Ontologiebasierte Dokumentindizierung und -suche

Oktober 2000

Jonny Newald

Die größte Dokumentenansammlang der Welt ist das Internet.

Einführendes Beispiel: Information Retrieval

Stellen Sie sich vor, Sie suchen im **Internet** Informationen über ein bestimmtes Thema.

Suchmaschinen bieten Volltextsuche, Metadatensuche und eine hierarchische Kategorisierung.

Dokumente über ein sehr **spezielles Thema**, wie z.B. „Fernsehen“, sind nur sehr schwer exakt auszumachen. Der Begriff „Fernsehen“ hat auch mehrere Bedeutungen und steht in einem gewissen **Kontext**. Meine ich „Fernsehgeräte“, das „Fernsehprogramm“ oder das Medium „Fernsehen“?

Die Volltextsuche liefert zu viel, die Metadatensuche ist ungenau und nicht verläßlich, die Kategorisierung ist oft zu oberflächlich und semantisch nicht eindeutig.

Nicht nur im Internet stellt sich diese Problematik, auch in Unternehmen.

Die Grundlage jeglicher Dokumentation ist zum Beispiel Lotus Notes.

Ausgangsbasis

Lotus Notes ist

- ✓ ein umfangreiches Dokumentenhaltungs- und Groupwaresystem
- ✓ unternehmensweit einsetzbar
- ✓ in das vorhandene Intranet integrierbar
- ✓ über WEB-Browser benutzbar.

Lotus Notes hält die gesamte Information in **strukturierten Dokumenten**, die sich in **Dokumentendatenbanken** befinden.

Hohes Datenaufkommen bedingt das Problem der Wiederauffindbarkeit.

Knowledger bietet weitergehende Hilfsmittel.

Ausgangsbasis (Fortsetzung)

Das Produkt **Knowledger** ist

- ✓ eine spezialisierte Lotus Notes - Anwendung der Firma ***Knowledge Associates***
- ✓ eine Reihe spezieller Datenbankschablonen, die der Haltung verschiedenster Arten von Wissensdokumenten dienen
- ✓ optimiert für die Benutzung über WEB-Browser.

Knowledger bietet Unterstützung bei der Bestimmung der Dokument-**Metadaten** und der **Kategorie-Informationen**.

Knowledger´s Ansätze sind konventionell und einfach.

Knowledger ist ein einfaches Wissensmanagement-System.

Wissensmanagement unter Lotus Notes / Knowledger

Lotus Notes bietet

- ✓ eine automatische Pflege der Standard-Metadaten
- ✓ grundsätzliche Möglichkeiten zur Definition spezifischer Ansichten auf Dokumentlisten (Views)
- ✓ eine Suchmöglichkeit über Volltextsuche
  (auch in File-Attachments)

Knowledger bietet

- ✓ verschiedene, thematisch getrennte Datenbanken
- ✓ Anwendergruppenorientierte Navigatoren (getrennt für Administratoren, Manager und normale Mitarbeiter)
- ✓ im Dokument abgelegte, frei editierbare Zusatzfelder zur inhaltlichen Kurzbeschreibung

Eine effiziente inhaltsbezogene Suche bedarf ausgeklügelteren Ansätzen.

Eine effiziente Recherche bedingt die Schaffung gewisser Voraussetzungen.

Grundbefürfnisse effizienterer Lösungen

- ✓ Die Beschränkung auf ein **kontrolliertes Vokabular** beim Kategorisieren und Suchen vermeidet Inkonsistenzen und erhöht die Trefferquote.
- ✓ Eine **Projektion des Dokumentinhalts in das Wissensmodell des Unternehmens** läßt sich formalisieren und durch Computerverarbeitung unterstützen.
- ✓ **Ansprechendere Benutzerschnittstellen** fördern die Motivation der Mitarbeiter.

Ein Lösungsansatz ist die Verwendung graphischer Wissensmodelle.

Streng hierarchische Begriffsmodelle bilden einfache graphische Ontologien.

Einfache ontologische Modelle

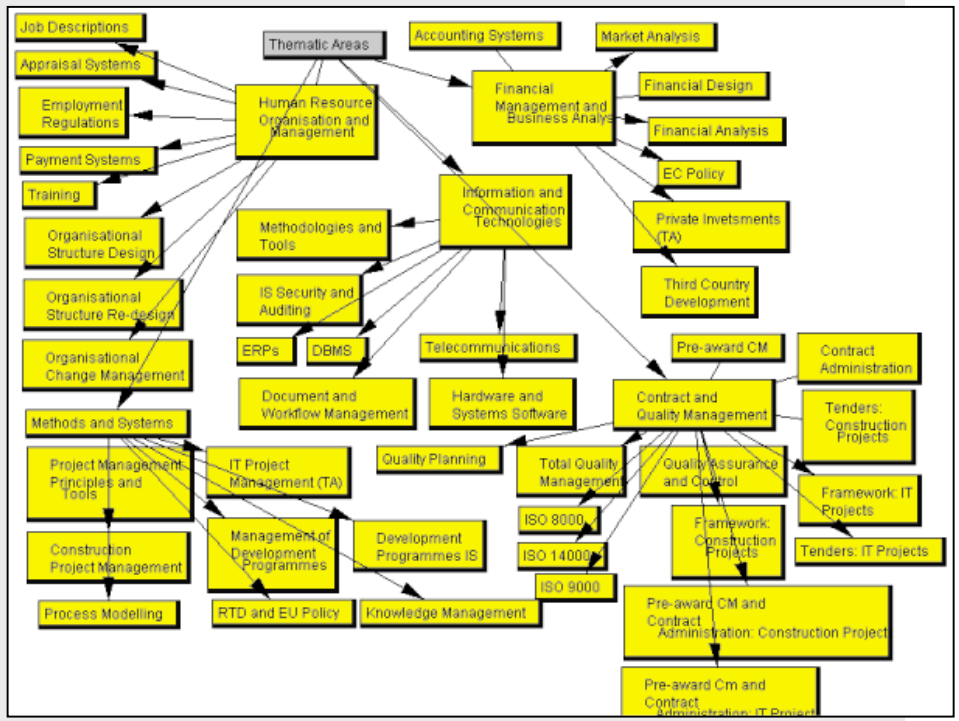
- ✓ Eine abgegrenzte **Wissensdomäne** läßt sich formal und abstrakt durch eine **Ontologie** modellieren.
- ✓ Hierzu überlegt man sich relevante **Konzepte** und **Begriffe**, die eindeutig sind, und die man miteinander in **Beziehung** setzt.
- ✓ Die einfachsten Strukturen sind **Hierarchien**. In vielen Fällen genügt eine solche Struktur.
- ✓ Dokumente werden durch eine bestimmte Auswahl aus der Begriffsmenge indiziert und sind somit in das Wissensmodell des Unternehmens projiziert.

Hierarchische Begriffsmodelle werden jedoch sehr schnell unübersichtlich.

Zu komplexe Begriffshierarchien sollten in Teilmodelle unterteilt werden.

Komplexe Begriffsmodelle

- ✓ Komplexe Modelle sind unter Umständen notwendig, ihre **Unübersichtlichkeit** kann jedoch gebrochen werden.
- ✓ Abhilfe bieten **hypertextmäßig verschachtelte Teilmodelle**.
- ✓ Ein solches Teilmodell wird dann nicht zusammen mit dem übergeordneten Modell dargestellt.



Beispiel einer komplexen Ontologie ohne Schachtelung

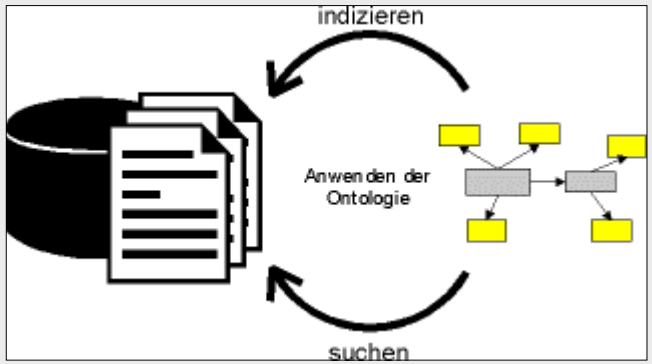
Die Umsetzung des geschachtelten Modellansatzes erfordert neue Software.

Zwei neue Softwarekomponenten unterstützen diese Verfahrensweise.

Erforderliche Softwarekomponenten

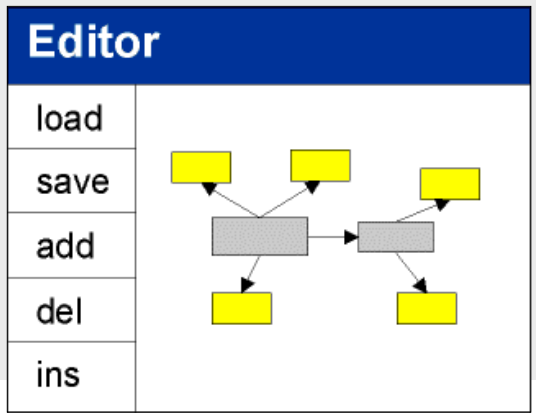
- ✓ *Eine* Benutzerschnittstelle zur ontologiebasierten Indizierung und Suche bietet guten Bedienkomfort:

  das **Index-Retrieval-Interface** (IRI)



- ✓ Ein graphischer Editor für den komfortablen Entwurf der Ontologien:

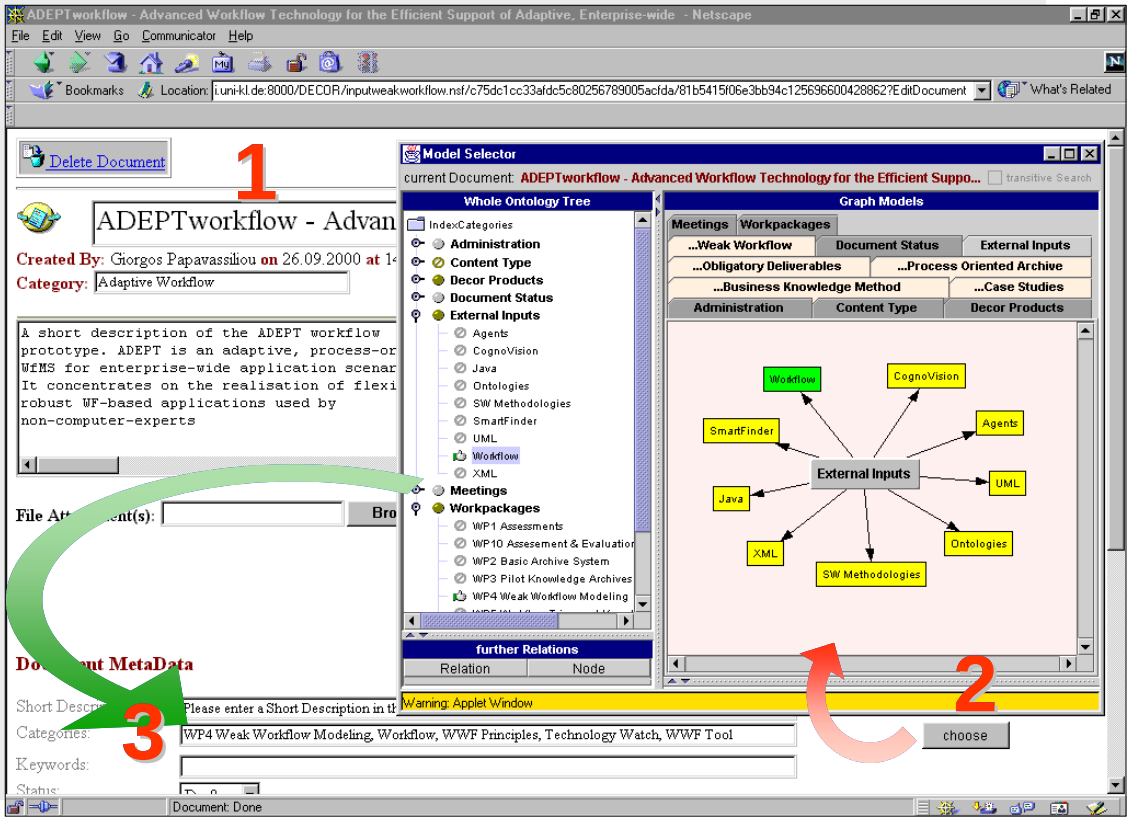
  der **Ontology Editor**



Beiden Komponenten arbeiten mit der selben zentral gehaltenen Ontologie.

Das Index-Retrieval-Interface: Indizieren

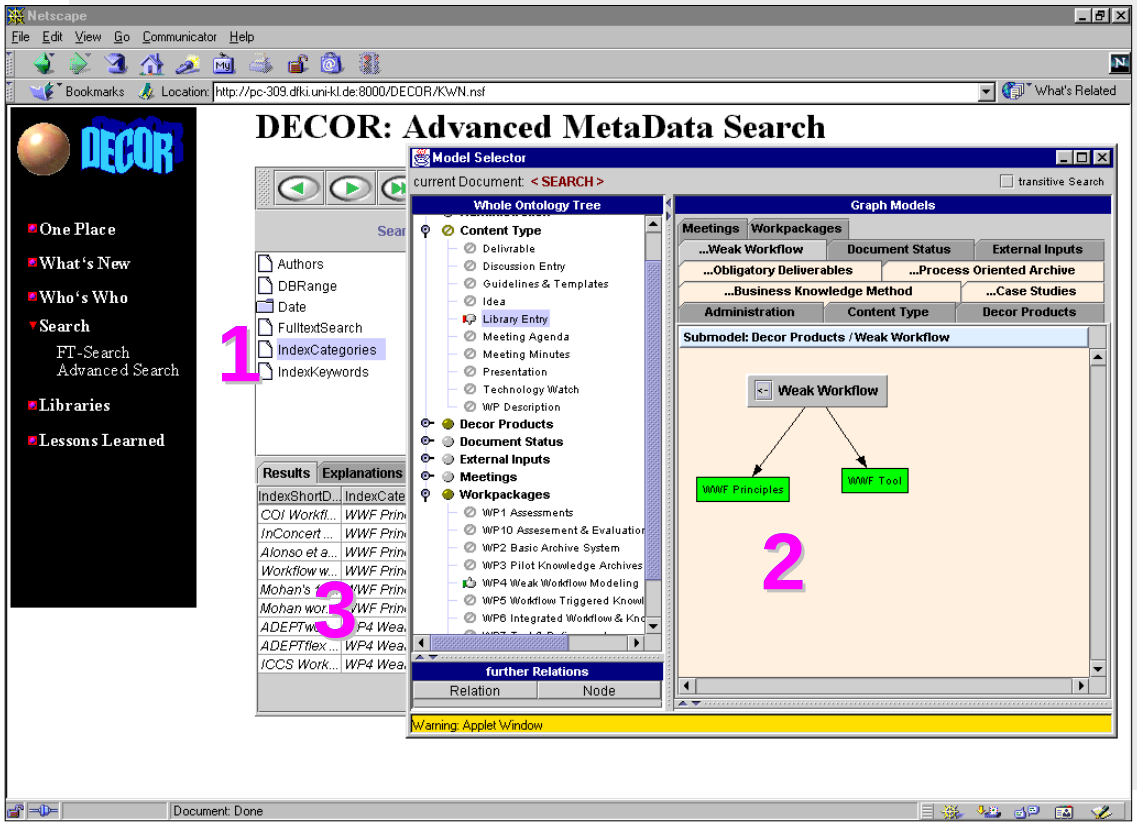
1. Ein Lotus Notes - Dokument wird ganz normal im WEB-Browser präsentiert.
2. Nach Betätigung eines Buttons erfolgt die **Indizierung** über die graphische Selektion aus der durch einen JAVA-Frame dargestellten Ontologie.
3. Die Kategorisierungsinformationen werden direkt in ein spezielles Feld übertragen.



Das selbe Interface dient der Suche über die graphischen Modelle.

Im Suchmodus findet eine Kommunikation mit dem früheren Suchapplet statt.

Das Index-Retrieval-Interface: Suchen

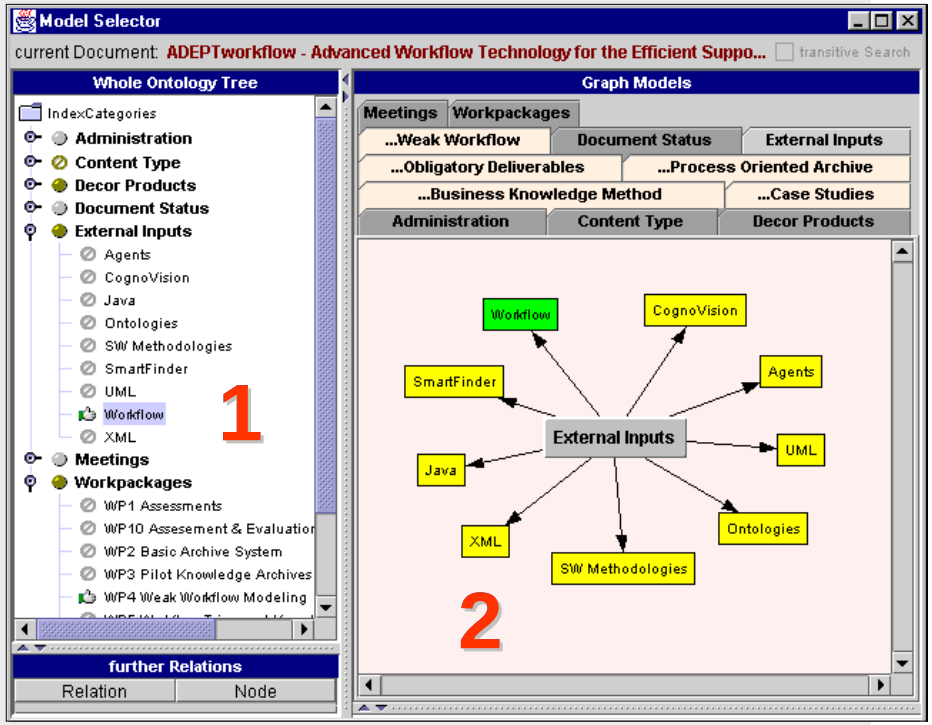


1. Nach Selektion des Vaterkonzepts für die graphischen Modelle im Suchapplet befindet sich das IRI im Suchmodus.
2. Jede einzelne Knoten-Selektion führt zu einer neuen Abfrage.
3. Das Abfrageergebnis wird sofort als Liste präsentiert, aus der einzelne Dokumente geöffnet werden können.

Zu jeder Zeit existiert höchstens eine IRI-Instanz bzw. dessen JAVA-Frame.

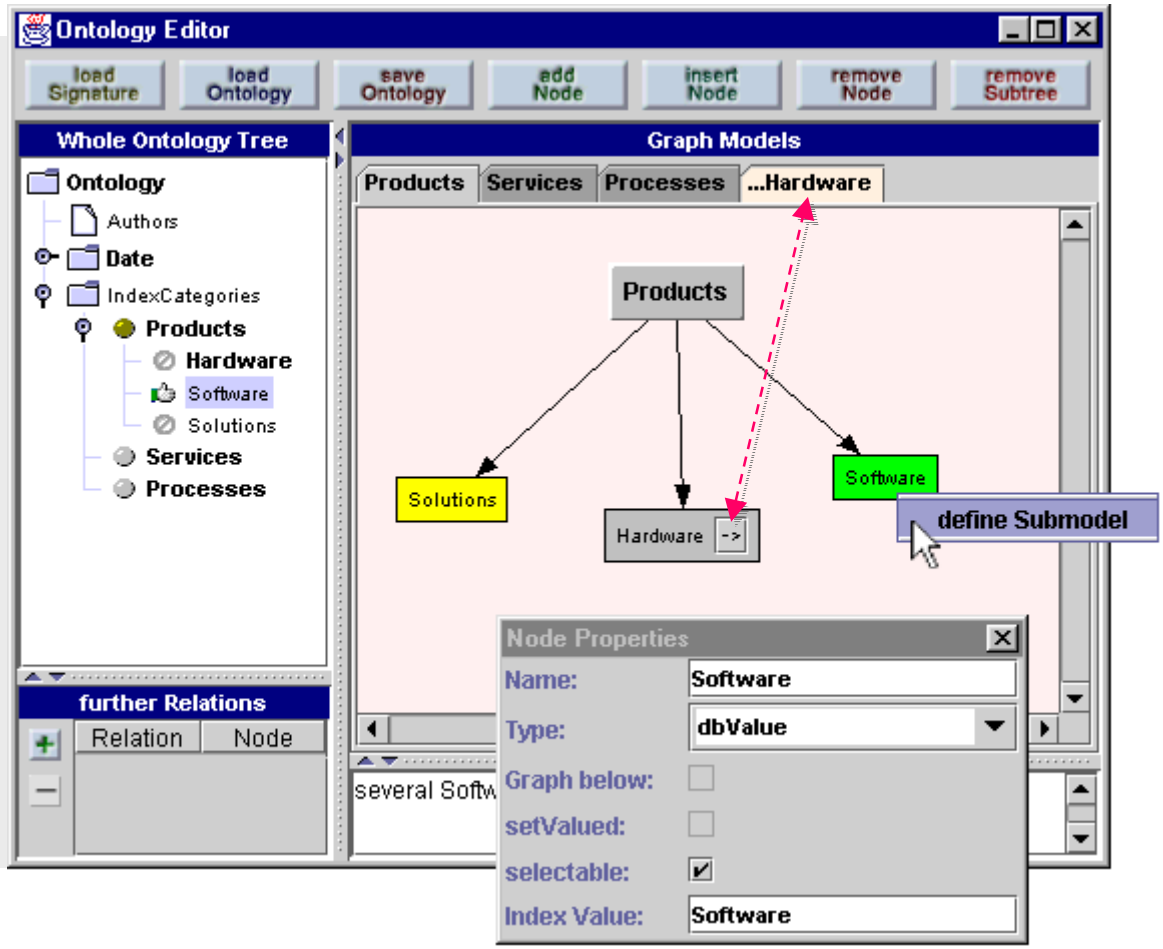
Merkmale des Index-Retrieval-Interface im Überblick

- ✓ reine Java-Komponente (in **Knowledger** eingebettet)
- ✓ komfortable Oberfläche durch Verwendung der JAVA-Klassenbibliothek **SWING**
- ✓ nahtloser Übergang zwischen Such- und Einstell-Anwendung: automatischer **Moduswechsel**
- ✓ einfache Navigation durch komplexe Ontologien mittels einer synchronisierten Baum- und Graph-Darstellung (1, 2)



Der Ontology Editor

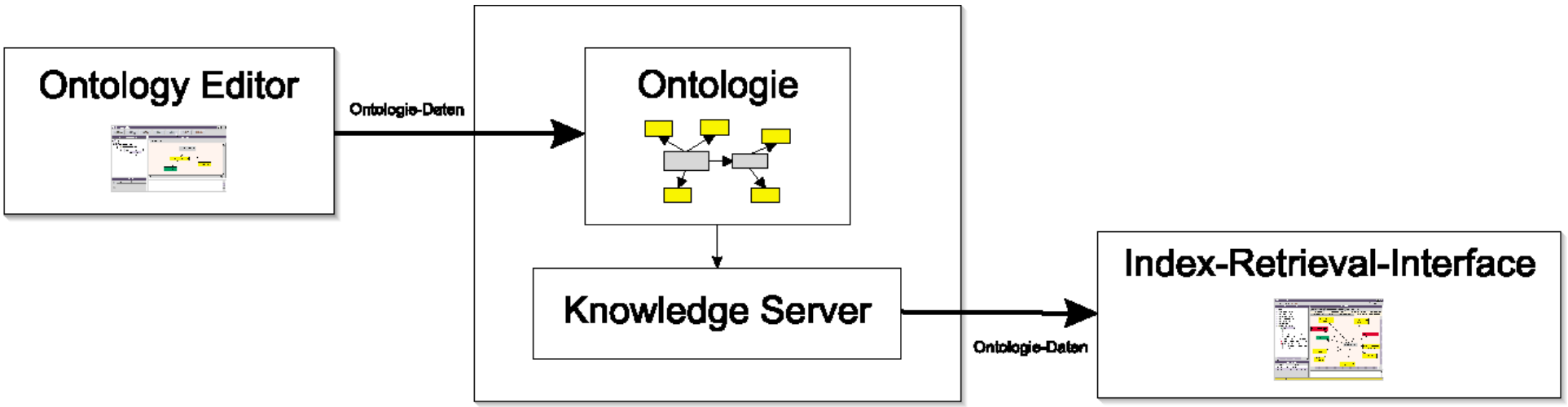
- Der Editor erzeugt graphisch **verschachtelte Teilmodelle**, die über Linkbuttons miteinander verbunden sind.
- Die Darstellung und Interaktion ist der des **IRI** identisch.
- Er bietet die Möglichkeit der Definition beliebiger **Querbeziehungen** und erklärender **Knotenkommentare**.



Die Editor-Anwendung läßt sich auf jedem JAVA-fähigen Rechner starten.

Der Ontologiefluß

Eine im Editor erstellte Ontologie wird dem **Knowledge Server** übergeben, so daß er das Index-Retrieval-Interface mit den Modell-daten beliefern kann.



Das Konzept der Knowledge Servers stammt aus dem KnowNet-Projekt.

Ausblick: Ontologiemodifikationen

- ✓ Im Editor umgesetzt ist bereits die **Protokollierung** der getätigten Änderungen.
  Nach der Speicherung einer geänderten Ontologie wird eine Änderungs-protokoll-Datei angelegt.
- ✓ Die Datei liefert einen Ansatzpunkt zur weiteren **Verarbeitung**, wie
  - automatische Unterrichtung der Autoren über die Ontologieänderungen
  - „intelligente“ Anpassung der Kategorisierungsinformationen der betroffenen Dokumente an die neue Situation

Solche automatischen Reaktionen bedürfen weiterer konzeptionelle r Überlegungen.

Ausblick: Beziehungssemantiken

Beispiel:

In einer IT-Beraterfirma existiert ein Teilmodell, das eine Beziehungsstruktur verschiedener Softwareprodukte bestimmt. Man könnte sich eine Beziehungsart „isIncompatible“ vorstellen, die beispielsweise zwischen dem Datenbankprodukt „Informix“ und der Betriebssystemgruppe „MS Windows“ definiert ist.

Solche beliebige Beziehungen können im Editor zwar formal definiert werden, es fehlt jedoch noch eine entsprechende Verarbeitung einer zu definierenden **Semantik** der Beziehungen.

So könnte eine Dokument-Abfrage, die aus der Selektion beider Produkte besteht, im Vorfeld abgewehrt werden mit dem Kommentar der Inkompatibilität.

Die Einführung der Beziehungssemantiken ist für eine intelligente Suche unabdingbar.

Die wesentlichen Aspekte dieser Arbeit zusammengefaßt...

Essenz

- ✓ Umsetzung des Prinzips der **hypertextmäßig verschachtelten Teilmodelle**
- ✓ **Vereinheitlichte Oberfläche** in allen drei Anwendungen (Indizieren, Suchen, Editieren)
- ✓ generell vereinfachte Bedienung
- ✓ Einbettung der neuen Komponenten in das vorhandene Programmsystem von **KnowNet**