Daten zu beziehen.

3 Eigenschaften

Ziel des in dieser Arbeit vorgestellten XML-Editors ist es, wie bereits in der Einleitung erwähnt, die kooperative Verarbeitung von XML-Daten in Mehrbenutzer-Medienproduktionsprozessen zu unterstützen. Um diesen Zweck zu erfüllen, bietet er folgende Eigenschaften:

- Änderungen eines Nutzer werden sofort anderen Nutzern angezeigt. Dadurch besitzen alle Nutzer immer Kenntnisse über den aktuellen Stand des Projekts.
- Ein Nutzer kann sehen, welche Teile eines Dokuments von anderen Nutzern zeitgleich bearbeitet werden. Dies wir ihm durch farbliche Hervorhebungen in der grafischen Repräsentation der Daten angezeigt.
- Nutzer können einzelne Arbeitschritte verwerfen ohne ihren gesamten Arbeitsfortschritt zu verlieren.
- Das verwendete Sperrprotokoll fordert keine strikte Serialisierbarkeit, wodurch ein Informationsaustausch in beliebiger Richtung möglich ist. Es nutzt die Semantik der Operationen auf der speziellen Baumstruktur der XML-Daten aus und erzielt dadurch hochgradig parallele Arbeitsabläufe.

4 Demonstration

Bei der Demonstration wird ein Workgroup-Szenario im kleinen Rahmen realisiert. Mehrere Benutzer (z.B. die Teilnehmer der Konferenz) können gleichzeitig eine Soundszene bearbeiten. Dabei wird der Umgang mit der grafischen Oberfläche und dem System im Allgemeinen verdeutlicht. Dies umfasst grundlegende Schritte wie Verbinden mit dem Server, Herunterladen der zu bearbeitenden Daten und Ausführung von Operationen (Updates) auf den Daten. Ein besonderer Schwerpunkt liegt darauf, die Kooperativität des Systems zu demonstrieren. Es wird gezeigt, wann Änderungen propagiert werden, wie bei Transaktionsabbrüchen verfahren wird und wie Aktivitäten anderer Nutzer grafisch visualisiert werden. Weiterhin soll gezeigt werden, warum es vorteilhaft ist, Transaktionen, konkret unser spezielles Transaktionsmodell, zu verwenden. Ein letzter Schwerpunkt liegt auf der Demonstration unseres Sperrprotokolls. Unter anderem wird gezeigt, wie die Semantik der Operationen auf der speziellen Baumstruktur ausgenutzt wird.

Literatur

[GS08] Francis Gropengießer und Kai-Uwe Sattler. An Extended Cooperative Transaction Model for XML. In *PIKM'08 icw CIKM'08*, Seiten 41–48, 2008.

[HH03] Michael Peter Haustein und Theo Härder. taDOM: A Tailored Synchronization Concept with Tunable Lock Granularity for the DOM API. In *ADBIS*, Seiten 88–102, 2003.

[NW94] Edgar Nett und Beatrice Weiler. Nested Dynamic Actions - How to Solve the Fault Containment Problem in a Cooperative Action Model. In *Symposium on Reliable Distributed Systems*, Seiten 106–115, 1994.