Dawn – Eine kollaborative Erweiterung für GMF-Editoren auf Basis modellgetriebener und webbasierter Technologien

Martin Flügge

1 Abstrakt

GMF, das Graphical Modeling Framework, ermöglicht es mit Hilfe von modellgetriebenen Technologien grafische Editoren zu erzeugen. Diese Editoren können auf vielfältigste Weise genutzt werden. Es fehlt ihnen aber die Möglichkeit den Editor kollaborativ zu verwenden. Um diese Lücke zu schließen, wurde Dawn, eine kollaborative, webbasierte Erweiterung für GMF-Editoren, entwickelt. Dawn bietet verteilten Zugriff auf GMF-Diagramme und stellt auch Verfahren für die Erkennung und Vermeidung von Konflikten bereit. Neben der Kommunikationsinfrastruktur bietet das entwickelte System ein Konzept zur Darstellung von GMF-Diagrammen in einem Browser an. Diese Funktionalität kann dazu genutzt werden, auch auf mobilen Endgeräten GMF-Diagramme anzeigen zu können. Da mobile Endgeräte nicht immer in stabilen Netzwerken agieren, ist Dawn auch ohne Netzwerkverbindung lauffähig. Um das System vor unbefugtem Zugriff zu schützen, wurde außerdem ein generisches Nutzerkonzept entwickelt, welches auch ohne Netzwerkverbindung weiter genutzt werden kann.

2 Einleitung

Das Graphical Modeling Framework (GMF, [1]) stellt als Framework Konzepte zur Erstellung grafischer Editoren zur Verfügung. Im Umfeld der Modellgetriebenen Softwareentwicklung (MDSD, Model Driven Software Development [2]) können diese Editoren als grafische domänenspezifische Sprachen (DSL, Domain Specific Language [2]) genutzt werden. Des Weiteren können GMF Editoren auch in anderen Bereichen beispielsweise zu dokumentarischen Zwecken in Geschäftsprozessmodellen bis hin zur Automatisierung innerhalb von Workflow-Managementsystemen genutzt werden. Durch den Zugriff auf den generierten Code können beliebige Änderungen implementiert werden. GMF stützt sich dabei auf das Eclipse Modeling Framework (EMF, [3]) und das Graphical Editing Framework (GEF, [4]), nutzt also eine ausgereifte, stabile Basis. Im Zuge der starken Globalisierung ist es allerdings erforderlich Arbeiten an derartigen Editoren auch kollaborativ in Echtzeit ausführen zu können. Derzeit existiert kein Framework, welches es ermöglicht GMF-Editoren derart einzusetzen. Szenearien in denen mehrere Entwickler an verschiedenen Standorten verteilt an einem Diagramm arbeiten sind nicht umsetzbar. Geeignete Technologien zu untersuchen und Verfahrensweisen zur Verfügung zu stellen, um diese Lücke zu schließen war Ziel einer Diplomarbeit, welche im Wintersemester 2008/2009 an der Hochschule für Technik und Wirtschaft Berlin (HTW Berlin) erstellt wurde. Das Ergebnis dieser Arbeit wird in den folgenden Kapiteln beschrieben und wurde in Anlehnung an sonnenbasierte Ereignisse, welche die Basis der Namensgebung für das Eclipse Projekt waren (Eclipse = Sonnenfinsternis), *Dawn* (Sonnenaufgang) genannt [5].

3 Konzepte

Der Grundgedanke von Dawn basiert auf einer freiheitlichen Kollaboration der Nutzer in der Bearbeitung von GMF Diagrammen. Hierbei war es Ziel, das System derart flexibel zu gestalten, dass es unter beinahe beliebigen Anwendungsfeldern nutzbar ist. Hierzu zählen sowohl eine flexible Kommunikation, als auch die netzwerkunabhängige Nutzung des Systems. Des Weiteren wurden

Komponenten entwickelt, die auch die mobile Nutzung des Systems ermöglichen. Neben diesen Anforderungen muss auch die Integrität der Daten bei Mehrfachzugriff sichergestellt werden.

3.1 Kommunikation

Dawn wurde als Client-Server-Architektur konzipiert um eine zentrale Komponente sowohl für die Ablage der Daten als auch die Verwaltung von Zugriffen auf das System zu ermöglichen.

Zusätzlich verfügt das System über eine flexible Kommunikationsarchitektur, welche es ermöglicht das zugrunde liegende Kommunikationsprotokoll austauschbar zu machen. Hierzu wurde das Konzept der Kommunikationsadapter entwickelt. Diese, hinter einer Schnittstelle versteckten Implementierungen, bilden die programmatische Grundlage für die Kommunikation. In der aktuellen Version unterstützt Dawn SOAP und RMI als Übertragungsmethodiken. Zusätzliche Protokolle und Technologien können einfach in das System integriert werden und ermöglichen so anderen Nutzern die Kommunikation in Dawn an ihre Bedürfnisse anzupassen. Darüber hinaus kann der Anwender des Systems das Kommunikations-protokoll zur Laufzeit austauschen.

Im Rahmen der Kommunikations-implementierung war es zusätzlich von Interesse Dawn so zu gestalten, dass eventuelle Beschränkungen durch Firewalls nicht zu einer Unbrauchbarkeit des Systems führen. So kann Dawn auch in stark reglementierten Netzwerken genutzt werden. Durch den SOAP-Kommunikationsadapter besteht jeder Zeit ein Fallback-Protokoll, welches über das HTTP-Protokoll, also Port 80 agiert. Dieser ist in der Regel immer freigeschaltet, so dass die Kollaboration weiterhin gewährleistet bleibt.

3.2 Konflikte

In allen Systemen, in denen mehrere Komponenten zeitgleichen Schreibzugriff auf eine Ressource haben, besteht Potential für die Entstehung von Konflikten. Auch in Dawn können Konflikte durch den mehrfachen Zugriff auf die Diagrammdaten entstehen. Um inkonsistente Zustände zu vermeiden wurden Konflikterkennungs- und Konfliktbeseitigungs-mechanismen implementiert.

Im System wurden auf Grund der Natur des Systems drei mögliche Arten von Konflikten identifiziert. *Änderungskonflikte* entstehen dabei durch die gleichzeitige Manipulation des Modells. Ändert ein Nutzer einen Teil seines Modells und wurde dieses aber bereits von einem anderen Nutzer gelöscht, entsteht ein *lokaler Löschkonflikt*. Dementgegen steht der *globale Löschkonflikt*, bei dem der Konflikt durch die lokale Löschung eines global geänderten Objektes entsteht. Für die Detektierung von Konflikten wurde ein Three-Way-Merge Verfahren genutzt, welches Änderungen ohne Konflikte direkt zusammenführen kann.

Wurden Konflikte erkannt, so wird das Publizieren der nicht synchronisierten Daten verhindert und das Problem geeignet visualisiert. Hierbei wurden für die Anzeige von Konflikten die betroffenen Komponenten innerhalb der Diagramme soweit möglich farblich markiert (vgl. Abbildung 1, rot und blau markierte Objekte). Diese Vorgehensweise ermöglicht ein schnelles visuelles Erfassen der Probleme. Da sich aber nicht alle Konflikte auf diese Weise anzeigen lassen, verfügt Dawn zusätzlich über eine View, welche tabellarisch alle Konfliktzustände anzeigen kann.

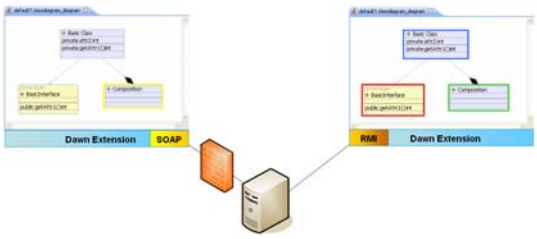

Abbildung 1 - Konflikte und Locking in Dawn

Die Beseitigung von Konflikten in Dawn verfolgt einen liberalen Ansatz, bei dem der Nutzer entscheiden kann, ob er Änderungen eines anderen Nutzers akzeptieren oder stattdessen seine eigenen Änderungen publizieren möchte.

3.3 Locking

Um im Voraus Konfliktsituationen gänzlich zu vermeiden, können in Dawn Teile des Diagramms exklusiv für einen Nutzer gesperrt werden. Auch hierbei wird der Nutzer durch eine farbliche Visualisierung unterstützt (vgl. Abbildung 1, grün und gelb markierte Objekte). Alle Locks werden sofort an alle beteiligten Nutzer weitergeleitet. Das System sperrt daraufhin die Objekte, sodass keine Änderungen an diesen mehr vorgenommen werden können, bis die Sperrung aufgehoben wird. Zur Markierung, welche Objekte von anderen Nutzern gesperrt sind, werden diese mit einer gelben Umrandung versehen.

Für die Umsetzung der Locking-Mechanismen ist es erforderlich zu wissen, welcher Nutzer Objekte reserviert hat. Hierzu bedarf es der Authentifizierung von Nutzern. Dawn bietet darüber hinaus auch die Möglichkeit zur Autorisierung für bestimmte Aktionen. So können beispielsweise Schreibrechte auf die Diagramme reglementiert oder die Rechte der Verwaltung von Diagrammen beschränkt werden. Hierzu stellt Dawn eine webbasierte Verwaltungsapplikation zur Verfügung, über welche die Projekteinstellungen vorgenommen werden können. Änderungen in dieser Applikation werden sofort übernommen, sodass beispielsweise das Entfernen der Schreibrechte eines Nutzers unmittelbar auf den Editor übertragen wird, sodass augenblicklich keine Schreiboperationen mehr ausgeführt werden können. Dieses Rechtemanagement kann durch eine generische Implementierung einfach erweitert werden, sodass auch die Rechte einfach an die Bedürfnisse von Endnutzern angepasst werden können.

3.4 Verbindungslose Verbindung

Um einen möglichst hohen Grad an Flexibilität zu erreichen und somit die Akzeptanz des Systems zu erhöhen, können Dawn-Diagramme auch ohne Verbindung zum Server bearbeitet werden. In diesem Offline-Modus werden Änderungen auch über einen längeren Zeitraum protokolliert und bei wiederhergestellter Netzwerkverbindung mit dem Server synchronisiert. Somit muss die Arbeit an den Diagrammen nicht unterbrochen werden, wenn das Netzwerk einmal nicht zur Verfügung steht, sei es in Hotels, Bars oder Flugzeugen.

3.5 Generatoren

Für die Bereitstellung der Erweiterung ist es wichtig, dass die Implementierung so geschaffen wurde, dass kein bestehender Quellcode der generierten GMF-Editoren verändert wird. Dies ist zum einen nötig, da GMF-Editoren oft nach der Generierung erweitert werden. Des Weiteren wäre eine direkte Code-Manipulation nicht möglich wenn nur der kompilierte Code vorliegt. Um dies zu erreichen, wurden die bestehenden GMF-Plug-ins durch Fragmente erweitert, welche die nötigen Erweiterungen zur Verfügung stellen. Somit können zum einen Editoren erweitert werden, bei denen kein Zugriff auf den Quellcode besteht, beispielsweise wenn kein Quellcode, sondern nur das Modell und der kompilierte Bytecode vorhanden sind. Zum anderen ermöglicht dieses Vorgehen die kollaborative Erweiterung jeder Zeit entfernen zu können, ohne Eingriffe in das bestehende System vornehmen zu müssen.

Zur Erstellung dieser Fragmente wurde ein Generator entwickelt, welcher mit Hilfe der Transformationssprache XPand [6], aus einem dem GMF-Editor zugrunde liegenden Modell, die Erweiterungen erzeugt. Zusätzlich generiert diese Erweiterung die webbasierten Darstellungen der Knoten und Kanten, damit Änderungen am Modell auch über einen Browser betrachtet werden können.

3.6 Web-Interface

Nicht alle Geräte können Eclipse und somit GMF-Editoren nutzen. Mobile Endgeräte, welche keine Java-Laufzeitumgebung unterstützen, sind per se von dieser Nutzung ausgeschlossen. Aber selbst auf Endgeräten, die über eine entsprechende Umgebung verfügen, kann die Nutzung von Eclipse auf Grund mangelnder Systemressourcen unterbunden werden. Um trotzdem nicht auf die

kollaborative Arbeit verzichten zu müssen, können Diagrammänderungen in einem WebViewer (vgl. Abbildung 2) verfolgt werden. Durch diesen wurde eine Möglichkeit geschaffen GMF-Diagramme mit Hilfe von Web-Technologien in einem Browser visualisieren zu können.

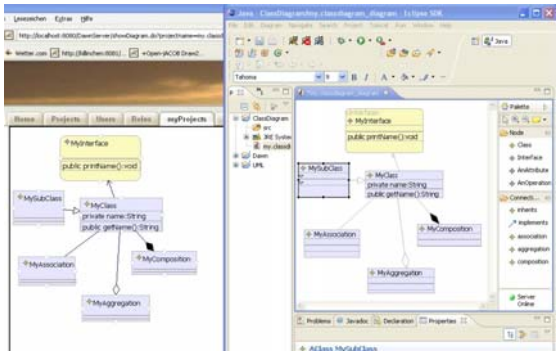

Abbildung 2 - Vergleich WebViewer (link) und GMF Editor (rechts)

Als Grundlage dient die Transformation der in GMF definierten Modelldaten in eine geeignete Visualisierungsform. Unter Verwendung des openArchitectureWare Frameworks XPand können HTML- und JavaScript-basierte Views generiert werden. Diese Transformation basiert auf den Modellen, die ebenfalls von GMF genutzt werden, um die grafischen Repräsentationen (Figures) der Diagramm-Elemente zu visualisieren. Die generierten JavaScript-Klassen basieren dabei in ihrer Funktionalität auf dem Open Source Projekt OpenJacob [7]. Die so generierten Elemente stellen die Visualisierung der Java-Figures im Web dar. Da ihre Erstellung ausschließlich auf dem GMF-Modell beruht, wird, sobald das Aussehen über das Modell geändert wird, auch die Visualisierung im Browser mit verändert, so dass keine Inkonsistenzen zwischen dem GMF-Editor und dem Pendant im Web entstehen können.

3.7 Autorisierung und Authentifizierung

Um das verteilte System vor unbefugtem Zugriff zu schützen, wurden Sicherheits-maßnahmen entwickelt. Um Zugang zu den Daten zu erhalten, muss der Nutzer sich am System authentifizieren. Zusätzlich besteht die Möglichkeit Manipulationen am Modell zu autorisieren. So können Rechte für das Erstellen, Verändern oder Löschen von Objekten vergeben werden. Diese Rechte werden kann derart vom System ausgewertet, dass der Editor beispielsweise auf nur Lese-Rechte reduziert werden kann.

4 Zusammenfassung und Ausblick

Dawn bietet eine für den Nutzer einfache Möglichkeit der kollaborativen Nutzung von GMF-Editoren. Dabei muss nicht auf Features wie Sicherheit, Authentifizierung oder Autorisierung verzichtet werden. Die flexible Kommunikations-struktur erlaubt es das System auch in stark reglementierten Netzwerken zu nutzen, beziehungsweise Clients aus derartigen Netzen mit in das System einzubinden. Durch das einfache Erweiterungskonzept der Kommunikationsadapter können eigenen Protokolle spezifiziert und so Dawn besser in die eigene Netzwerkstruktur integriert werden. Der Offline Modus sowie die WebViewer-Komponente erlauben eine stärkere Fokussierung auf mobile Endgeräte.

5 Quellen

[1] Graphical Modeling Framework
http://www.eclipse.org/modeling/gmf/

[2] Thomas Stahl, Markus Völter, Sven Efftinge, Arno Haase:
Modellgetriebene Softwareentwicklung (2. Auflage)
dpunkt.verlag GmbH, Heidelberg, 2007

[3] Eclipse Modeling Framework
http://www.eclipse.org/modeling/emf/

[4] Graphical Editing Framework
http://www.eclipse.org/gef/

[5] Dawn
http://www.mftech.org/dawn

[6] Elipse M2T Project - XPand
http://www.eclipse.org/modeling/m2t/?project=xpand

[7] OpenJacob Projekt
http://www.openjacob.org/