Abschlussbericht

Studienprojekt:

InfoLine

Lehrstuhl für Computerlinguistik
Ruprecht-Karls-Universität Heidelberg

Projektgruppe: **Marit Rautso, Margeth Sopp, Iryna Zhmaka**

5.11.2003

Was ist InfoLine?

- InfoLine ist ein Prototyp eines telefonischen Auskunftssystems für das Studentensekretariat der Universität Heidelberg

- Es ermöglicht dem Benutzer (Anrufenden, vor allem Studierenden der Uni Heidelberg), Informationen zu verschiedenen Themen, die mit dem Studium zusammenhängen, zu bekommen

Was ist InfoLine?

- Die Bedienung des Systems erfolgt über natürlichsprachliche Eingaben des Anrufenden mit Hilfe eines Spracherkenners, die Ausgabe erfolgt über Text-to-Speech Synthese

Ziele des Projekts (1)

1. benutzerfreundliche Gestaltung des Gesprächs

   (Testen durch verschiedene Personen hat gezeigt, daß die Gesprächsgestaltung nicht immer als angenehm empfunden wird)
2. Ermöglichung freier Formulierung von Antworten durch den Anrufer (zum Teil frei)
3. Trennung von statischen Daten und oft geänderten Daten bei der Generierung von Systemantworten

Ziele des Projekts (2)

4. direktes Weiterverbinden mit entsprechenden Kontaktpersonen
5. Zuschicken von angeforderten Merkblättern und Formularen per E-Mail (Fax geplant, aber nicht implementiert, da nicht sinnvoll)
6. Erkennung von beliebigen E-Mail-Adressen

   (nicht erreicht, Gründe und Lösung weiter unten)
7. Memory: das System soll sich merken können, welches Thema der Benutzer gerade angefragt hat und nach Wunsch zur Menüauswahl dieses Themas zurückkehren

Projektphasen (1)

- Informationen gesammelt, Meinungen der Studenten studiert
- Dialoge für das Auskunftssystem in Form von Flußdiagrammen erstellt
- den Spracherkenner und die Sprachsynthese gewählt
- Implementierung:
  - Programmstruktur ausgearbeitet
  - Grammatiken für die Spracherkennung geschrieben

Projektphasen (2)

- die α-Version erstellt (ohne Datenbankanbindung und Zusatz-funktionen wie E-Mail)
- Getestet
- Zusatzfunktionen implementiert, Sprachsynthese verbessert
- β-Version des Programms erstellt und getestet

Implementierung

- Die Middleware-Plattform Robot5
- Spracherkenner Philips SpeechPearl 2000
- Implementierung erfolgte in Zusammenarbeit mit Voicerobots
- Sprachsynthese Lernout & Hauspie

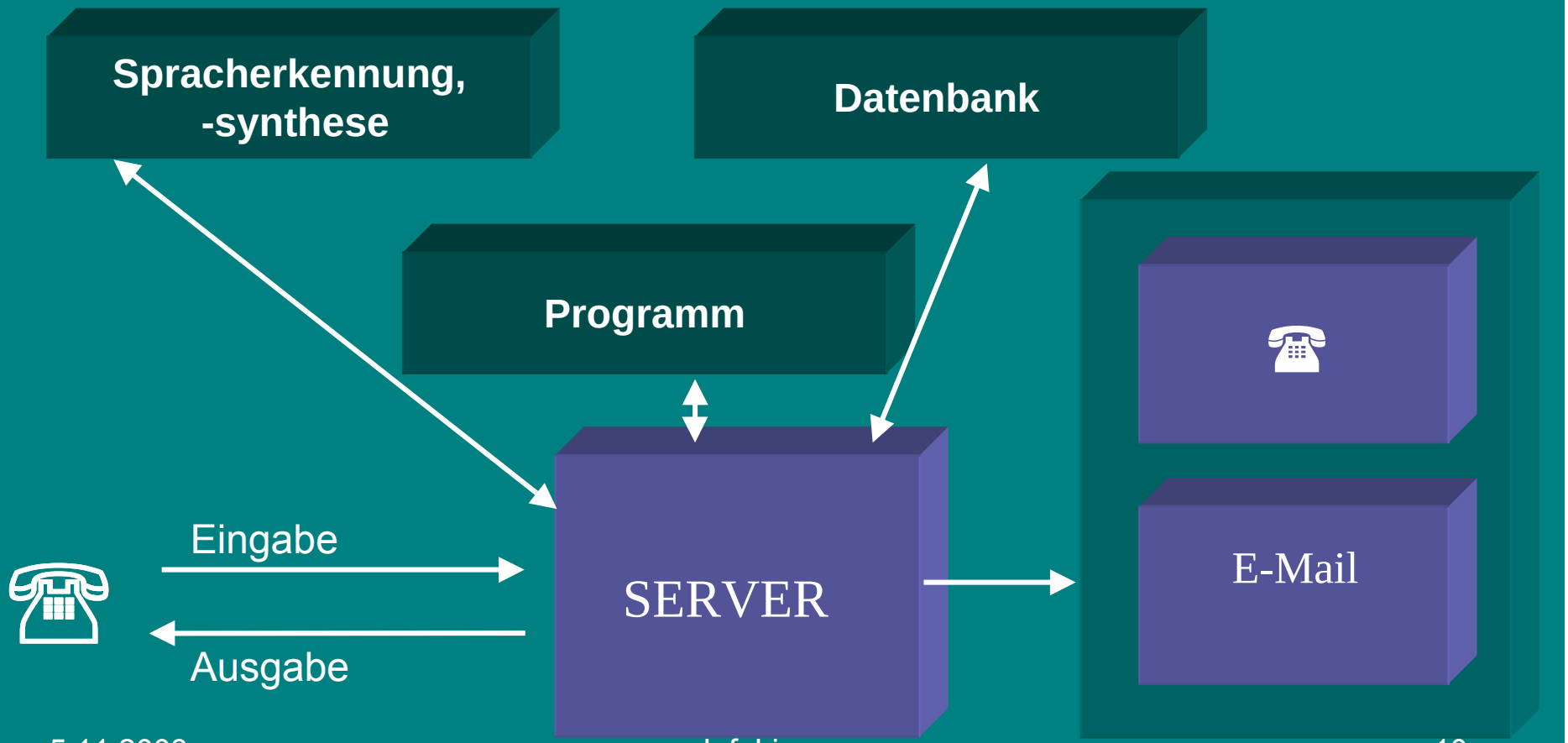
Die Middleware-Plattform Robot5

- Robot 5:
- objektorientiertes Programmierwerkzeug
- 2 Kernelemente: Server und Moderator

- Der robot 5 **Server** kann
  - einkommende Anrufe entgegennehmen
  - Programme ausführen
  - ausgehende Anrufe selber veranlassen

- Der **Moderator** ist der Applikationsgenerator für Sprachcomputeranwendungen

Systemarchitektur



Datenstrukturen: Stack

- Beim ersten Sprung aus einem übergeordneten Programm in ein Unterprogramm wird ein Stack gebildet
  (durch *chain*)
- Auf dem Stack werden die aktuellen Zustände des übergeordneten Programms gespeichert
- Bei jedem weiteren Sprung kommt ein neuer Eintrag auf den Stack
- Auf solche Weise ist es möglich, zu der Stelle im übergeordneten Programm wieder zurückzukehren, von der man in das Unterprogramm gewechselt hat.
  (durch *return*)

Datenstrukturen: Listen

- Dienen dazu, Listen oder Arrays ähnlicher Variablen zu manipulieren.
- Die Linelisten werden im Modul Listen.r5 verwendet, um mehrere Erkennungsergebnisse nach Erkennungswahrscheinlichkeit zu sortieren und weiterzubearbeiten
- In LL1 werden die Erkennungsergebnisse gespeichert
- In LL2 die dazugehörigen Wahrscheinlichkeiten
- In LL3 die Ergebnisse mit gleicher Wahrscheinlichkeit >8000

  z.B. $(attribut, fristen, 4000)$(attribut, wohnen, 10000) $(attribut, wohnen, 10000)

Spracherkennung

- Da keine umfangreichen semantischen Ressourcen vorhanden, muss das Gespräch durch Fragen gesteuert werden
- Das Thema ist vorgegeben, die Antworten können zum Teil frei formuliert werden (es wird durch offene Grammatiken ermöglicht)
- Ausnahme: Ja/Nein-Fragen

Grammatiken: Speech Recognition Control Language (SRCL)

- Speech Recognition Control Language - ein Grammatikformalismus
- eingeführt von: Speech Recognition API Committee und Enterprise Computer Telephony Forum
- ein spezieller Typ der Backus-Naur Form (BNF)
- legt syntaktisches Muster und spezifisches Vokabular fest
- SRCL weist kleine Unterschiede von Anbieter zu Anbieter auf
- wir benutzen den Formalismus von Philips SpeechPearl 2000

Offene vs. geschlossene Grammatiken

- Geschlossene Grammatiken:

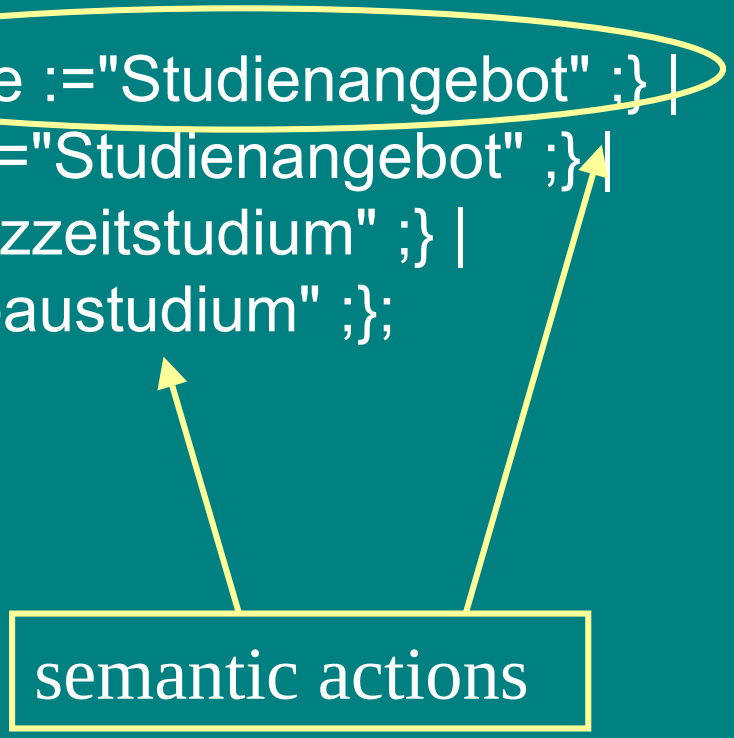
  der Anrufer ist auf die Wortsequenz angewiesen, die in der Grammatik definiert ist
- Offene Grammatiken(in unserem System):

  der Recognizer sucht nach bestimmten Konzepten, die in der Grammatik definiert sind, alles andere wird ignoriert

Grammatiken: semantische Informationen

- Erkannte Wörter oder Phrasen werden auf bestimmte Konzepte zurückgeführt:

- <Thema> = Studienangebot {eingabe :="Studienangebot" ;} |
  Studienmöglichkeiten {eingabe :="Studienangebot" ;} |
  Kurzzeitstudium {eingabe :="Kurzzeitstudium" ;} |
  Aufbaustudium {eingabe :="Aufbaustudium" ;};



Grammatik (in SRCL): ein Beispiel

```
declarations
{
    String studierender: <Command>, <Studierender>;
}
conceptset { <Studierender> }

<Command> = [<OptionalePhrase>] <Studierender> [<OptionalePhrase>]
{
        studierender := <Studierender>.studierender;
};
<Studierender> = deutscher Studierender  {studierender := "deutsch";} |
        ausländischer Studierender  {studierender := "ausländisch";} |
        Deutscher  {studierender := "deutsch";} |
        Ausländer {studierender := "ausländisch";};

<OptionalePhrase> = ich möchte mich bewerben als |
        ich will mich bewerben als |
        als |
        ich möchte mich als |
        ich will mich als |
        ich bewerbe mich als |
        bewerben ;
```

Probleme mit Spracherkennung (1)

- Bedingt durch offene Grammatiken werden mehrere Ergebnisse zurückgegeben
- Nebengeräusche
- Schlechte Ergebnisse bei Einzelbuchstabenerkennung und bei sehr kurzen Wörtern

Probleme mit Spracherkennung (2)

- Lösungsversuche:

  durch Ändern von Grammatiken (*additional words*), Erhöhung der zugelassenen Wahrscheinlichkeit,

  Einführung von Linelisten, um im nachhinein das beste Erkennungsergebnis herauszufiltern
- Weitere Möglichkeiten wären: Trainieren, korpus-unterstützte Techniken, Ändern von allgemeinen Tuning Parametern

Implementierung der E-Mail-Funktion

- Das ursprüngliche Ziel war: jede beliebige E-Mail-Adresse zu erkennen
- Die Fehlerrate bei Einzelbuchstaben-erkennung ist zu hoch
- Lösung: die Matrikelnummer erkennen und die entsprechende E-Mail-Adresse aus DB (mit allen Uni-Adressen) holen
- Fehlerrate kleiner, aber immer noch zu hoch für den praktischen Einsatz

TTS (Lernout & Hauspie)

- Hauptprobleme mit der Betonung und Intonation
- Verbesserungsversuche durch Satzumstellung, Umformulierung, Hinzufügen von Blanks und Buchstaben *(Studenten sekre tariaat)*

Datenbank

- **Back-End**: MS SQL-Server
- **Front-End**: MS Access
- Einerseits zum Abspeichern von Daten, die oft geändert werden müssen: Fristen, Gebühren, aktuelle Matrikelnummern usw.
- Andererseits damit die Daten dynamisch einglesen werden können (z.B. je nach SS, WS, die in Kalenderwochen definiert sind)

Zusammenfassung und Ausblick(1)

- Mit gegebenen Mitteln ist es nicht möglich eine freie Gesprächsführung zu ermöglichen
- Das Projekt zeigt eine Möglichkeit, dass die Antworten im Rahmen der vorgegebenen Themen vom Anrufenden zum Teil frei formuliert werden können

Zusammenfassung und Ausblick (2)

- Mit den uns verfügbaren Ressourcen, wäre eine noch flexiblere Gesprächs-steuerung durch den Anrufenden möglich, allerdings verbunden mit einem erheblichen Entwicklungsaufwand.
- Eine wirklich freie Gesprächsführung wäre nur mit entsprechenden semantischen Ressourcen möglich