Spezifikation

Studienprojekt:

InfoLine

Lehrstuhl für Computerlinguistik
Ruprecht-Karls-Universität Heidelberg

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

28.05.2003

Ziele des Projekts

Ein telefonisches Auskunftssystem für das Studentensekretariat zu implementieren, das folgende Aufgaben erfüllen soll:

- Antworten auf häufig gestellte Fragen geben, um das Studentensekretariat zu entlasten
- direktes Weiterverbinden mit entsprechenden Kontaktpersonen
- Zuschicken von angeforderten Merkblättern und Formularen per E-Mail / Fax

Phasen des Projektablaufs (1)

- Informationen sammeln, Meinungen der Studenten studieren
- Dialoge für das Auskunftssystem in Form von Flußdiagrammen erstellen
- Wahl des Spracherkenners und der Sprachsynthese
- Implementierung:
    - Programmstruktur ausarbeiten
    - Grammatiken für die Spracherkennung schreiben
    - die α-Version erstellen (ohne Datenbank-anbindung und Zusatzfunktionen wie E-Mail)

Phasen des Projektablaufs (2)

- Zusatzfunktionen implementieren, Sprachsynthese durch Phonemschrift verbessern

- Testen
- β-Version des Programms erstellen und testen

Implementierung

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

Die Middleware-Plattform Robot5

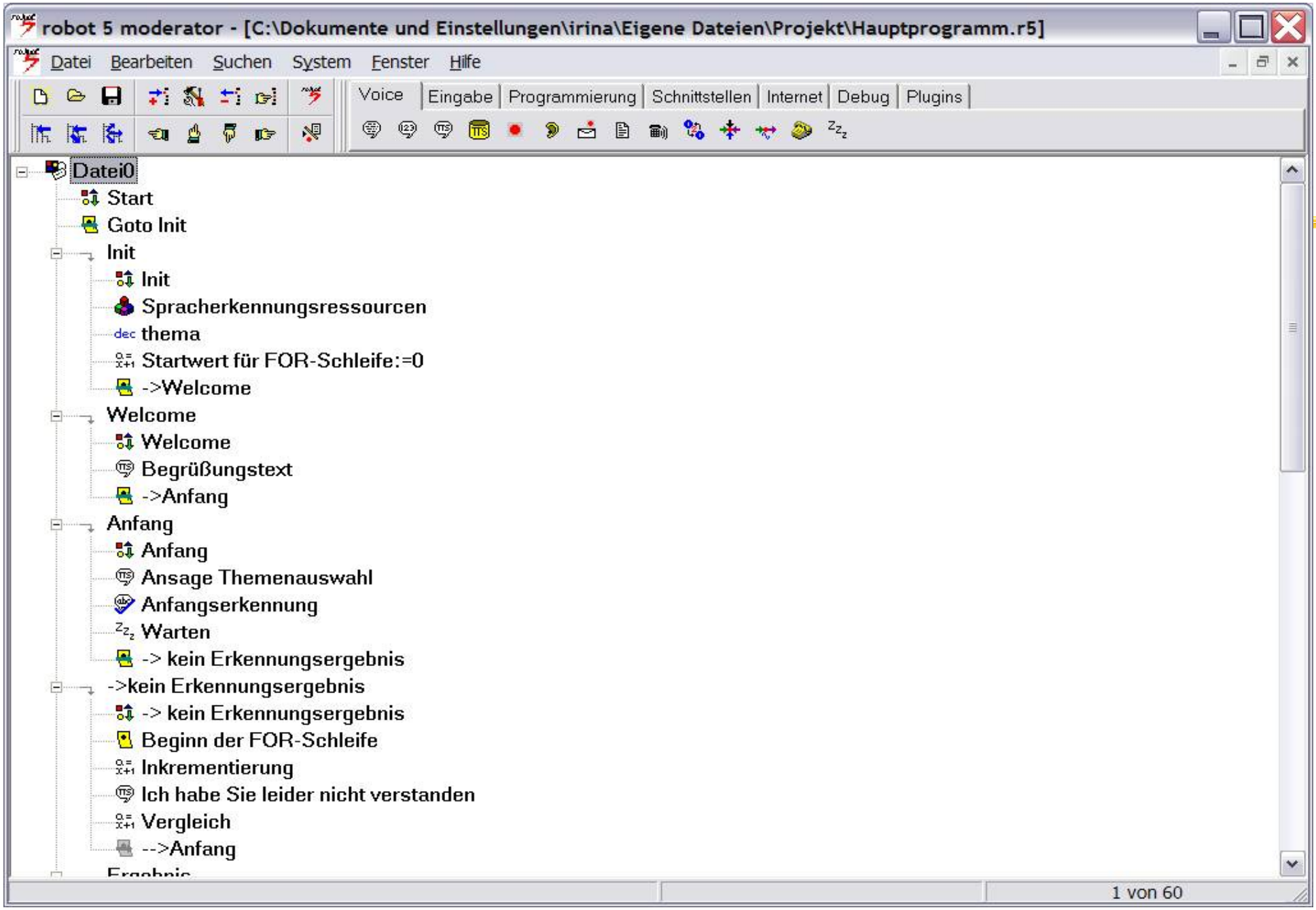
Robot 5:

- 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


robot 5 moderator - [C:\Dokumente und Einstellungen\irina\Eigene Dateien\Projekt\Hauptprogramm.r5]
Datei Bearbeiten Suchen System Fenster Hilfe
Voice Eingabe Programmierung Schnittstellen Internet Debug Plugins
Datei0
Start
Goto Init
Init
Init
Spracherkennungsressourcen
thema
Startwert für FOR-Schleife:=0
->Welcome
Welcome
Welcome
Begrüßungstext
->Anfang
Anfang
Anfang
Ansage Themenauswahl
Anfangserkennung
Warten
-> kein Erkennungsergebnis
->kein Erkennungsergebnis
-> kein Erkennungsergebnis
Beginn der FOR-Schleife
Inkrementierung
Ich habe Sie leider nicht verstanden
Vergleich
-->Anfang
1 von 60

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

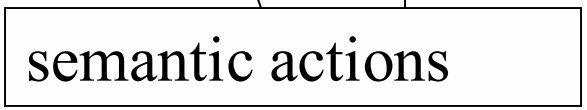
Offene Grammatiken (in unserem System) **:**

• 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:

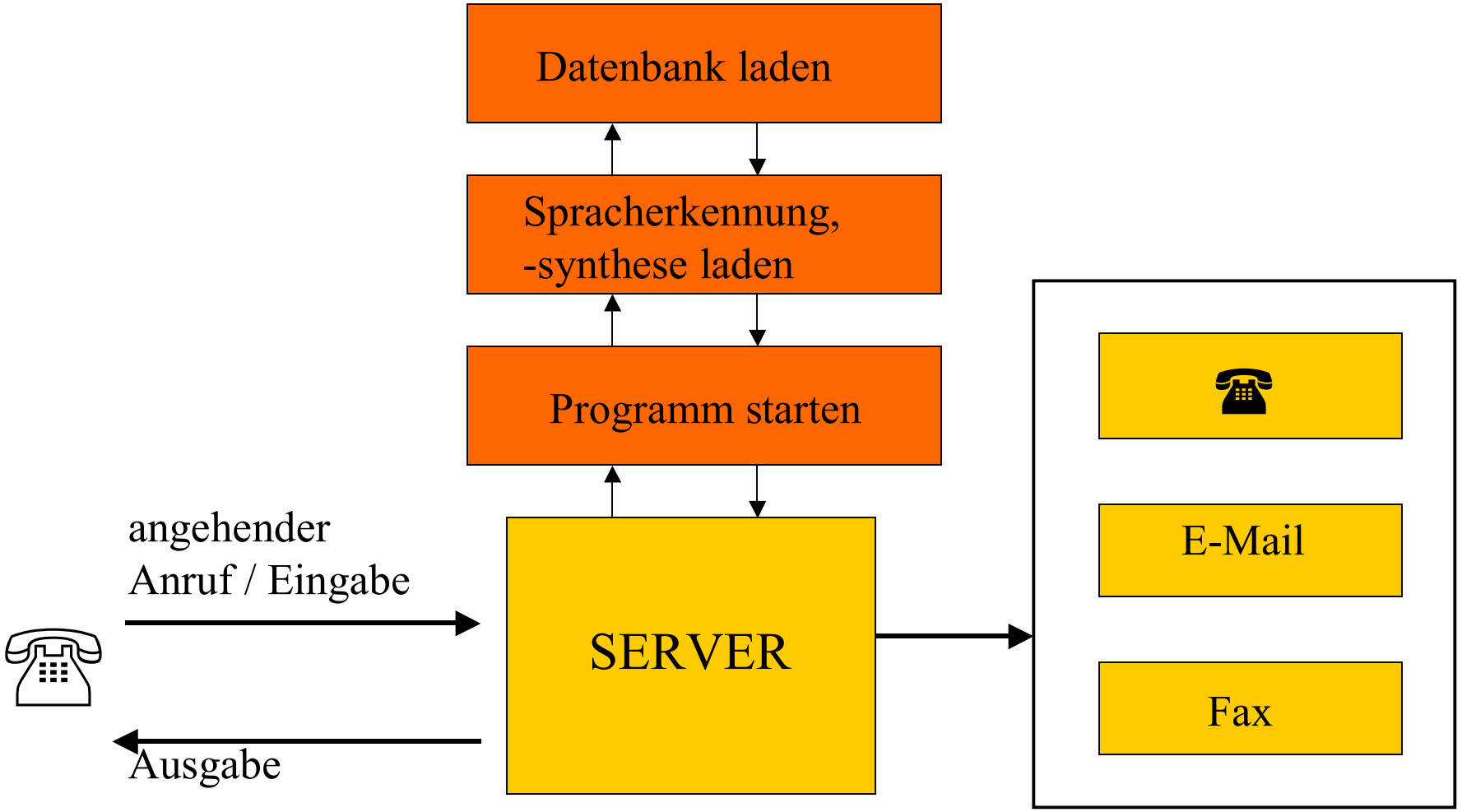
<Thema> = Studienangebot {eingabe :="Studienangebot" ;} |
Studienmöglichkeiten {eingabe :="Studienangebot" ;} |
Kurzzeitstudium {eingabe :="Kurzzeitstudium" ;} |
Aufbaustudium {eingabe :="Aufbaustudium" ;};



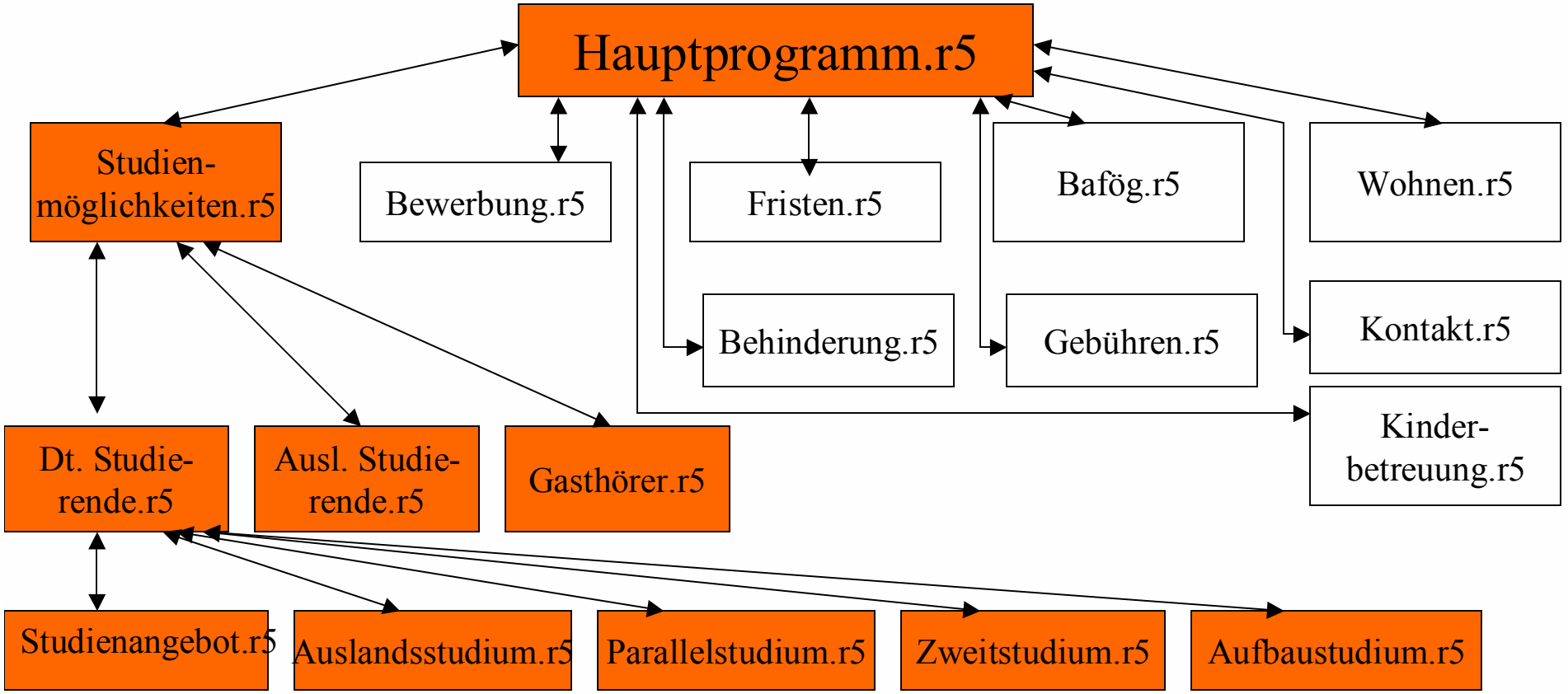
Grammatik (in SRCL): ein Beispiel

```
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 ;
```

Systemarchitektur



Programmstruktur (1)



Programmstruktur (1), Datenstruktur Stack

- Beim ersten Sprung aus einem übergeordneten Programm in ein Unterprogramm wird ein Stack gebildet
- 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.

Programmstruktur (2)