Verteiltes WebCoObRA Framework
Ein wolkiger Ausblick

Marcel Hahn, Ruben Jubeh, Albert Zündorf

University of Kassel, Software Engineering Research Group,
Department of Computer Science and Electrical Engineering,
Wilhelmshöher Allee 73,
34121 Kassel, Germany
[hahn | ruben | zuendorf]@cs.uni-kassel.de
http://www.se.eecs.uni-kassel.de/

Zusammenfassung Wir verwenden seit geraumer Zeit erfolgreich das CoObRA Persistenz Framework [3] innerhalb des Fujaba-Projektes [1] als Persistenzschicht für modellgetrieben entwickelte Anwendungen. Im Kontext von größeren Web-Anwendungen zeigen sich jedoch Probleme - das Instanz-Modell wird als komplette Einheit betrachtet und sowohl auf dem Server als auch allen Clients vollständig im Speicher gehalten. Es gibt nur eine zentrale Instanz, welche die Daten persistiert. Fein granulare Rechte und Sichtbarkeiten, z.B. für verschiedene Nutzerrollen, sind nicht möglich. Wir möchten einen Ausblick auf geplante Aktivitäten in unserer Forschungsgruppe geben, das CoObRA-Framework fit für verteilte Web-Anwendungen zu machen. Dabei ist uns besonders wichtig, dass bestehende Anwendungen einfach migriert werden können und der strikt modellgetriebe Ansatz fortgeführt wird.

1 Einleitung

Das existierende Framework arbeitet nach folgender Funktionsweise: persistente Objekte haben eindeutige Bezeichner (UUIDs). Diese Objekte feuern alle Änderungen an Attributen und Links als Events. Das CoObRA-Framework zeichnet diese im sogenannten Änderungs-Protokoll auf. Dieser wird textuell, pro Änderung eine Zeile, auf die Platte geschrieben. Vorteil des Protokoll ist, dass die komplette Historie der Objektstruktur vorhanden ist. Es kann jederzeit ein Undo/Rollback gemacht werden. Zusätzlich können Gruppen von Änderungen in Transaktionen gekapselt werden. Beim Laden von Anwendungen wird das Änderungsprotokoll vorwärts durchlaufen. Objekte werden dabei instanziert und schrittweise mit Attributwerten befüllt und verlinkt. Das CoObRA-Framework unterstützt weiterhin eine Client-Server-Struktur: mehrere Clients können über ein Versionierungsprotokoll (Checkout/Update, Checkin/Commit) mit einer zentralen Serverinstanz Änderungen an derselben Objektstruktur, welche dann auf allen Clients und dem Server instanziert ist, abgleichen. Das Framework bietet mehrere Konflikterkennungs- und Lösungsstrategien. CoObRA stellt somit ein Versionierungssystem für Modelle dar. Je nach dem mit welchem Metamodell es arbeitet, kann es einerseits auf der Anwendungsebene (z.B. für Webanwendungen) als Laufzeitpersistenz benutzt werden, als auch auf der Tool-Ebene als Persistenzschicht für unser Fujaba-Werkzeug (UML-Metamodell) [4].

Die existierende WebCoObRA-Erweiterung [1] erlaubt nun dem Client-Anteil von Webanwendungen (technisch: ECMAScript, das im Browser läuft), Änderungen mit dem Server auszutauschen. Dazu wird das Comet-Muster zur schnellen bidirektionalen Kommunikation verwendet. Weiterhin wurde der UUID Algorithmus angepasst, so dass hierarchische IDs entstehen, da sonst bei Objekterzeugung auf den Clients Kollisionen auftreten können.

[1] http://www.fujaba.de/

Das CASE-Tool Fujaba [2] mit dem Ansatz zur Modellierung ausführbarer Anwendungen mit Story Driven Modelling [5] erlaubt es, persistente Anwendungen aus einem UML- und Story-Diagrammen zu generieren. Wir möchten diesen Ansatz fortführen, so dass alle hier präsentierten Erweiterungen sollen ebenfalls im Modell via UML Profilen werden können.

2 Anforderungen

In [1] haben wir die grundsätzliche Methodik präsentiert, CoObRA-persistente Anwendungen als Web-Anwendungen zu generieren. Als grafische Oberfläche wird ein auf Google Web Toolkit [2] basierende Widget-Bibliothek gesetzt. Das dort präsentierte Beispiel, das Spiel 'Mensch-Ärgere-Dich-Nicht', ist allerdings zu klein und erfüllt viele Anforderungen an reale Webanwendungen nicht. Wir evaluieren zur Zeit folgende webbasierte Anwendungen: Ein virtuelles verteiltes Verkehrsnetzplanungssytem, das mit den Rohdaten von z.B. OpenStreetMap als Objektmodell gefüllt wird, sowie ein kollaborative Kanban-Board-Anwendung für verteilte agile Softwareentwicklung. WebCoObRA-Clients sollen sprach- und platformunabhängig sein. Weiterhin haben wir folgende Anforderungen bisher identifiziert:

- Es fallen größere Datenmengen an, die effizient verwaltet werden müssen. Insbesondere die Clients sind nicht der Lage, komplette Objektstrukturen im Speicher zu halten. Auch auf der Serverseite soll die Objektstruktur zwecks Ausfallsicherheit und Lastverteilung partitioniert und auf mehrere Server verteilt werden. Inaktive Teile des Objekt-Graphen können per Lazy Loading bei Bedarf nachgeladen werden. Damit dies effizient realisiert werden kann, muss es möglich sein, Pfad-Anfragen zu definieren.
- Der Replay des kompletten Änderungsprotokolls beim Anwendungsstart ist ineffizient. Deshalb sollen Anwendungszustände serialisiert gespeichert werden und das vorangegangene Change-Protokoll als separate Historie.
- Ein Client kann per Publish/Subscribe-Mechanismus nur für bestimmte Objekt-Änderungen registrieren - dies reduziert die übertragene Datenmenge erheblich.
- Im aktuellen Ansatz fehlen noch die Berücksichtigung von Sicherheitsanforderungen. Wir planen ein hierarchisches, rollenbasiertes Rechtemanagement auf Objektebene.

Literatur

1. N. Aschenbrenner, J. Dreyer, M. Hahn, R. Jubeh, C. Schneider, and A. Zündorf. Building Distributed Web Applications based on Model Versioning with CoObRA: an Experience Report. In *Proc. 2009 Intl. Workshop on Comparison and Versioning of Software Models*, pages 19–24. ACM, May 2009.
2. T. Fischer, J. Niere, L. Torunski, and A. Zündorf. Story diagrams: A new graph rewrite language based on the unified modeling language. In *Proc. of the* 6^{th} *International Workshop on Theory and Application of Graph Transformation*. Paderborn, Germany, 1998.
3. C. Schneider. *CoObRA: Eine Plattform zur Verteilung und Replikation komplexer Objektstrukturen mit optimistischen Sperrkonzepten*. PhD thesis, 2007.
4. C. Schneider, A. Zündorf, and J. Niere. CoObRA - a small step for development tools to collaborative environments. In *Workshop on Directions in Software Engineering Environments in 26th international conference on software engineering*, Edinburgh, Scotland, UK, May 2004.
5. A. Zündorf. Rigorous object oriented software development. Habilitation Thesis, University of Paderborn, 2001.

[2] http://code.google.com/intl/de/webtoolkit/