Teil III

Benutzungshandbuch Firmenagent

Inhaltsangabe Teil III

Kapitel 1

Einleitung

1.1 Aufgabenstellung und Motivation

Gegenstand der Projektgruppe 343 war die Implementation einer Bibliothek, mit deren Komponenten die Implementation eines Agenten, auch Softbot genannt, wesentlich vereinfacht werden sollte. Nachdem diese Bibliothek Namens BotIshelly fertiggestellt war, bestand die Aufgabe der Projektgruppe nun darin, an einem Beispiel zu zeigen, daß BotIshelly auch das Geforderte leistet. Die Projektgruppe beschloß, zur Demonstration der Bibliotheksfunktionalität einen Agenten mit BotIshelly zu implementieren, der Homepages von Firmen im WWW findet, und zwar ohne das übliche "Durchklicken" durch Unmengen von Suchmaschinenergebnissen und das Ausprobieren von ad hoc gebildeten URLs. Genau diese immer gleichen Vorgehensweisen können nämlich automatisiert werden und werden auch von dem Agenten angewandt.

Das Hauptziel bei der Entwicklung des hier beschrieben Agenten war somit nicht eine bestmögliche Performanz, sondern vielmehr war im zeitlich begrenzten Rahmen der Projektgruppe die Demonstration der Fuktionalität der Bibliothek BotIshelly das Ziel. Damit ist die korrekte Funktion der von BotIshelly bereitgestellten Einzelkomponenten, wie z. B. die der Planer, die der Klassifikatoren oder auch die der Datenbeschaffung gemeint, und vor allem auch das reibungslose Zusammenspiel dieser, wie es z. B. in den Klassen der Systemsteuerung nötig ist.

1.2 Realisierung

Realisiert wurde der Firmen–Homepage–Agent als Javaservlet, da in BotIshelly schon Unterstützung für diese Lösung enthalten ist. Außerdem ermöglicht ein Servlet, den Agenten auf einem schnellen Rechner mit schnellem und permanentem Netzzugang auszuführen und seine Bedienung bzw. Steuerung von einem anderen Rechner aus zu tätigen. Darüberhinaus kann ein Agent durch diese Architektur mehrere Anfragen gleichzeitig bearbeiten[1]. Die hauptsächliche Arbeit bei der Realisierung bestand darin, problemspezifische Operatoren zu implementieren. Vorher waren jedoch die Signaturen und damit deren jeweilige Funktionalität festzulegen. So gibt es z. B. einen Operator, der aus der URL einer HTLM–Seite, die für eine Seite über ein bestimmtes Produkt gehalten wird, eine neue URL bildet, für die er dann testet, ob es sich um eine Firmenhomepage handelt. Da die Pläne des Agenten maßgeblich von den Operatorensignaturen bestimmt werden, ist hier sehr viel Abstimmungsarbeit zu leisten. Die Performanz des Agenten steht und fällt mit der Menge der Operatoren, für die man sich entscheidet. Bei der Entwicklung dieses Agenten wurde die anfänglich angedachte Menge an relativ einfachen Operatoren durch eine kleinere Menge

[1] wobei die Systemressourcen dem Parallelitätsgrad schnell ein jähes Ende setzen

komplexerer Operatoren ersetzt, da sonst die Anzahl der möglichen Pläne “auszuufern” drohte. Das Potential des Agenten ist an dierser Stelle sicher nicht (voll) ausgeschöpft worden. Eine genauere Beschreibung der realisierten Operatoren folgt später im Text. Nun soll jedoch beschrieben werden, wie der Agent benutzt werden kann.

Kapitel 2

Benutzung des Agenten

2.1 Auf die Plätze, ...

In diesem Abschnitt wird erklärt, was alles benötigt wird, um den Firmen–Homepage–Agenten "laufen" zu lassen.

Zunächst wird die Bibliothek BotIshelly benötigt, welche wiederum das JDK 1.2.1 (Solaris 2) benutzt. Daneben werden noch die agentenspezifischen Javaklassen benötigt. Da der Firmen–Homepage–Agent als Javaservlet implementiert ist, wird ein Programm benötigt, welches ein solches ausführen kann. Getestet wurde der Agent mit dem Servletrunner des JSDK der Version 2.0 (Solaris 2). Nicht zu vergessen sind die von BotIshelly benötigten, aber nicht enthaltenen Javaklassen bzw. –bibliotheken[1]. Schließlich ist noch ein Browser, z. B. Netscapes Communicator, mit dem auch getestet wurde, nötig, um den Agenten zu steuern.

2.2 ... fertig, ...

In diesem Abschnitt wird gezeigt, wie die im vorigen Abschnitt beschrieben "Einzelteile" benutzt werden, um den Firmen–Homepage–Agenten zu starten.

Im Agentenverzeichnis befindet sich u. a. die Datei `config.sys` (s. II D.1), in der Einstellungen für das Verhalten des Agenten vorgenommen werden. Hier müssen Sie mindestens den Pfad des Arbeitsverzeichnisses `Working Directory` auf das Agentenverzeichnis setzen. Desweiteren müssen Sie den Pfad `ClassDatabase` auf die gewünschte Klassifizierer-Datenbank setzen. Eine Klassifizierer Datenbank für den Firmen-Suchagenten finden Sie in `share/FirmenSuchagent/FirmenSuchagent.cldb`.

Wird der Servletrunner des JSDK 2.0 benutzt, müssen weitere Einstellungen in der Datei *Agentenverzeichnis*`/io/servlet.properties` vorgenommen werden. Die Javaklasse, die der Java–Servlet–Runner ausführen muß, ist `/io/InputmaskServlet` im Agentenverzeichnis. Dies entspricht dem Wert der Variable `servlet.agent.code` in `/io/servlet.properties`. Der Servletrunner muß dem Konstruktor dieser Klasse beim Starten das Argument `configfile=`*Agentenverzeichnis*`/config.sys` übergeben (Variable `servlet.agent.initArgs`).

Als letztes müssen Sie noch die Umgebungsvariable `CLASSPATH`, die dem Javainterpreter angibt, wo er die benötigten Klassen suchen soll, richtig setzen. U. a. müssen sich das Verzeichnis, in dem BotiShelly installiert ist, das Agentenverzeichnis, das JDK und das Verzeichnis des Servletrunner (JSDK) im Classpath befinden. **Dabei muß das Agenten- vor dem BotIshelly–Verzeichnis stehen.**

[1]siehe dazu Teil II, BotIshelly–Handbuch

Gestartet wird der Servletrunner mit `servletrunner -d` *Agentenverzeichnis*`/io` aus dem Verzeichnis *Agentenverzeichnis*`/io`. Sie können zum Starten auch den Befehl **`make newrun`**, ausgeführt im Agentenverzeichnis, benutzen.

2.3 ... los!

In diesem Abschnitt wird schließlich anhand eines Beispiels beschrieben, wie mit dem laufenden Agenten die Homepage einer bestimmten Firma gesucht — und hoffentlich auch gefunden — wird.

Nachdem das Agentenservlet mit Hilfe des Servletrunners wie oben beschrieben gestartet wurde, kann nun mit einem Browser die Eingabeseite des Agenten geladen werden. Deren URL wird typischerweise durch Konkatenation

- des Protokolls (“`http:`”),
- des Rechnernames (hier “`//herz`”),
- des Ports, über den der Servletrunner zu erreichen ist, (hier “`:8080`”) und
- des Identifikator des Agenten im Servletrunner (hier “`/servlet/agent`”) gebildet.

Nachdem der Browser die Seite mit der Eingabemaske geladen hat, kann durch die Eingabe bestimmter Daten der gesuchten Firma die Suche gestartet werden. Zur Veranschaulichung siehe den Screenshot der Eingabemaske.

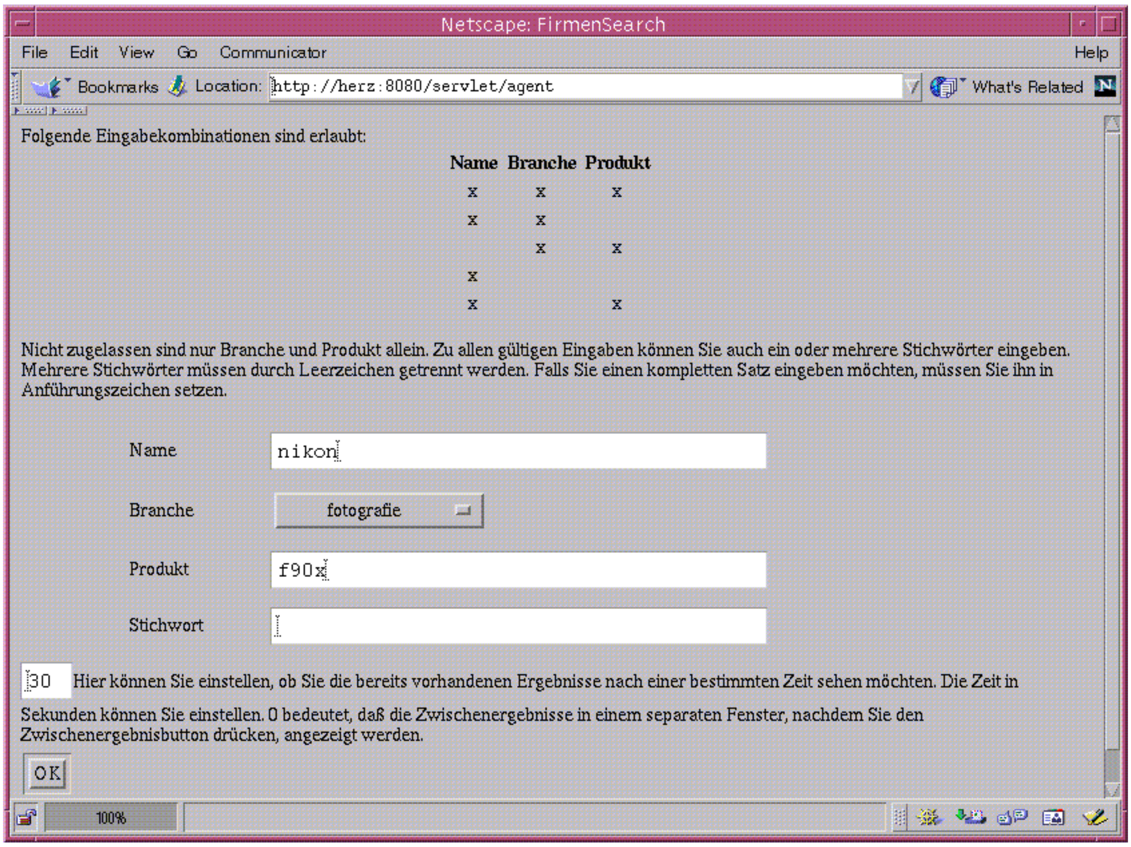

Abbildung 2.1: Eingabemaske des Firmen–Homepage–Agenten

Im oberen Teil der Eingabemaske werden die zulässigen Kombinationen an Firmeninformationen (Name, Branche, Produkt) beschrieben, für die eine Suche möglich ist. Die Branche kann dabei aus einer Liste ausgewählt werden, wobei auch die Wahl "alle Brachen" möglich ist. Name und Produkt der Firma sind Freitexteingaben. Wird ein Produkt angegeben, so werden nur dann Firmen–Home–Pages ausgegeben, wenn eine passende Produktseite gefunden wird. Im allgemeinen wird entweder versucht, von einer Firmenseite durch "Spidern" eine passende Produktseite zu finden, oder aber ausgehend von einer Produktseite, z. B. durch das Folgen von "Home–Links" oder das "Abschneiden der URL", die passende Firmenseite. Die jeweiligen "Startseiten" werden dabei durch das Befragen von Suchmaschinen gefunden. Wird eine Branche angegeben, so wird bei der Suchmaschine *Yahoo!* in der entsprechenden Kategorie gesucht und bei anderen Suchmaschinen die Anfrage durch Synonyme ergänzt. Wahlweise können zusätzlich Stichworte angegeben werden, die die Suche dadurch einschränken, daß nur Firmen–Home–Pages gesucht werden, auf denen diese Worte vorkommen.

Schließlich kann noch eingegeben werden, in welchem Intervall die Ergebnisausgabe aktualisiert werden soll. Nachdem die Daten der Suche eingegeben wurden, wird die Suche durch klicken des OK–Buttons gestartet. Dadurch wird der Browser auch veranlaßt, die Seite, auf der die Zwischenergebnisse angezeigt werden, zu laden. Ein Beispiel ist in Form eines weiteren Screenshots gegeben.

Abbildung 2.2: Zwischenergebnismaske des Firmen–Homepage–Agenten

Diese Zwischenergebnisseite wird nun periodisch in dem zuvor angegebenem Intervall aktualisiert. Um den angegeben Links zu folgen, ist es günstig, die bei den meisten Browsern vorhandene Funktion "Öffne Link in neuem Fenster" zu verwenden. Um die Suche vorzeitig zu beenden, kann — und sollte — der ABORT–Button der Zwischenergebnismaske verwendet werden. Anderenfalls wird das Ende der Suche angezeigt werden.

Kapitel 3

Hinter den Kulissen

3.1 Die Ziele des Planers

Die Pläne, die der Firmenagent verfolgt, bestehen aus Operatoren, die von den Planern in eine sinnvolle Reihenfolge gebracht wurden. „Sinnvoll“ heißt dabei, daß nach Abarbeitung der Folge von Operatoren (also des Plans) durch den Planausführer die Zielprädikate erfüllt sein sollen. Um das Ziel, eine Homepage einer Firma zu finden, erreichen zu können, muß der Plan eine Teilmenge der folgenden Prädikate (vgl. I 5) erfüllen:

- `has_compage (companyname, compage)`
- `contains_keyword (compage, keyword)`
- `contains_productname (prodpage, prodname)`
- `has_productpage (companyname, prodpage)`

Um die Suche zu starten, haben wir fünf mögliche Eingabekombinationen von Firmenname, Branche und Produktname zugelassen, die optional durch ein Stichwort ergänzt werden können. Die erlaubten Kombinationen sind:

- Name, Branche, Produkt
- Name, Branche
- Branche, Produkt
- Name
- Name, Produkt

Beispiel: Wird als Eingabe die Kombination Name = *nikon*, Branche = *fotografie* und Produkt = *f90x* gewählt, so ist das Ziel, die Homepage der in der Fotobranche tätigen und das Produkt f90x vertreibenden Firma Nikon zu finden. Daraus folgt, daß drei der oben genannten Zielprädikate erfüllt sein müssen, nämlich

- `has_compage ("nikon", compage)`
- `contains_productname (prodpage, "f90x")`
- `has_productpage ("nikon", prodpage)`

3.2 Die Operatoren als Trittsteine zum Ziel

Von einer gegebenen Anzahl von Prädikaten aus muß der Planer die Operatoren geschickt hintereinandersetzen, damit, ein Operator nach dem anderen, das Ziel erreicht werden kann. Bei der Umsetzung des Firmenagenten haben wir uns für Operatoren entschieden, die sich in fünf Kategorien einteilen lassen:

1. Operatoren, die sich durch Suchmaschinenanfragen eine Menge von URLs besorgen
2. Operatoren, die von einer einzelnen Adresse aus eine beschränkte Tiefensuche durchführen (Spider–Operatoren)
3. Operatoren, die aus einer vorhandenen URL eine weitere raten
4. Operatoren, die aus einer vorhandenen Seite Informationen zu extrahieren suchen
5. Operatoren, die das Vorkommen eines Wortes auf einer Seite abtesten

Im Folgenden werden die zu den einzelnen Kategorien gehörenden Operatoren kurz vorgestellt. Dabei bezeichne **V** eine Vorbedingung, **N** die erfolgreiche Nachbedingung (unsere Operatoren haben nur eine Nachbedingung) und ein + vor einem Prädikat– oder Prädikatargumentnamen ein Erzeugen in der Add–Liste der Nachbedingung.

3.2.1 Operatoren zur Suchmaschinenanfrage

Diese Operatoren haben gemeinsam, daß sie Suchmaschinen abfragen und bei Erfolg Webadressen zurückliefern, die von den Operatoren abhängende Eigenschaften haben sollen. Im Firmenagenten gibt es drei Operatoren dieser Kategorie, die jeweils für die Suchmaschinen Altavista, Nathan und Yahoo ausprogrammiert sind.

`FindComPageOperator`

V: `in_trade (comname, trade)`
`relates_to (synonym, trade)`
N: `+ has_compage (comname, + compage)`

Aufgabe: Anfrage an eine Suchmaschine mit Hilfe von Firmennamen, Branche und zur Branche passendem Stichwort. Die URLs, die die Operatoren liefern, sollen Firmenseiten sein.

`FindContainsProductOperator`

V: `relates_to (synonym, trade)`
`productname (prodname)`
N: `+ contains_productname (+ productpage, prodname)`

Aufgabe: Ähnlich dem vorangegangenen Operator, nur daß hier nach einer Produktseite gesucht wird. Dabei ist der Firmenname nicht bekannt.

`FindProdPageOperator`

V: `in_trade (comname, trade)`
`deals_in (comname, prodname)`
`relates_to (synonym, trade)`
N: `+ has_productpage (comname, + prodpage)`
`+ contains_productname (prodpage, prodname)`

Aufgabe: Sucht wie der vorherige Operator nach Produktseiten, wobei hier andere Vorbedingungsprädikate einfließen.

3.2.2 Spider–Operatoren

Spider–Operatoren führen von einer Webseite aus eine Tiefensuche durch. Dabei soll eine Unterseite gefunden werden, die gewisse Eigenschaften aufweist. Da diese Suche unter Umständen lange dauern kann, sollte man die Suchtiefe beschränken.

`FindProdPageFromHomeOperator`

V: `has_compage (comname, compage)`
`productname (prodname)`
N: `+ has_productpage (comname, + prodpage)`
`+ contains_productname (prodpage, prodname)`

Aufgabe: Von einer Firmenseite aus wird nach einer Produktseite gesucht, die einen gegebenen Produktnamen enthalten soll. Dabei wird nur auf dem gegebenen Server gesucht, bis zu zwei Linkebenen in die Tiefe.

3.2.3 Aus URLs URLs ratende Operatoren

Häufig kann aus einer gegebenen URL eine andere URL geraten werden. So kann mit ziemlicher Sicherheit aus einer Adresse http://www.bla.de/verkauf/zitronen.html geraten werden, daß die Homepage unter http://www.bla.de zu finden ist. Diese Aufgabe haben die folgenden Operatoren:

`GuessCompageOperator`

V: `in_trade (comname, trade)`
N: `+ has_compage (comname, + compage)`

Aufgabe: Es wird versucht, die Homepage einer Firma zu raten. Die vier programmierten Operatoren raten die Adressen *www.Bla.de*, *www.Bla.com*, *www.Blacom.de* und *www.Bla-com.de* zu einem Firmennamen.

`GuessCompageFromProdpageOperator`

V: `trade (trade)`
`has_productpage (comname, prodpage)`
N: `+ has_compage (comname, + compage)`
`+ in_trade (comname, trade)`

Aufgabe: Die beiden Operatoren `GuessCompageFromProdpageCutOperator` und `GuessCompage-FromProdpageHomeOperator` versuchen beide, die Homepage einer Firma zu finden. Der erste Operator schneidet einfach die Host–URL aus der Adresse der Produktseite heraus, während der zweite nach einem Link auf die Home–Seite innerhalb des HTML–Kodes sucht.

3.2.4 Aus Seiten extrahierende Operatoren

Diese Kategorie Operatoren soll auf einer Seite nach Informationen suchen, z. B. dem (noch nicht bekannten) Namen einer Firma.

`GuessCompanynameOperator`

V: `contains_productname (prodpage, prodname)`
N: `+ has_productpage (+ comname, prodpage)`
`+ deals_in (comname, prodname)`

Aufgabe: Dieser Operator versucht, aus der übergebenen Produktseite den Namen der herstellenden Firma zu gewinnen. Die ausprogrammierte Strategie ist eher simpel und beschränkt sich darauf, den Namen der Firma aus der URL der Produktseite herauszuraten und diesen Namen zur Bestätigung auf der Seite zu suchen.

3.2.5 Überprüfende Operatoren

Überprüfende Operatoren durchsuchen eine Seite nach einem bestimmten Wort. Je nach Ausprogrammierung des Operators kann diese Suche weit über einen reinen Stringvergleich hinausgehen. So könnte der Satz „Bernd aß Bananen“ durchaus als das Verb „essen“ enthaltend klassifiziert werden.

`CompageContainsKeywordOperator`

V: `keyword (key)`
`has_compage (comname, compage)`
N: `+ contains_keyword (compage, key)`

Aufgabe: Der für den Firmenagenten programmierte Operator überprüft, ob das übergebene Wort als String irgendwo auf der Seite vorkommt. Groß–/Kleinschreibung wird ignoriert.

3.3 Funktioniert das?

Abschließend sei hier gezeigt, daß mit den von uns gewählten Operatoren tatsächlich Ziele erreicht werden können. Dazu führen wir hier das Beispiel aus III 3.1 weiter fort. Gegeben haben wir eine mit Synonymen zu den einzelnen Branchen gefüllte A–Box sowie eine Reihe von instantiierten oder halbinstantiierten Prädikaten. Dazu zählen natürlich die Zielprädikate

```
has_compage ("nikon", compage)
contains_productname (prodpage, "f90x")
has_productpage ("nikon", prodpage)
```

Wenn Sie sich die Operatorsignaturen angucken, dann wird klar, daß wir mit einem der `FindProdPage`–Operatoren weitermachen können, der nach Instantiierung und Ausführung vielleicht so aussehen könnte:

`FindProdPageOperator`

V: `in_trade ("nikon", "fotografie")`

```
    deals_in ("nikon", "f90x")
    relates_to ("fotografen", "fotografie")
N: + has_productpage ("nikon", + "www.nikon.de/Produkte/")
   + contains_productname ("www.nikon.de/Produkte/", "f90x")
```

Damit sind zwei der drei Zielprädikate schon erfüllt! Für das letzte zu erfüllende Zielprädikat fällt die Wahl auch nicht weiter schwer:

GuessCompageFromProdpageOperator

```
V: trade ("fotografie")
   has_productpage ("nikon", "www.nikon.de/Produkte/")
N: + has_compage ("nikon", + "www.nikon.de")
   + in_trade ("nikon", "fotografie")
```

Daß hier das `in_trade`–Prädikat noch einmal erzeugt wird, ist nicht schlimm, da es schließlich immer noch Gültigkeit besitzt.

Durch Anwendung dieser beiden Operatoren haben wir alle Zielprädikate vollständig instantiieren können und damit das Gesamtziel erreicht. Natürlich sind noch vielfältige andere Kombinationsmöglichkeiten denkbar, die häufig auch durchgespielt werden, da die Operatoren bei der Ausführung ja auch scheitern können.

Vielleicht haben Sie ja jetzt Lust bekommen, ein wenig mit den Operatoren herumzuexperimentieren und sie sogar zu verbessern. Wir wünschen Ihnen auf jeden Fall viel Spaß dabei!