Designer codes dialogue as a sequence of prompts and responses.

- Especially for adding in common functionality to each state such as "cancel" or "go back" and confirmations.

Form Filling Dialogue

- Confirmations
- Shortcuts

Overview
Form-Filling Dialogues with CSLU Toolkit

- Single state with self-loop and complex Tcl code for form
- Use package command to bring in TTS functionality, speecho
- Can change recognizer grammar with Tcl code
- TTS output from nel command
- Drives dialogue based on forms
- All domain information stored in data structures

Form-Filling Dialogues

- Very similar to a frame
- Collection of items called form
  + Might not be needed
  - Interpretation Grammar
    - Recognition Grammar
  - Prompt where do you want to go to
  - Ask destination
    - ask line
    - ask option
    - ask destination
- Have data structure that drives the dialogue flow
Controlling what slot to work on

Work on the first slot that does not have a value

Tcl Code

set formslots {dest origin}
set prompt(dest) "Where do you want to fly to"
set grammar(dest) "portland | new%% york|newyork | ..."
set prompt(origin) "Where do you want to fly from"
set grammar(origin) "portland | new%% york|newyork | ..."

proc Converse {} {
    while {1} {
        DoPrompt
        GetResponse
    }
}

proc Converse {} {
    while {1} {
        DoPrompt
        GetResponse
    }
}
Repeat

- Allow user to say 'repeat' so that the system will repeat what it just said.

```tcl
proc GetResponse {} {
    # set Speech recognizer to user grammar for slot $::doslot
    # run Speech recogniser to user grammar to slot $::doslot
    #
    set ::value($::doslot) $words
    if {$words == {repeat}} {
        set ::grammar "$::grammar
Repeat"
    }
} {}
```

Understanding the Response
Explicit Confirmations

- Form can indicate which items should have immediate confirmations

  - If no, repeat original prompt
  - Load recognition grammar of yes and no
  - "Did you say..."

- Could have Tcl code ask for a confirmation after each query

- Could delay confirmation query till end of form

- Form could specify how to build a confirmation

- Could delay confirmation query till end of form

Overview

• Form Filling Dialogue

• Confirmations

• Shortcuts
Overview

Form Filling Dialogue

- Confirmation
- Showcasing

Implicit Confirmations

- At each step, could say previous item and prompt for next
  - So San Francisco. Where do you want to fly to?
- Or designer can specify implicit confirmation prompt
  - Flying from San Francisco. Where do you want to fly to?
- Add "no" to grammar of next prompt

- Will this work? Will users know to do this naturally?
  - Or each step, could say previous item and prompt for next

What else could we do?
Actually, need something more complicated that allows any order of extra information.

```plaintext
foreach slot $::form {
    if {![info exists ::value($slot)] && $slot != $::doslot} {
        append grammar " \$::implicitgrammar($slot)"
    }
}

set grammar $::explicitgrammar($::form)
```

### Concatenating Grammars

- Have two forms of grammars for each item
- Note that extra information had contextual clues around it
- When system asks question, user might give extra information

### Allowing Multiple Items in Response

- When item is explicitly prompted for:
  - e.g. Denver from Chicago
  ```plaintext
  set egrammar(origin) "portland | newyork | ..."
  ```
- When item is not explicitly prompted for:
  - e.g. Denver
  ```plaintext
  set igrammar(dest) "to (portland | newyork | ...)
  ```
- Build current grammar by concatenating item grammars
Initial prompt

Rather than ask for a specific parameter, ask a general question:

What flight would you like?

Make grammar by concatenating subgrammars (explicit versions)

From first response, fill in as much information as possible

Then start prompting for missing items

Filling in the values

How do we get values for each parameter?

Explicitly given values often have contextual clues

Ensure odd numbered tokens have the slot name

Ensure even numbered tokens have the slot value

Denver from Chicago

Explicit grammars for origin:

'airport (Portland | New York | ...)'

Result from recognizer will then have appropriate slot values embedded