Übung Netzwerke und Agenten:

Projekt-Themen

Projekt 1: Ereignis-Filterung (2 Personen)

Ein Agent erhält über Konversationen Ereignisse zugestellt. Anhand einer vom Benutzer frei konfigurierbaren Menge von Regeln (modelliert in XML, um leichter übertragen werden zu können) werden diese gefiltert und entsprechende Aktionen gesetzt (welche ebenfalls frei definiert werden können: eine kleine Auswahl ist zu programmieren). Die eigentliche Verarbeitung erfolgt über ein Inferenzsystem (Jena von HP Labs). Die Regeln sind daher als XML einzulesen, in das entsprechende Format zu konvertieren und dem System einzuspeisen. Jedes Ereignis wird ebenfalls umgewandelt, eingespeist sowie die Folgen abgelesen und ausgeführt.

Projekt 2: Remote UI (2 Personen)

Für die neue WeLearn Version wird eine Agentensystem-Anbindung erstellt. Diese benötigt auch die Darstellung eines Userinterfaces für Agenten. Es ist daher ein Transferformat bzw. –konzept zu erstellen und zu implementieren, wie ein Agent ein graphisches Userinterface über WeLearn anzeigen kann. Dort wird die Millstone Bibliothek (http://www.millstone.org/) verwendet, d. h. die Darstellungsanweisungen des Agenten müssen in entsprechende Objekte umgebaut und die Eingaben wieder zurück zum Agenten transferiert werden. Dies kann nicht direkt als Millstone- Objekte erfolgen (diese sind nicht serialisierbar; nochmals genau überprüfen!). Daher muß über XForms (http://www.w3c.org/MarkUp/Forms/) gearbeitet werden. Hierzu ist nach existierende Möglichkeiten und Bibliotheken zu suchen und diese darzustellen. Ein Objektmodell ist zu schaffen (bzw. kurz zu dokumentieren bei Verwendung einer Bibliothek), mit dem programmatisch Formulare zusammengestellt werden können. Diese sind als Datei zu speichern (Teil 1). Anhand dieser Datei erzeugt der Teil 2 ein Userinterface mittels Millstone, das genau diese Element darstellt (d. h. ein XForms User Interface realisiert in Millstone mittels eines Servlets). Das Ergebnis (XML) ist wieder in eine Datei zuschreiben. Diese liest Teil 1 ein und integriert die Ergebnisse zurück in die Objekte. Die einzelnen Aktionen (speichern in Datei, einlesen im Servlet, zurücklesen in die Objekte) werden jeweils händisch ausgelöst.

Wird keine passende Bibliothek gefunden, so ist nicht der gesamte XForms Standard zu implementieren. Die Funktionalität muß jedoch zumindest einem normalen HTML-Formular entsprechen (Textfelder, Labels, Checkboxen, Radioboxes, Listboxes, Buttons).

Projekt 3: Website Erstellung (1 Person)

Der Agent soll regelmäßig ein Verzeichnis darauf überprüfen, ob eine neue ZIP-Datei dort abgelegt wird (durch einen Timer und eine Liste der bekannten Dateien; beim Start des Agenten wird die Liste gefüllt, sodaß erst später hinzukommende Dateien bearbeitet werden). Ist dies der Fall, wird sie in ein konfigurierbares Verzeichnis (User-Interface) in ein neu erstelltes Unterverzeichnis entpackt. Anschließend wird eine erweiterbare (definierte Schnittstelle!) Folge von Aktionen darauf ausgeführt. Diese Liste soll auch dynamisch erweiterbar sein, indem ein neuer Filter zur Laufzeit hinzugefügt wird:

- Prüfen von HTML Seiten auf ungültige bzw. externe Links: Identifizieren aller Links (a href, img source, applet, link, ...), konstruieren kompletter Links (base Attribut falls vorhanden). Ist dies ein Weblink, so wird weiter geprüft ob er zugänglich ist. Handelt es sich um einen relativen Link, so muß er innerhalb des Ausgabeverzeichnisses bleiben und die referenzierte Datei muß auch tatsächlich vorhanden sein.
- Umwandlung von E-Mail Adressen in Bilder mit der entsprechenden Adresse: Die Webseiten sind nach E-Mail Adressen zu durchsuchen (wo immer diese auch im Text auftreten, jedoch nicht in E-Mail Links; solche sind als Fehler zu melden). Anschließend wird die E-Mail Adresse in ein Bild umgewandelt (GIF oder PNG). Hierzu können die Klassen von javax.imageio verwendet werden.
- Auf einer zentralen Link-Seite einen neuen Link zur neuen Website eintragen: Welche Seite dies ist, kann über das UI eingestellt werden. Darin befindet sich ein HTML Kommentar ("z. B. <!—NEXT_WEBSITE -->"), an dessen Stelle ein relativer Link zur neuen Website eingebaut wird. Als Text dient der Titel der Startseite (diese ist index.htm, index.html, default.htm oder default.html; ansonsten der Namen der ZIP-Datei).

Werden Fehler erkannt (z. B. fehlende verlinkte Datei), so wird eine entsprechende Meldung erzeugt und an eine Liste angehängt. Diese Liste kann über das Benutzerinterface angezeigt werden (Text, keine besondere Bearbeitung/Darstellung nötig). Jede Meldung muß zusätzliche Informationen enthalten, um den Fehler leicht lokalisieren zu können: Uhrzeit, welches ZIP-Paket, welche Datei, Stelle in der Datei, etc., je nach dem Problem.

Zu erstellende Struktur (Bsp.):

\Upload (dies ist das zu überwachende Verzeichnis)

\Upload\WS1.zip, \Upload\WS2.zip (zwei abgegebene Websites, gepackt in ZIP-Dateien)

\Home (das Zielverzeichnis)

\Home\WS1\..... (Wo die Webseiten von WS1.zip hin entpackt werden)

\Home\WS2\..... (Wo die Webseiten von WS2.zip hin entpackt werden)

\Home\default.htm (Die Datei mit der Liste der Sub-Websites)

Projekt 4: Personen mit ähnlichen Interessen finden (1 Person)

Für jede Person in einem Lernsystem existiert ein eigener Agent, welcher bestimmte Worte kennt, die diese Person interessieren ("Keywords", welche der Person zugeordnet sind). Es ist ein Agent zu erstellen, der über Konversationen diese Interessen aller existierenden Agenten (=Broadcast + Timeout) sammelt und herausfindet, welche Personen ähnliche Interessen besitzen, d. h. bei denen die Überschneidung der Keywords besonders groß ist. Für die eigentliche Berechnung soll eine frei verfügbare Bibliothek in Java gesucht sowie ein Überblick über diese erstellt werden. Zwischen beliebigen zwei Personen soll auch eine numerische "Distanz" berechnet werden können. Die Ergebnisse sollen über Konversationen abgerufen werden können.

Zwei Ergebnisse sollen zur Verfügung stehen:

- Für eine Person die Liste der Personen bestimmen, die "ähnlich" sind
- Für eine Person eine Grafik mit allen Personen erstellen (javax.imageio), wobei die Entfernung die Überschneidung repräsentiert. Hierzu ist auch eine Imagemap zu erstellen, sodaß auf jede Person geklickt werden kann. Im Agenten führt dies zu keiner Reaktion (diese ist nicht zu programmieren)

Projekt 5: Gewicht von Schlüsselwörtern berechnen (1 Person)

Für eine Person ist eine Liste mit Schlüsselwörtern zu erstellen. Es ist bekannt, welche Dokumente eine Person wann (inkl. Dauer) besucht hat. Für diese Dokumente stehen auch Schlüsselwörter zur Verfügung. Es ist nun nach einer Bibliothek zu suchen und ein Vergleich zu erstellen, anhand der bestimmt werden kann, welche Schlüsselwörter zu einer Person passen, d. h. woran sie interessiert ist. Die Besuche und die zugehörigen Schlüsselwörter werden aus einer Datei übernommen. Als Ergebnis ist die Liste der zugeordneten Schlüsselwörter und deren Gewicht (=Bedeutung) über ein UI darstellbar.