# YAWA - YET ANOTHER WRAPPING ARCHITECTURE

## EIN GENERISCHES, XML-BASIERTES UND MODULARES WRAPPER-FRAMEWORK

Martin Lang, Rainer Lay
Department of Database Systems, University Erlangen-Nuremberg
Martensstr. 3, 91058 Erlangen, Germany
Martin.Lang@m-lang.de, Rainer.Lay@cs.fau.de

Im Kontext der Realisierung eines medizinischen Informationssystems wurde zum Zweck der Integration heterogener Quellen in ein globales Schema ein möglichst generisches Framework zur Wrapper-Generierung konzipiert. Dieses Wrapper-Framework stellt die von den meisten Wrappern genutzten Funktionalitäten unabhängig von einem speziellen Ausgangs- und Zielschema sowie Ausgangs- und Zieldatenmodell zur Verfügung. Diese Funktionen, wie die Schemaabbildung, die Datentransformation, die Datenbereinigung sowie die Funktionen zur Fehlerbehandlung wurden in speziellen Komponenten realisert. Der Fokus bei der Umsetzung des Frameworks lag auf der Generierung von Wrapper-Instanzen durch Konfiguration generischer Komponenten. Weitere Entwurfsziele waren eine möglichst einfache Konfigurierbarkeit und Modifizierbarkeit sowie Sicherung der Autonomie der zu wrappenden Datenquellen. Die von diesen Komponenten bereitgestellten

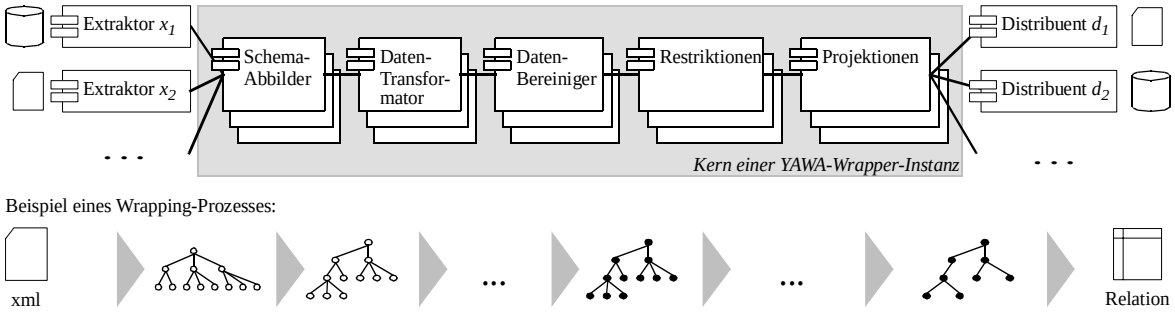


*Abb. 1: Schamtischer Aufbau*

Funktionen sind zudem das Abbilden und Erweitern der Anfrageschnittstelle, der Datenquellenzugriff, die Datenextraktion, die Datentransformation und das Zurückliefern relevanter Daten. Die Metadaten werden für jede Wrapper-Instanz in einer Konfigurationsdatei lokal gehalten.

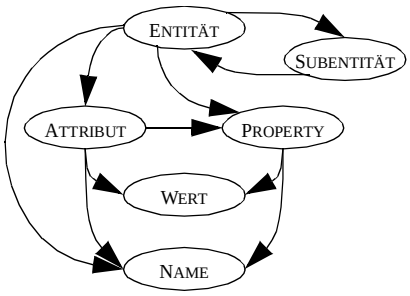


*Abb. 2: Schema des YAWA Datenmodells*

Um die Erweiterbarkeit und Vielseitigkeit des YAWA-Frameworks zu erreichen, wurde ein wrapper-internes, XML-basiertes, generisches Datenmodell zur Entkopplung der Komponenten von Datenmodell und -schema der Quell- und Zielsysteme konzipiert. Die Daten werden im YAWA-Datenmodell als attributierte, gerichtete, zyklenfreie Graphen $G = (V,E, r, v)$ dargestellt [BMNe00]. $E$ stellt dabei die Menge der Kanten dar, die die Enthaltenseinbeziehungen repräsentieren. Die Menge der Knoten $V = V_a \cup V_c$ dieser Graphenstruktur setzt sich aus den komplexen Objekten $V_c$, den Entitäten und den atomaren Objekten $V_a$, den Attributen zusammen. $r \in V$ ist dabei das Wurzelement. Die Relation $v : V_a \rightarrow D$ beschreibt Werte von atomaren Objekten, wobei hier $D$ das Universum der atomaren Werte bezeichnet. Die Metainformationen zu jedem Knoten $v \in V$ werden durch eine Menge von Bezeichner-Wert-Paaren, den Properties repräsentiert. Die in Abbildung 2 aufgeführten Subentitäten sind jeweils Mengen von Entitäten eines bestimmten Typs und ein rein optimierungsspezifischer Aspekt. Weiterhin besitzt das YAWA-Datenmodell diverse Erweiterungen gegenüber reinen XML-Datenmodellen. Unter

Anderem eine expliziten Repräsentation von NULL-Werten sowie Methoden zur Überprüfung von Identität und struktureller Gleichheit zweier Entitäten. Durch die Verwendung von Schemavorlagen für Entitäten (Entity Pattern) werden Default-Werte ermöglicht und eine homogene Struktur von Entitäten gleichen Typs gewährleistet. Dies hat eine Reduzierung der Komplexität der Abbildungsdefinitionen zur Folge.

Beim Wrapping-Prozess (siehe Abbildung 1) werden die Daten durch die Extraktionskomponente aus dem lokalen Schema und Datenmodell der Datenquelle in das wrapper-interne YAWA-Datenmodell, d.h. auf die oben charakterisierte hierachische und zyklenfreie Graphenstruktur abgebildet (Model Mapping). Im Anschluß an das Model Mapping erfolgen die diversen anderen Abbildungen, wie die Schemaabbildung, die Datentransformationen sowie die Datenbereinigung, welche ebenfalls in Komponenten realisiert wurden. Die in Abbildung 3 aufgeführte Mapping Engine beinhaltet diese Komponenten $t_1$ bis $t_n$ und ist ein baumtraversierender Umsetzer (Tree-walking Tree Transducer) [BMNe00] mit Registern, welcher mittels dieser Komponenten die gewünschten Abbildungen durchführt. Eine Abbildung $map$ ist eine gerichtete Abbildung von $E_\Gamma \rightarrow E'_{\Gamma^*}$, einer Entität $E$ eines Typs $\Gamma$ auf eine Entität $E'$ vom Typ $\Gamma^*$. Die Metadaten zu diesen Abbildungen werden ebenfalls in der wrapper-eigenen Konfigurationsdatei spezifiziert. Nach Abschluss der beschriebenen Abbildungen werden die Daten in der gewünschten Struktur, Relevanz und Qualität an die nächst höhere Schicht weiter gegeben. Diese Abbildung des Datenmodells wird von der Ergebnis-Distributions-Komponente vorgenommen.

Hieraus folgt die in Abbildung 3 abgebildete 3-Schicht-Schemaarchitektur einer Wrapper-Instanz, bestehend aus dem Import-, dem internen und dem Export-Schema.

- Die Komponente des Anfragerezeptors $a_i$ übersetzt die Anfragen von der informationssystemglobalen in die wrapper-interne Anfragesprache. Beim Zugriff auf die Datenquelle wird, falls möglich, die Anfrage vom internen Format auf die Anfragesprache und -fähigkeiten der Quelle abgebildet und erweitert. Konzeptionell ist die interne Anfragesprachen nicht festgelegt. Denkbar wären bespielsweise Komponenten für XML-QL, XQL, XQuery, XPath, Quilt oder XSQL. Derzeit ist jedoch lediglich ein primitiver Anfragemechanismus realisiert.
- Die Extraktor-Komponenten $x_j$ sind jeweils für den Zugriff auf eine bestimmte Klasse von Quellsystemen verantwortlich. Sie realisieren die Abbildung vom lokalen in das interne Datenmodell.
- Durch die Transformatorkomponenten $t_1$ bis $t_k$ der Mapping Engine werden unter anderem Schemaabbildungen, Datentransformationen und Datenbereinigungen realisiert.
- Die Komponente des Ergebnisdistribuenten $d_m$ liefert die für die Anfrage relevanten und transformierten Daten in einer bestimmten Struktur und in einem bestimmten Datenmodell an den anfragenden Prozess der übergeordneten Schicht zurück. Diese Komponente realisiert somit die Abbildung aus dem wrapper-internen Datenmodell auf das Datenmodell der übergeordneten Schicht.

Der große Vorteil der YAWA-Architektur ist, dass die Komponenten des Wrapper-Kerns für alle Wrapper-Instanzen wiederverwendet werden können. Dies ist möglich, da diese Komponenten ausschließlich auf dem wrapper-internen Datenmodell operieren. Somit wird der Implementierungs-, Pflege- und Verwaltungsaufwand auf ein Minimum reduziert, da die Implementierungen von den einzelnen Wrappern lediglich instanziert werden müssen. Ein weiterer Vorteil der YAWA-Architektur besteht darin, dass die Anfragerezeptions-, Extraktions- und Ergebnis-Distributions-Komponenten an den Schnittstellen zu den über- bzw. untergeordneten Schichten jeweils Klassen von Quell- bzw.

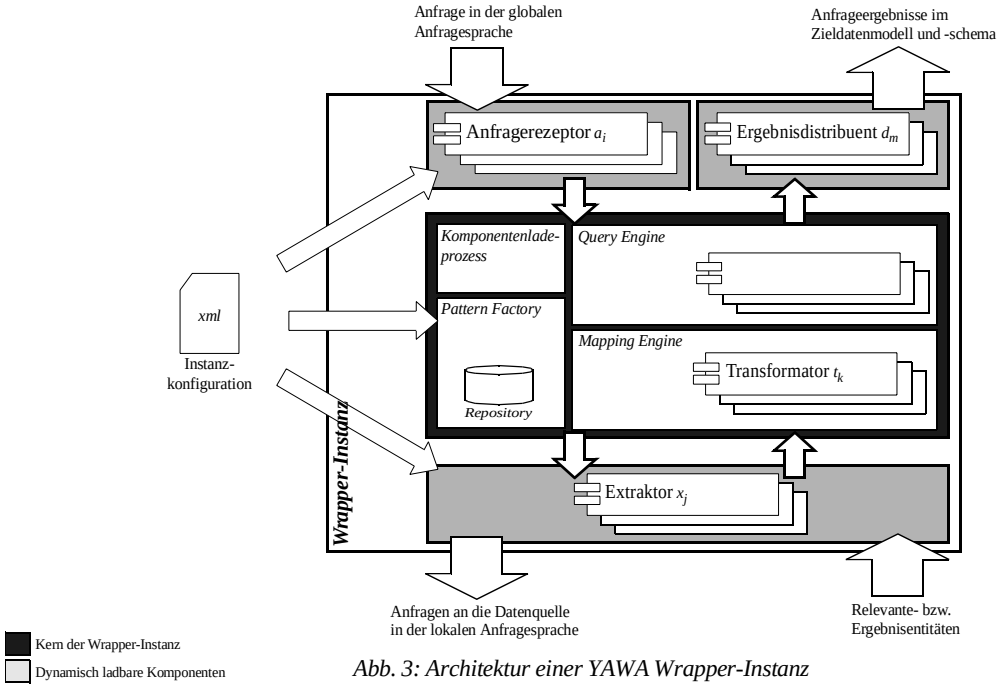


*Abb. 3: Architektur einer YAWA Wrapper-Instanz*

Zielsystemen verarbeiten können. Somit lässt sich beispielsweise eine Instanz der Extraktionskomponente für ODBC-Quellen für beliebige Datenquellen aus dieser Quellklasse konfigurieren. Da meist mehrere Quellen aus einer Quellklasse integriert werden müssen und Informationssysteme üblicherweise eine einheitliche Anfragesprache und ein globales Datenmodell besitzen, wird der Implementierungsaufwand weiter reduziert. Somit lassen sich nahezu beliebige Quellsysteme und Zielsysteme kombinieren, da die Modifikationen der Daten im YAWA-Datenmodell erfolgen und die speziellen Extraktions-, Ergebnis-Distributions- und Anfragerezeptionskomponenten austauschbar sind.

Das YAWA-Wrapper-Framework wurde in der Programmiersprache Java implementiert und wird derzeit anhand von diversen Wrapper-Instanzen im laufenden Klinikalltag der Augenklinik Erlangen verwendet. Hierbei kommen unter anderem Extraktoren für Access97, CSV-Dateien und XML-Dateien zum Einsatz. Die Daten dieser Quellen werden durch eine JDBC Ergebnis-Distributions-Komponente in das Zielsystem integriert.


| | |
|---|---|
| BMNe00 | Bex, G. J.; Maneth, S.; Neven, F.: *A Formal Model for an Expressive Fragment of XSLT*, 1st International Conference of Computational Logic, London, Juli 2000 |
| BMRy99 | Beech, D; Malhotra, A; Rys, M: *A Formal Data Model and Algebra for XML*, http://ww-db.stanford.edu/dbseminar/Archive/FallY99/malholtra-slides/malholtra.pdf, 10. September 1999 |
| JaGr00 | Jackson, S.; Gregorski, B. F.; *The Wrapper Component*, http://sirius.cs.ucdavis.edu/teaching/289F-SQ00/project/Reports/wrapp_init.ps, April 2000 |
| Suci98 | D. Suciu: *An Overview of Semistructured Data*, SIGACT News, 29(4):28-38, Dezember 1998 |