SHORE

Ein Werkzeug für übergreifende Vernetzung und Auswertung von Dokumenten

Michael Meyer, Helge Schulz

sd&m
software design & management
GmbH & Co. KG

Zusammenfassung

Das ***s**d&m **H**ypertext **O**bject **Re**pository* (SHORE) kombiniert die Eigenschaften eines Repositories mit denen eines Hypertextsystems. Es kann alle Grundlagen und Arbeitsergebnisse eines Softwareentwicklungsprojekts (Pattern, Konzepte, Anforderungen, Schnittstellen, Entwürfe, Quelltexte, Testfälle, Testprotokolle, Fehlerbeschreibungen, Änderungsprotokolle usw.) aufnehmen und ermöglicht darüber eine komfortable Navigation inklusive komplexer Anfragemöglichkeiten. Arbeitsergebnisse gelangen in Form von Dokumenten in das System, die durch dokumentypspezifische Parser untersucht und in eine einheitliche Form (*Extensible **M**arkup **L**anguage*, XML) gebracht werden. Der Inhalt der Dokumente stellt ein *Modell* des Softwaresystems dar. Welche Dokumenttypen existieren und welche Dinge samt ihrer Beziehungen untereinander für die Navigation und Anfragen relevant sind, bestimmt ein frei definierbares *Metamodell* in Verbindung mit den projektspezifischen Parsern. SHORE arbeitet als Client/Server-System bei dem als Server ein erweiterter Web-Server zum Einsatz kommt und als Client ein marktüblicher Web-Browser dient.

Einleitung

Softwareprojekte gehören zu den aufwendigsten, komplexesten und teilweise risikoreichsten Vorhaben, die Wirtschaftsunternehmen durchführen. Viele Unternehmen des wachsenden Dienstleistungsbereichs betreiben nur eine reine Informationsverarbeitung, deren Erfolg unmittelbar von den betrieblichen Informationssystemen abhängt. Als Softwarehaus für derartige „Kerngeschäftssysteme“ führt sd&m ebenfalls eine reine Informationsverarbeitung durch. Dabei werden die Anforderungen unseres Kunden in verschiedenen Phasen in ein Informationssystem transformiert. Wegen der großen Informationsmenge kommt es darauf an, daß die vielen Einzelfakten leicht gefunden, durch entsprechende Anfragemöglichkeiten vollständig genutzt und über die verschiedenen Phasen konsistent bleiben. Je besser dies gelingt, um so weniger Aufwand entsteht durch Informationssuche, Einarbeitung, Abhängigkeitsanalysen und sich einschleichende Inkonsistenzen. Letztendlich kommt eine Verbesserung in diesem Bereich der Qualität des Resultats und der Effizienz des Prozesses zu gute.

Für die Verwaltung aller anfallenden Informationen in einem Softwareprojekt werden Repositories oder Data-Dictionaries eingesetzt. Sie verwalten Fakten zu einzelnen Aspekten oder Teilen des Softwaresystems und ihrer Beziehungen untereinander. Hierfür bieten sie auch entsprechende Anfragemöglichkeiten an. Da es sich dabei um Ergebnisse des Softwareentwicklungsprozesses handelt, werden sie bei sd&m auch als

Ergebnisdatenbanken bezeichnet. Welche Informationen sie speichern, hängt meist eng von der Methodik und/oder Entwicklungsumgebung (CASE-Tool, Programmiersprache, Datenbank usw.) ab, für die sie entwickelt wurden. Leider lassen sich viele Werkzeuge in diesem Punkt nicht an das jeweilige Projekt anpassen, so daß der Entwickler nur das speichern kann, wofür das Werkzeug eine Schublade vorsieht. Die Ablage von isolierten Fakten bringt ebenfalls einige Probleme mit sich. Denn einzelne Fakten sind aus dem Zusammenhang gerissen oft schwer verständlich, die Verantwortlichkeiten lassen sich nur schwer bestimmen und eine Versionsführung stellt sich sehr aufwendig dar.

Mit Dokumenten existiert eine bewährte Gliederungseinheit zur Zusammenfassung von Einzelfakten. Diese werden auch schon seit je her in Softwareprojekten eingesetzt. Denn bei allen fachlichen/technischen Spezifikationen, Programmquellen, Testfallspezifkationen, Protokollen, Änderungsanforderungen usw. handelt es sich in der Regel um Dokumente. Deshalb entstand bei sd&m der Begriff der dokumentenorientierten Softwareentwicklung [1]. Bei einer derartigen Arbeitsweise erstellen die Teammitglieder verschiedene Arten von Dokumenten, die von der Anforderungsbeschreibung bis zur Programmquelle immer formaler werden und aufeinander aufbauen. Demgegenüber steht die transaktionsorientierte Vorgehensweise, die viele CASE-Tools nahe legen. Bei dieser bearbeitet der Entwickler viele einzelne Fakten mittels vorgegebener Dialoge und stellt den Zusammenhang hauptsächlich über Grafiken her. Wegen der oben genannten Problematik der Verwaltung von Einzelfakten ziehen viele sd&m Projekte eine dokumentenorientierte Arbeitsweise vor.

Zur Unterstützung der dokumentenorientierten Arbeitsweise eignen sich Hypertextsysteme sehr gut. Sie können den Zusammenhang zwischen den in den Dokumenten beschriebenen Fakten über Querverweise herstellen und über entsprechende Navigationsmöglichkeit dem Benutzer eine komfortable Unterstützung beim Erfassen und Prüfen von Sachverhalten geben. Üblicherweise muß der Benutzer eines solchen Systems die Querverweise in den zu Grunde liegenden Dokumenten speziell kennzeichnen. Diese Arbeit läßt sich aber durch Scanner oder Parser automatisieren. Dabei tritt das Problem auf, daß bei der Bearbeitung eines Dokuments nicht klar sein muß, auf welche Textstelle in einem anderen Dokument eigentlich der Verweis erfolgen muß. Deshalb liegt es nahe, den Ansatz eines Repositories mit dem eines Hypertextsystem zu kombinieren. Dabei sammeln die Scanner oder Parser Informationen auf, die in einem Repository abgelegt werden. Eine Hypertextsystem setzt dann darauf auf und kann mittels Anfragen an das Repository eine entsprechende Navigation anbieten.

Anforderungen

Die Anforderungen ermitteltete das SHORE Projekt durch Befragung repräsentativer sd&m-Projekte. Ein Ausschnitt aus der Anforderungsliste ist hier aufgeführt:

- Verwaltung von Dokumenten, enthaltenen Objekten und Verweisen dazwischen
- Hypertextnavigation über Dokumente (über alle in den Dokumenten enthaltenen Objekten)
- frei definierbares Metamodell
- Scanner-/Parser API für C und Perl
- DB-API (analog zu GET_OBJ, GET_REL) für C und Perl
- einfache Query-Möglichkeiten
  - select from where, ohne Verknüpfungen
  - insoweit rekursiv, daß man einen Funktionsaufrufbaum erstellen kann
  - Beantwortung von in Textdateien hinterlegten Anfragen (werden evtl. beim Starten des Servers übersetzt)
- Volltextsuche auf den im Repository abgelegten XML-Dokumenten
- offene Verweise zugelassen

SHORE

- Anbindung einer Versionsverwaltung
- Oberfläche ist ein marktgängiger oder freier WWW-Browser
- Transaktionsmechanismus
- Multi-User-Fähigkeit
- Dokumentenorientierung
- Eignung für größere Projekte
  - 50 Entwickler
  - 200.000 Objekte
  - 1.000.000 Beziehungen
  - 20.000 Dokumente
  - 2 Mio LOC

Dokumente, Objekte und Beziehungen

Alle Informationen, die das Hypertext-Repository SHORE aufnimmt, gelangen in Form von Dokumenten in das System. *Dokumente* bilden zu einem bestimmten Zeitpunkt vorhandenes Wissen ab, das durch einen oder mehrere Autoren zusammengetragen wurde und vom adressierten Leser ohne zusätzliche Hilfe verstanden werden kann. Daher zählen zu den Dokumenten auch Grafiken und Diagramme mit obigen Eigenschaften (insbesondere mit einer festgelegten Semantik). Dokumente liegen üblicherweise in Form von Dateien vor.

In den Dokumenten sind sog. *Objekte* durch einen Abschnitt des Dokuments definiert. Objekte können andere enthalten, indem ein Unterabschnitt des ursprünglichen Abschnitts ein weiteres Objekt beschreibt. Objekte und Dokumente können Verweise auf Dokumente und Objekte beinhalten. Durch diese Verweise entstehen gerichtete *Beziehungen* zwischen den Ausgangsdokumenten/-objekten und dem jeweiligen Ziel, die bidirektional genutzt werden können. Die „enthalten sein“-Beziehung zwischen Objekten verwaltet SHORE nicht implizit, statt dessen müssen explizit Verweise eingefügt sein.

Dokumente, Objekte und Beziehungen sind getypt. Die Typen und ihre Beziehungen untereinander legt der Benutzer in einem frei konfigurierbaren *Metamodell* pro Repository fest. Das Metamodell liegt wiederum in Form eines Dokuments vor, das die Typen als Objekte enthält. Zwischen Dokument-, Objekt- und Beziehungstypen gibt es multiple Vererbung bezogen auf die jeweilige Art des Typs. Typen, von denen andere Typen erben, werden als *abstrakte* Typen bezeichnet. Von ihnen können keine Instanzen angelegt werden. Wenn beispielsweise der Objekttyp „C-Funktion“ von dem abstrakten Objekttyp „Programmelement“ erbt, bedeutet dies, daß es in der SHORE-Datenbasis keine Objekte vom Typ „Programmelement“ geben darf.

Eine einfache Vererbung auf der Ebene der Ressourcetypen (Objekt- und Dokumenttypen) ist als Spezialisierung zu verstehen. Eine Mehrfachvererbung zwischen Ressourcetypen kann man sich als eine Vereinigung der Merkmale der Obertypen vorstellen. Der Wertebereich des abgeleiteten Typs setzt sich aus der Schnittmenge der Instanzen aller Obertypen zusammen. Beispielsweise könnte der Objekttyp „INOUT-Parameter“ von den beiden Obertypen „IN-Parameter“ und „OUT-Parameter“ erben.

Vererbung auf Beziehungstypen läßt sich als Teilmengenbeziehungen auffassen. Beispiel: Die Vererbungsbeziehung „ruft“ erbt von „benutzt“ bedeutet anders ausgedrückt, daß die Menge aller Instanzen der „ruft“-Beziehung (Untertyp) eine Untermenge aller Instanzen der „benutzt“-Beziehung (Obertyp) ist. Multiple Vererbung auf Beziehungstypen kann man sich als das Bilden der Schnittmenge aller Instanzen der Obertypen vorstellen. Ein Beispiel hierfür wäre, daß der Beziehungstyp „ist-IN/OUT-Parameter“ von den beiden Obertypen „ist-IN-Parameter“ und „ist-OUT-Parameter“ erbt.

Jede Beschreibung eines Objekt- und Beziehungstyps im Metamodell legt fest, in welchen Typen von Dokumenten Instanzen bzw. Verweise vorkommen können. Auch den Ziel- und Quelltyp eines Beziehungstyps gibt der Benutzer dort an. Vererbung auf Beziehungstypen schränkt damit auch die Ziel- und Quelltypen von „erbenden“ Beziehungstypen ein. Beim obigen Beispiel heißt letzteres, daß die „ruft“ Beziehungen nur bei Objekten/Dokumenten beginnen oder enden kann, deren Typ bei „benutzt“ festgelegt wurde - oder Untertypen davon.

Architekturübersicht und Datenfluß

Bei SHORE handelt es sich um ein Client/Server-System. Jede *SHORE-Server-Instanz* verwaltet eine Menge von Dokumenten und alle Objekte/Beziehungen daraus auf Basis eines Metamodells. Jede Instanz läuft auf einer Server-Maschine und läßt sich von den Clients unter einer Netzwerkadresse ansprechen. Auf einem Rechner können mehrere Instanzen gleichzeitig unter verschiedenen Netzwerkadressen unabhängig voneinander arbeiten. Abbildung 1 zeigt eine Architekturübersicht mit allen wichtigen Komponenten. Die Pfeile geben den Datenfluß zwischen dem SHORE-System und seiner Umgebung sowie innerhalb des Systems wieder.

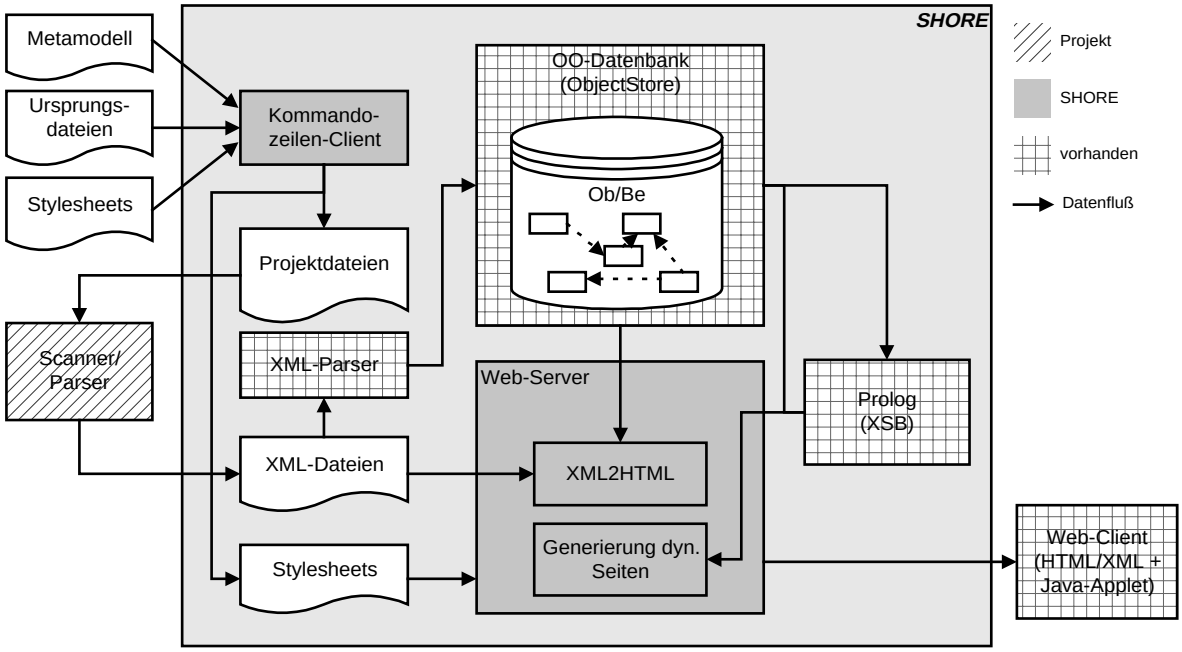

Abbildung 1: SHORE Architekturübersicht

Dokumente gelangen durch den Aufruf eines Kommandozeilen-Clients in das SHORE-System. Dieser Client läuft auf dem jeweiligen Client-Rechner und überträgt die (Ursprungs-) Datei über das Netzwerk zum Server-Rechner. Dort angelangt, legt der Server sie in einer *Projektdokumentenablage* ab. Um die Objekte und Beziehungen eines Dokuments erfassen zu können, muß der Benutzer für jeden Dokumenttyp einen entsprechenden *Scanner/Parser* auf dem Server-Rechner bereitstellen. Dieser wandelt die Ursprungsdatei in eine XML-konforme Datei um. Bei der Umwandlung markiert der Parser Objekte und Beziehungen gemäß dem XML-Nutzungskonzept. Anschließend untersucht ein *XML-Parser* die entstandene XML-Datei und trägt Objekte sowie Beziehungen in eine *OO-Datenbank* ein.

Der gesamte Import eines Dokuments von seiner Übertragung auf den Server-Rechner bis zum Schreiben in die Datenbank ist unteilbar und erfolgt in einer Transaktion. Hat sie Erfolg, so wird die XML-Datei in der

XML-Dateiablage für die Hypertextnavigation freigegeben. Ansonsten wird die alte Projektdatei wieder hergestellt, die Datenbankänderungen zurückgenommen und weiterhin die alte XML-Datei benutzt.

Handelt es sich bei der Ursprungsdatei schon um eine XML-Datei oder XML konforme Datei (z.B. HTML), so tut der Parser entweder nichts (Dummy-Parser) oder führt nur die Markierung der Objekte und Beziehungen durch. Zusammengesetzte Dokumente mit Grafiken u.ä. kann der Parser in diesem Schritt auch in mehrere Dateien zerlegen.

Als primäre Benutzerschnittstelle dient bei SHORE ein Web-Client in Form eines handelsüblichen HTML- oder XML-*Browsers*. Er kommuniziert über das Hypertext Transfer Protokoll (HTTP) mit einem erweiterten SHORE *Web-Server*, der Teil einer SHORE Server-Instanz ist. Dieser stellt die entsprechenden Hypertext-Navigationsmöglichkeiten bereit. Handelt es sich beim Client um einen HTML-Browser, so erfolgt beim Versenden des Dokuments eine online XML nach HTML Konvertierung. Der Konverter (Komponente *XML2HTML* in der Abbildung) wandelt dabei nur Auszeichnungen (Markup) in HTML um, die Objekte und Beziehungen laut des jeweiligen Metamodells markieren. Dadurch wird ermöglicht, daß Ursprungsdateien in HTML wieder korrekt angezeigt werden. Beim Umwandlungsschritt erfolgt auch eine besondere Markierung des Zielobjekts bei der Navigation, so daß es bei der Ausgabe hervorgehoben und vom Benutzer leichter auffindbar ist. Wie diese Hervorhebung aussieht, bestimmt ein Stylesheet, auf das das Dokument verweist und so vom Browser automatisch abgerufen wird.

Der Web-Client erhält neben den Dokumenten auch Programmcode in Form von Java und/oder Javascript, der Menüs und Dialoge realisiert, die zur Ausführungen von Suchen sowie zur Definition und Ausführungen von Anfragen usw. dienen.

Die Abarbeitung von Anfragen erledigt ein *Prolog-System*, das deterministische Antworten liefert und keine doppelten Ergebnissätze zurückgibt. Es arbeitet eng gekoppelt mit der OO-Datenbank, so daß Objekte und Beziehungen nicht doppelt gespeichert werden müssen. Erste Ergebnisse können am Browser schon vor dem Ende der gesamten Anfragen angezeigt werden, da das Prolog-System diese satzweise übergibt. Die Transaktionssteuerung erfolgt so, daß die Anfrage sich immer auf den Stand der Datenbank bezieht, wie er beim Start der Anfrage vorlag. Der Benutzer kann frei formulierte Prolog-Anfragen oder hinterlegte Standardanfragen absetzen. Bei der Formulierung der Anfragen kann der Benutzer auf Prolog-Prädikate zurückgreifen, die direkt den Begriffen aus dem Metamodell entsprechen.

Einbindung von SHORE in die Entwicklungsumgebung

SHORE benötigt keine spezielle Einbindung in die Entwicklungsumgebung. Dank des Kommandozeilen-Clients kann der Im- und Exporten von Dokumenten flexibel gesteuert werden. Z.B. kann der Import über ein Skript mit dem Konfigurationsmanagement des Projekts gekoppelt werden, so daß bei der Freigabe einer Dokumentversion (check in) immer ein Import erfolgt.

Für spätere Stufen ist eine engere Kopplung mit der Entwicklungsumgebung vorstellbar, so daß z.B. Build-Vorgänge und die Steuerung des Konfigurationsmanagements aus der SHORE Oberfläche heraus erfolgen.

Benutzergruppen

Das SHORE-System hat mehrere Benutzergruppen (*Actors*). Diese stellen nicht konkrete Benutzer dar, sondern Rollen, die von SHORE-externen Personen angenommen werden können. Ein konkreter SHORE-Benutzer kann dabei je nachdem, was er mit dem System tun möchte, verschiedene Rollen annehmen und damit unterschiedliche Benutzergruppen repräsentieren.

Zu den Actors im weiteren Sinne gehören jedoch auch SHORE-externe Systeme, deren Dienste SHORE benutzt bzw. die SHORE-Dienste verwenden.

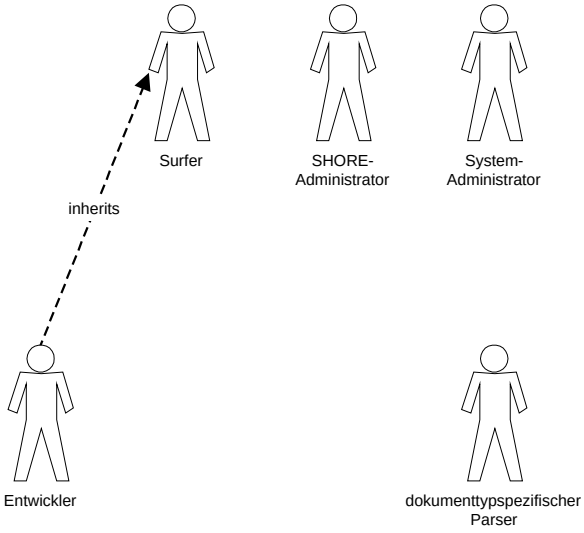

Abbildung 2: Benutzer des SHORE-Systems

Surfer

Der Surfer ist ein Benutzer, der nur durch die Dokumente einer SHORE-Anwendung navigiert und Anfragen an die SHORE-Datenbasis stellt. Surfer sind z.B. Mitarbeiter der auftraggebenden Fachabteilung oder QS-Beauftragte.

Entwickler

Der Entwickler ist ein Benutzer, der zusätzlich zu den Interaktionen des Surfers auch Dokumente mit dem SHORE-System austauscht, also z.B. neue Projektdokumente in das SHORE-System stellt. sd&m-Mitarbeiter in Projekten sind typische Entwickler im SHORE-Sinne.

SHORE-Administrator

Der SHORE-Administrator kümmert sich um die korrekten projektspezifischen Einstellungen des SHORE-Systems. Dazu gehört z.B. das Verwalten von dokumenttypspezifischen Parsern und projektspezifischen Anfragen an die SHORE-Datenbasis. Üblicherweise wird ein sd&m-Projektmitarbeiter die Aufgabe des SHORE-Administrators übernehmen.

System-Administrator

Der System-Administrator ist verantwortlich für die Maschine, auf der der SHORE-Server läuft. Zu seinen Aufgaben gehören die Installation und Deinstallation des SHORE-Servers und das Backup bzw. Restore des SHORE-Systems. Der System-Administrator wird in den meisten Fällen ein TI-Mitarbeiter sein.

Dokumenttypspezifischer Parser

Dokumenttypspezifische Parser sind keine menschlichen Benutzer, jedoch außerhalb des SHORE-Kernsystems liegende Systeme, die von diesem aufgerufen werden. Die dokumenttypspezifischen Parser werden entweder von sTm bereitgestellt oder vom SHORE-Administrator erstellt.

SHORE

Metamodellnotation

Im Metamodell wird festgelegt, welche Dokument-, Objekt- und Beziehungstypen es gibt, welches die Quell- und Zielobjekttypen der Beziehungstypen sind, welche Vererbungsbeziehungen es auf den einzelnen Typen gibt, sowie welche Objekttypen in welchen Dokumenttypen definiert werden und welche Beziehungstypen in welchen Dokumenttypen vorkommen dürfen. Dieses Metamodell ist jeweils projektspezifisch und muß dem SHORE-System in einer Metamodellspezifikation gemäß folgender Notation als Dokument zur Verfügung gestellt werden. Eine Metamodellspezifikation besteht aus drei Typen von Einträgen:

1. Deklaration von Dokumenttypen mit optionaler Angabe von Dokumenttyp-Supertypen und der Information, ob Dokumente dieses Typs für die Volltextsuche indiziert werden, in der Form

```
Dokumenttyp Dokumenttypname
    ( ist (ein|eine) Dokumenttypname )*
    [ ist nicht volltextindizierbar ]
```

2. Deklaration von Objekttypen mit optionaler Angabe der Dokumenttypen, in denen sie definiert werden und optionaler Angabe von Objekttyp-Supertypen in der Form

```
Objekttyp Objekttypname
    ( ist (ein|eine) Objekttypname )*
    ( ist definiert in Dokumenttypname )*
```

3. Deklaration von Beziehungstypen mit Angabe der Quell- und Zielresource und deren Kardinalitäten, mit optionaler Angabe der Dokumenttypen, in denen sie definiert werden können und optionaler Angabe von Beziehungstyp-Supertypen in der Form

```
Beziehungstyp Beziehungstypname [ alias Aliasname]
    von n1 bis n2|* Resourcetypname
   nach m1 bis m2|* Resourcetypname
    ( ist Teilmenge von Beziehungstypname )*
    ( ist definiert in Dokumenttypname )*
```

Die Reihenfolge der Einträge ist dabei beliebig, Schlüsselwörter und Bezeichner werden durch beliebige Whitespace-Zeichen (Newline, Tab, Blank) voneinander getrennt, mit ‘#’ an beliebiger Stelle einer Zeile kann der Rest der Zeile inklusive des Zeilenumbruchs als Kommentar gekennzeichnet werden.

Alle vorkommenden Typen müssen in genau einem Eintrag deklariert werden. Jeder Typ muß durch seinen Namen eindeutig gekennzeichnet sein. Namen mit gleicher Buchstabenreihenfolge aber unterschiedlicher Groß-/Kleinschreibung gelten als gleich (nicht „case sensitive“). Für Prolog werden die Anfangsbuchstaben der Namen ggf. in Kleinbuchstaben umgewandelt.

Bei der Spezialisierung von Beziehungstypen ist zu beachten, daß die zugehörigen Quell- und Zielressourcen entweder gleich bleiben oder ebenfalls spezialisiert werden. Zyklische Vererbungsbeziehungen sind nicht erlaubt.

Transitiv erschließbare Vererbungsbeziehungen müssen nicht explizit angegeben werden (z.B. ‘C-Funktion ist eine Funktion’ und ‘Funktion ist ein Programmelement’ impliziert ‘C-Funktion ist ein Programmelement’).

SHORE

Aufruf von Scannern/Parsern

Von SHORE wird ein Parser aufgerufen, der ein Projektdokument nach XML transformiert und eventuelle nichttextuelle Teile als Entitätendateien extrahiert. In SHORE ist dazu abgelegt, welcher Parser für welche Dateiendungen verantwortlich ist.

Die Dateiendungen sind dabei unabhängig vom Dokumenttyp des Dokuments. So können z.B. Word-Dokumente (.DOC) Spezifikationen, aber auch Testfälle enthalten. In solchen Fällen hat das SHORE einsetzende Projekt zwei Möglichkeiten:

- Es gibt einen Parser, der Word-Dokumenten zugeordnet ist. Dieser analysiert die Word-Dokumente und bestimmt ihren Dokumenttyp. In Abhängigkeit von diesem Ergebnis (Spezifikation oder Testfall) werden unterschiedliche XML-Dokumente erzeugt. Der Parser kann hierfür andere Programme (z.B. einen Parser *nur* für Spezifikationen) aufrufen.

- Beim Import eines Dokuments wird der Name des Parsers explizit mit angegeben. In SHORE hat dies Vorrang vor der automatischen Auswahl eines Parsers anhand der Dateiendung.

Dieser Parser bekommt die folgenden beiden Argumente übergeben:

1. den Namen der zu transformierenden Projektdatei inklusive relativem Pfad

2. den relativen Pfad zum Zielverzeichnis

Der Parser liest die angegebene Projektdatei und schreibt das erzeugte XML-Dokument (unter dem selben Namen wie die Projektdatei) und etwaige Entitätendateien in das angegebene Zielverzeichnis.

XML-Nutzungskonzept

In dieser ersten Stufe des SHORE-Systems ist das XML-Nutzungskonzept so einfach wie möglich gehalten. Insbesondere wird auf die Prüfung (Validierung) von Dokumenten an Hand einer XML-Dokumenttypbeschreibung (DTD) verzichtet und auf die Verwendung von Namespaces und der erweiterten Linking-Möglichkeiten der „XML Linking Language“. Die XML-Dokumente enthalten außerdem keine Informationen, die nur für die Navigation nötig wären. Diese Informationen trägt der XML zu HTML Konverter oder später ein XSL Stylesheet ein.

Die Nutzung von XML in SHORE folgt folgenden Grundsätzen:

- Es wird keine DTD verwendet, insbesondere werden die XML-Dokumente auch nicht validiert. Die Dokumente müssen jedoch wohlgeformt sein, insbesondere im Bezug auf das Inkludieren von externen Entitäten, die Quotierung von Sonderzeichen (<,>,&,..) und die XML-Deklaration.

- Die Menge der für das SHORE-System relevanten Tags wird dem System durch das Metamodell bekannt gemacht, alle anderen, unbekannten Tags werden beim Import ignoriert bzw. bei der Anzeige samt Inhalt kopiert.

- Die Navigationsmöglichkeiten erscheinen nicht im XML-Dokument, sondern werden bei der HTML-Konvertierung durch die Erzeugung von HREF-Attributen mit ‘find’- bzw. ‘info’-Anfragen an den Server als Inhalt realisiert oder später durch ein XSL-Stylesheet.

- Bei der Anzeige wird das vom Benutzer für den jeweiligen Dokumenttyp gewählte Stylesheet verwendet.

Anfragemöglichkeiten

Wie bei der Architekturübersicht angegeben verwendet das SHORE-System als Query-Maschine ein Prolog-System. An dieses kann der Benutzer entweder frei formulierte ad-hoc Anfragen oder vordefinierte Standardanfragen richten. Da sich Regeln auch sehr einfach rekursiv definieren lassen und auch kombinatorische Nebenbedingungen einfließen können, ist ein derartiges *deduktives Anfragesystem* mächtiger als andere häufig benutzte Anfragesprachen wie SQL oder OQL. Durch den Einsatz des XSB-Prolog-Systems [2] konnte auf eine bewährte Sprache bzw. System zurückgegriffen werden und auf die Entwicklung einer eignen Abfragesprache, wie z.B. GReQL [3], verzichtet werden. Das Prolog-System arbeitet eng gekoppelt mit der OO-Datenbank zusammen, so daß Daten/Fakten nur bei bedarf aus der Datenbank gelesen werden. Die praktische Machbarkeit einer solchen Kopplung überprüften wir an einem Prototypen, der im Rahmen einer Diplomarbeit [4] untersucht und vervollständigt wurde.

Als Anfragesprache dienen bei SHORE Prolog-Ausdrücke mit der Semantik von erweitertem Datalog. Die Ausdrücke enthalten logisch verknüpfte Prädikate und freie Variablen. Die Prädikate sind entweder durch Regeln selbstdefiniert oder vordefiniert. Zu jedem im Metamodell definierten Typ existieren mehrere vordefiniertes Prädikat verschiedener Stelligkeit, so daß sich sehr leicht Bedingungen für die gewünschte Zielmenge aus Instanzen dieser Typen angeben lassen. Bei der Auswertung der Anfrage sucht das Prolog-System alle mögliche Belegungen der freien Variablen heraus, bei denen der Anfrageausdruck insgesamt den Wert „wahr“ ergibt. Selbstverständlich kann eine freie Variable an mehreren Stellen vorkommen und es können in einer Anfrage auch mehrere verschiedene freie Variablen stehen. Das Ergebnis besteht aus einer Liste der möglichen Belegungen. Jede Belegung steht in einer eigenen Zeile wobei die Werte für verschiedene Variablen durch ein Komma getrennt sind. Wenn es sich bei den Werten um SHORE-Resourcen (Dokument oder Objekt) handelt, so gibt SHORE den Typ- und Resourcenamen als Hyperlink aus. Bei SHORE-Beziehungen gibt es jeweils beide Werte für Ziel und Quelle sowie den Beziehungstypnamen aus. Die Hyperlinks von einem Resourcennamen verweisen wie üblich auf die Definitionstelle des Objekts bzw. auf das Dokument.

Ausblick

SHORE muß sich jetzt im Einsatz in unseren Projekten bewähren. Die Praxis wird zeigen, an welcher Stelle noch weitere Funktionalität erforderlich ist. Folgende Erweiterungen sind für SHORE Stufe 2 jetzt schon geplant:

Integration externen Wissens

Externen Wissen soll integriert werden können, d.h. Objekte und Beziehungen in der SHORE-Datenbasis abzulegen und zu verwalten, die ihren Ursprung nicht in den XML- und damit den Projektdokumenten haben.

Attribute zu Objekten und Beziehungen

XML erlaubt, Informationen zu XML-Elementen (Tags) in beliebig vielen Attributen abzulegen. Diese Attribute benutzen die dokumenttypspezifischen Parser in SHORE, um gemäß dem XML-Nutzungskonzept die für die Datenbasis notwendigen Informationen in den XML-Dokumenten abzulegen.

Wenn ein SHORE einsetzendes Projekt es für sinnvoll hält, Informationen, die die dokumenttypspezifischen Parser beim Parsen der Projektdokumente aufsammeln, aufzubewahren, können diese Informationen in zusätzliche Attribute zu den XML-Elementen kodiert werden. Beispiele hierfür sind die Zeilen- und Spaltenposition einer Objektdefinition innerhalb eines Projektdokuments.

In SHORE Stufe 2 werden diese Attribute auch in der Datenbasis abgelegt und können über die Query-Engine abgefragt werden. Die Kodierung dieser Attribute in die XML-Dokumente ist jedoch auch schon in SHORE

Stufe 1 erlaubt, jedoch müssen die Projekte für die Auswertung dieser Attribute selbst die erzeugten XML-Dokumente auswerten.

Auswahl passender Anfragen zu einer Ressource

In SHORE Stufe 2 wird für jede Standardanfrage hinterlegt, welche Typen ihre Parameter annehmen können. Im Umkehrschluß kann damit für einen Typ des Metamodells herausgefunden werden, in welchen Standardanfragen eine Instanz dieses Typs als Parameter auftauchen kann.

Damit wird es für den Benutzer des SHORE-Systems möglich sein, zu einer angezeigten Ressource (ein Dokument, Objekt oder eine Relation) aus den für diese Ressource in Frage kommenden Standardanfragen eine auszuwählen, deren passenden Parameter mit dem Wert der aktuellen Ressource zu füllen und die Anfrage direkt auszuführen. Sollte die Standardanfrage noch weitere Parameter benötigen, müssen diese vor dem Ausführen der Anfrage explizit gefüllt werden.

Ausbau graphischer Darstellungen

Im Rahmen einer weiteren Diplomarbeit [5] wurde für das SHORE-Sytem eine graphische Erweiterung erstellt, die den Navigationspfad durch das System als Graphen darstellt und zur Orientierung eine graphische Darstellung des Metamodells bereithält. Durch Markieren von Knoten und Kanten im Metamodellgraphen können außerdem bestimmte Anfragen formuliert und abgesetzt werden. Diese graphischen Möglichkeiten könnten in späteren Stufen ausgebaut werden, so daß das Objekt/Beziehungsgepflecht direkt visualisiert werden könnte.

Literaturverzeichnis

[1] Ernst Denert. *Dokumentorientierte Software-Entwicklung*. Informatik Spektrum, Band 16, Heft 3, S. 159-164, Juni 1993.

[2] K. F. Sagonas, T. Swift, D. S. Warren, J. Freire, P.Rao. *The XSB Programmer's Manual Version 1.8*. Stony Brook University, New York, 1998 (erhältlich über http://www.cs.sunysb.edu/~sbprolog/xsb-page.html).

[3] Manfred Kamp, *GReQL - Eine Anfragesprache für das GUPRO-Repository - Sprachbeschreibung (Version 1.2)*, Fachbericht Informatik 14/98, Universität Koblenz-Landau, Fachbereich Informatik, 1998 (erhältlich über http://www.uni-koblenz.de/~ist/gupro.html).

[4] Harald Graich. *Verwendung deduktiver Anfragen für das XML-Repository SHORE*. Diplomarbeit, Institut für Informatik, Universität Augsburg, November 1998.

[5] Christine Herzog. *Navigation auf Objekt-Beziehungs-Netzen in XML-Dokumenten*. Institut für Informatik, Ludwig-Maximilians-Universität München, Mai 1999.