

A Rapid Prototyping Methodology for Application Dependent Dialogue Models

Martin Rajman

Miroslav Melichar

David Portabella

Objectives of this lecture

1. Present a methodology for rapid dialogue prototyping
2. Illustrate its implementation possibilities on the examples derived from concrete projects

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system; WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Introduction

- **Rapid dialog prototyping (RDP)** is a significant part in the development process of interactive systems, especially for the ones with a **vocal interface**.
- However, due to the **complexity** of spoken languages and their strong **dependence on the applicative context**, there does not exist yet a really generic approach for dialog design; each application requires the development of a **specific model**.

Projects using the RDPM

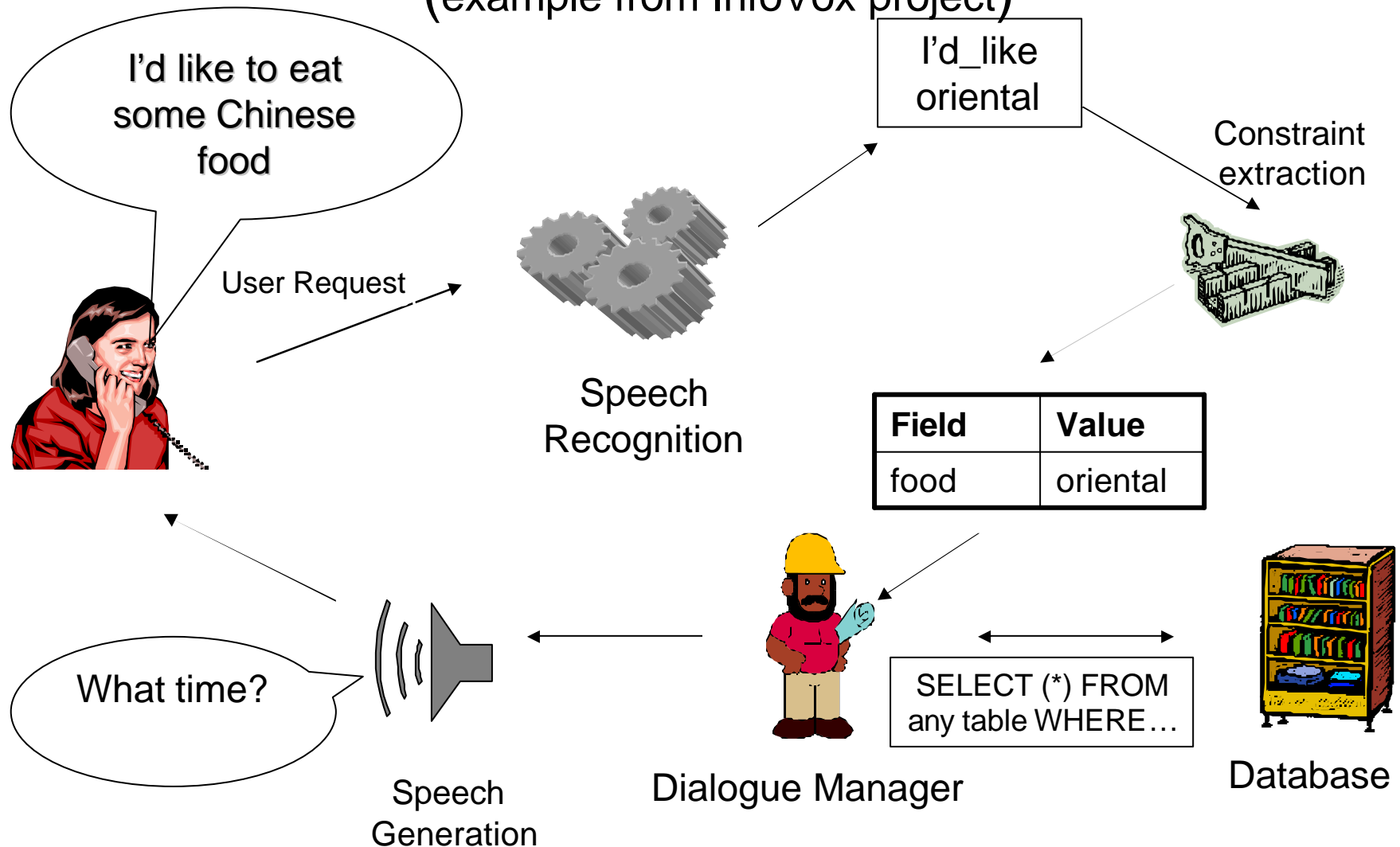
- InfoVox: Deployment of a dialogue-based vocal server providing information about the restaurants in the city of Martigny, Switzerland
- Inspire: Integration of an interactive, user friendly, dialogue-based assistant for the vocal control of home appliances installed in a smart home environment
- IM2.MDM (Archivus): Designing efficient, dialogue-based interaction mechanisms for a vocal interface to a database containing multimodal meeting recordings

Practical session

- Part 1.1: Installation
- Part 1.2: Quick hands-on

Typical dialogue-based vocal system

(example from InfoVox project)



Rapid Dialogue Prototyping

Main idea

Rapid design and production of a deployable dialogue based interactive system and its improvement through an iterative process based on WoZ experiments.

Five mains steps:

1. Produce a model of the targeted application (the task model)
2. Derive an initial dialogue model from the obtained task model
3. Carry out Wizard-of-Oz experiments to improve the initial dialogue model
4. Carry out an Internal Field-test to further refine the dialogue model
5. Carry out an External Field-test to evaluate the final dialogue model

Outline

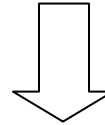
- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Task model

- The task is described in the form of a set of relational tables (frames), where the rows correspond to the possible target functions (also called “solutions” or “targets”) and the columns are the attributes needed to uniquely identify each of the functions and to invoke it.
- It can also be seen as the definition of the valid constraints (e.g. the list of available attributes and attribute combinations, as well as the possible associated values)

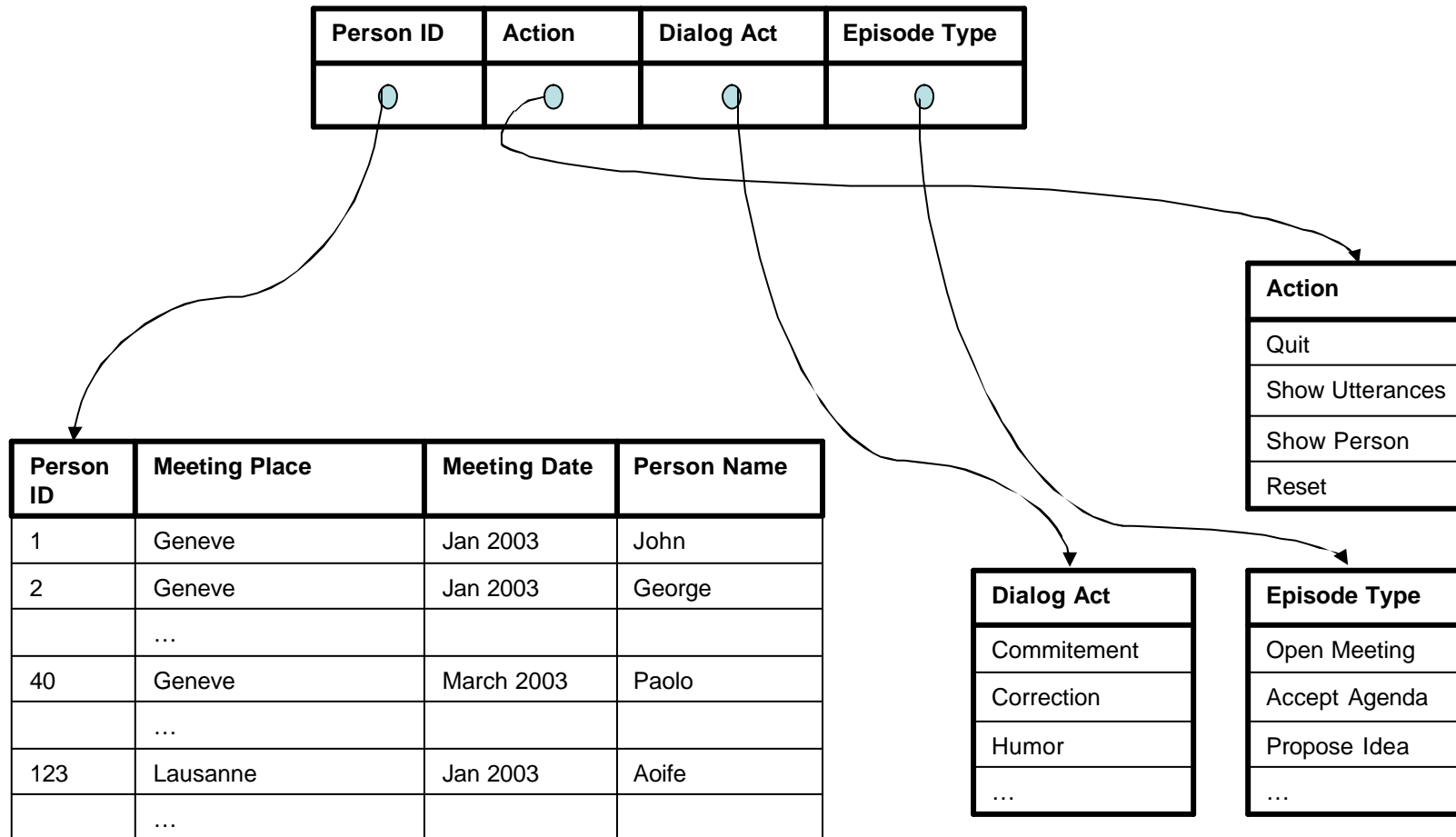
Task Modeling (InfoVox)

Select_restaurant(Type of kitchen, Price, Day, Time, Localization)

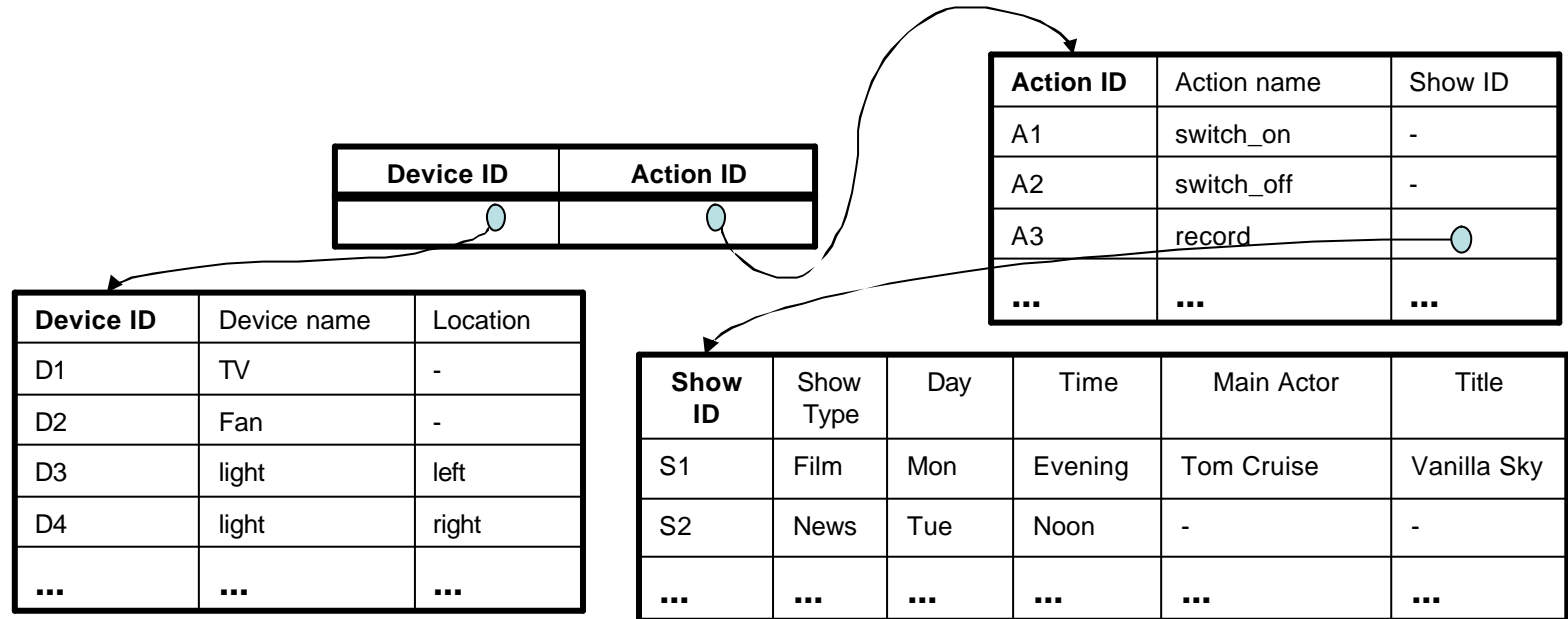


Restaurant ID	Type of kitchen	Price	Day	Time	Localization
R1	Chinese	Cheap	Mon-Sat	11:00 – 22:00	Train station
R2	Italian	Medium	Tue-Sun	11:00 – 23:00	Center
R3	Gastronomic	Expensive	Mon-Fri	18:00 – 23:00	Near Bourg
...

Task Modeling (Archivus)



Task Modeling (Inspire)



Solution Table

DeviceName	DeviceLocation	ActionName	ShowType	Day	Time	Main Actor	Title
Fan	-	switch_on	-	-	-	-	-
Light	Left	Dim	-	-	-	-	-
TV	-	Record	Film	Mon	Evening	Tom Cruise	Vanilla Sky
...							

Practical session

- Part 1.3: Task modeling: The Smart Home application
(which corresponds to a simplified version of the Inspire project task model)

Solution Table

DeviceName	DeviceLocation	ActionName	Day	TVShowName	Channel	TVShowAction
Fan	-	switch_on	-	-	-	-
Light	Left	Down	-	-	-	-
TV	-	Record	Mon	Tomorrow	TSR	Record
...						

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Constituents of the Dialogue Model

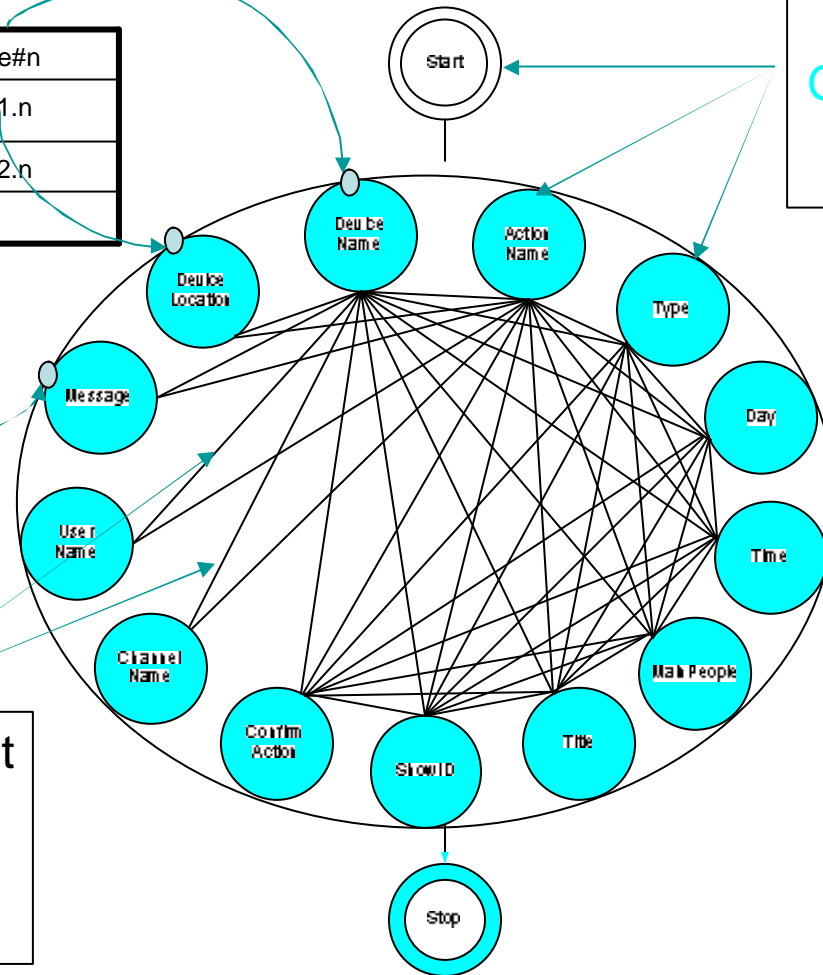
- The application-dependent, declarative specification of Generic Dialogue Nodes (GDNs)
- The application-independent (local and global) dialogue management strategies encoded in the corresponding (local and global) dialogue manager

Constituents of the Dialogue Model (example from Inspire)

Attribute#1	Attribute#2	...	Attribute#n
Value#1.1	Value#1.2	...	Value#1.n
Value#2.1	Value#2.2	...	Value#2.n
...			

Solution Table

Application dependent
Generic Dialogue Nodes
(GDNs)



Application independent
Dialogue Flow
Management Strategy

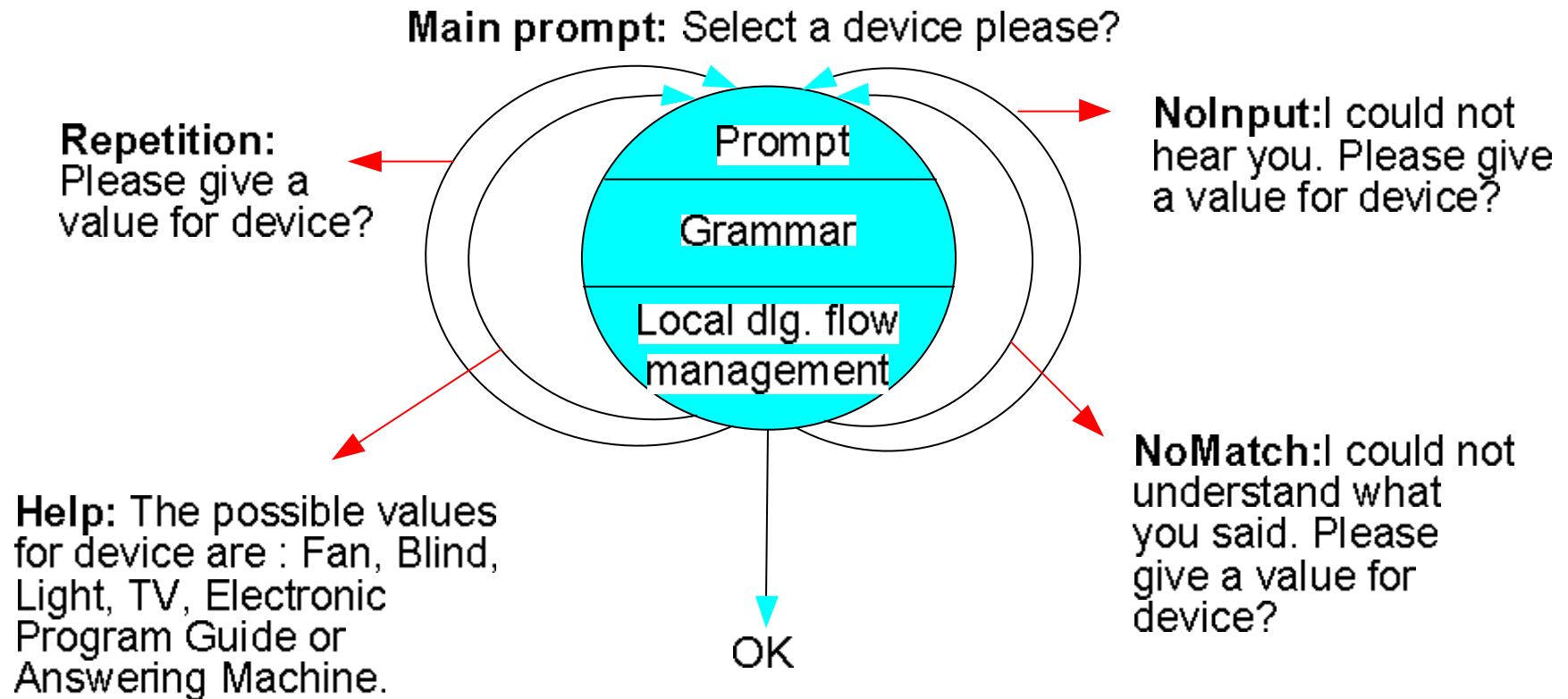
Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

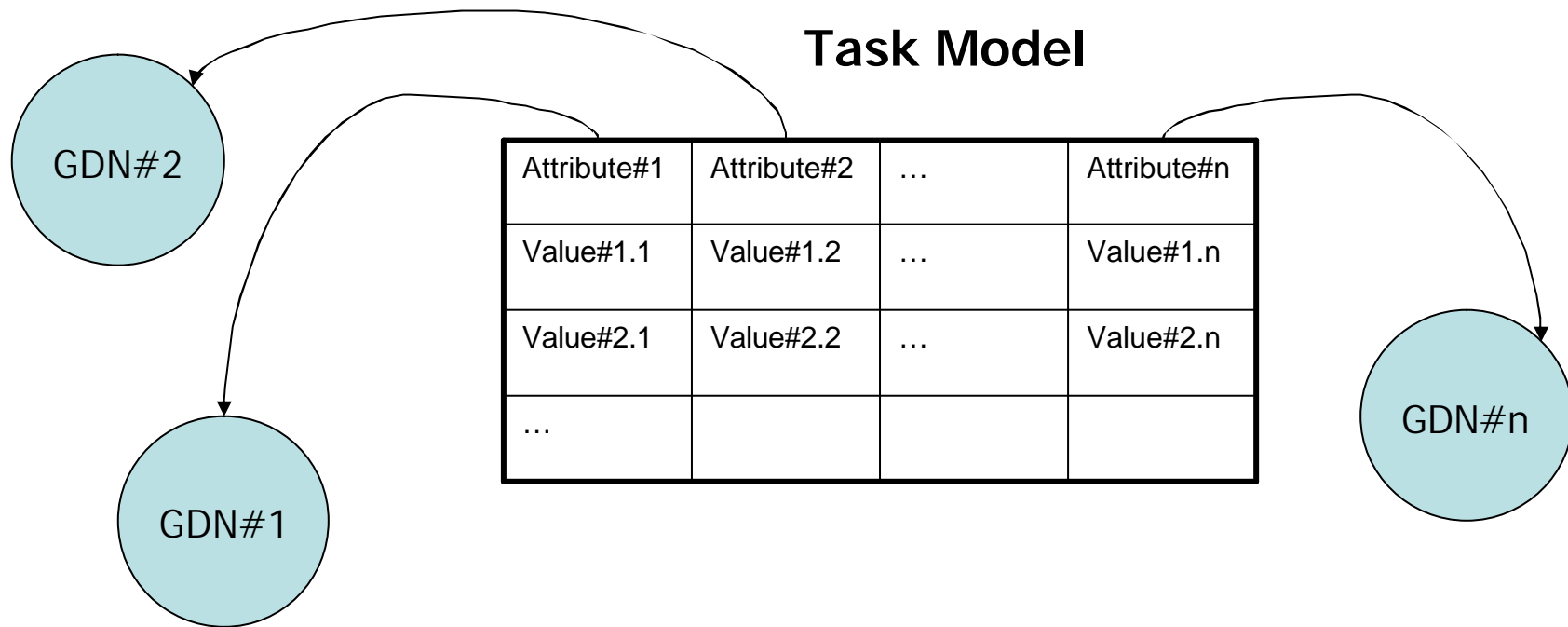
The role of the GDN

- The role of the Generic Dialogue Node (GDN) is to perform the interactions with the user that is required to obtain a valid value for the attribute.
- Different types of interactions are possible according to the nature of the values being sought (simple and list processing GDNs) to elicit a specific value for an attribute to allow the system to respond to a user query.
- In addition to the GDNs associated with some attribute in the task model, the system also contains Internal GDNs that are invoked by the dialogue manager in specific dialogue situations, such as at the beginning of the dialogue or when the user over-specified a request.

Generic Dialogue Node



Derivation of the GDNs



Notice: the dialogue model might also contain some GDNs That are not explicitly associated with an attribute in the Task model. Example: the initial (mixed initiative) start GDN

Generic Dialog Node: the prompts

- There are several prompts corresponding to the different situations of use of the GDN (normal question, repetition, help, ...)
- Notice that the formulation of the prompts is important for triggering various levels of mixed initiative. Example:
 - What can I do for you? (High level of mixed initiative is expected)
 - Do you want to operate the TV, yes or no? (Low level of mixed initiative is expected)

Examples of prompts

(from InfoVox project)

GDN prompts:

main prompt: "What type of cuisine would you like?"

help prompt: "You can choose Chinese, Italian or local cuisine."

reformulation prompt: "The possible choices are Chinese, Italian or local cuisine".

NoInput prompt prefix: "The system could not hear you."

NoMatch prompt prefix: "The system could not understand you."

Global dialogue management prompts:

implicit confirmation prompt prefix: "For the \$value as \$name"

explicit confirmation prompt prefix: "The system has understood that you'd like a \$value. Is it correct?"

incoherence prompt: "Could you repeat what type of cuisine you'd like?"

dead-end prompt: "The system was unable to find a restaurant corresponding exactly to your request. Do you want restaurant propositions that are close as possible?"

Examples of more advanced prompts

(from Inspire project)

GDN prompts:

<prompt type="main" cond="DeviceName:Fernseher"> Was moechten Sie mit dem Fernseher tun? </prompt>

<prompt type="main"> Was moechten Sie mit dem ausgewaehlten Geraet tun? </prompt>

<prompt type="help" cond="DeviceName:Rollo"> Ich kann das Rollo herauf- oder herunterlassen. </prompt>

<prompt type="help" cond="DeviceName:Ventilator">Ich kann den Ventilator ein- oder ausschalten. </prompt>

<prompt type="help">Ich kann folgendes tun: ...</prompt>

<prompt type="noInput">Ich konnte Sie nicht hoeren. Sagen Sie Hilfe, wenn Sie nicht weiter wissen. </prompt>

<prompt type="noMatch">Ich konnte Sie nicht verstehen. </prompt>

Global dialogue management prompts:

```
<implicitConfirmation>
  <begin> Ich habe</begin>
  <separator>,</separator>
  <lastSeparator>und</lastSeparator>
  <associationTemplate>$value als $gdnlabel</associationTemplate>
  <end>verstanden.</end>
</implicitConfirmation>
```

```
<incoherence>Meinen Sie $value1 oder $value2 fuer $gdnlabel? Bitte wiederholen Sie Ihre Eingabe.
</incoherence>
```

```
<implicitConfirmAfterGdnFailed>Ich habe keine verwertbare Eingabe erhalten. Ich werde versuchen
fortzufahren.</implicitConfirmAfterGdnFailed>
```

Generic Dialog Node: the grammar

Natural language understanding: making the connection between the surface forms in the user utterances and the concepts (attribute-value pairs) used in the task model.

Example: “What is on TV tonight?”

deviceName : TV

actionName : select_show

day : today

time : evening

Notice that the grammar is also used by the speech recognition engine to improve the quality of the recognition.

Example of grammar

(Java speech grammar format, JSGF)

The general form of a simple word spotting grammar is:

`<s> = <noise> | <keyword> | <s> <s>;` } Generic part, i.e. independent
from the application

`<noise> = word1 | word2 | ... ;`

`<keyword> = keyword1 {attribute1:value1}`

`| <keyword2> {attribute2:value2}`

`| ...;`

} Specific part, i.e. dependent
from the application

Example: `<keyword> = <actionName>;`
`<actionName> = (who | who is | give me the person that) {actionName:selectPerson}`
`| where {actionName:selectLocation}`
`| (exit | quit | good bye) {actionName:quite} ;`

More sophisticated grammars

If the application requires the use of more complex utterances, word spotting grammars might not be sufficient and more sophisticated type of grammars might have to be used.

Examples of more complex linguistic phenomena:

Negation: “Everything but a Chinese restaurant” \longrightarrow {restaurantType: not(Chinese)}

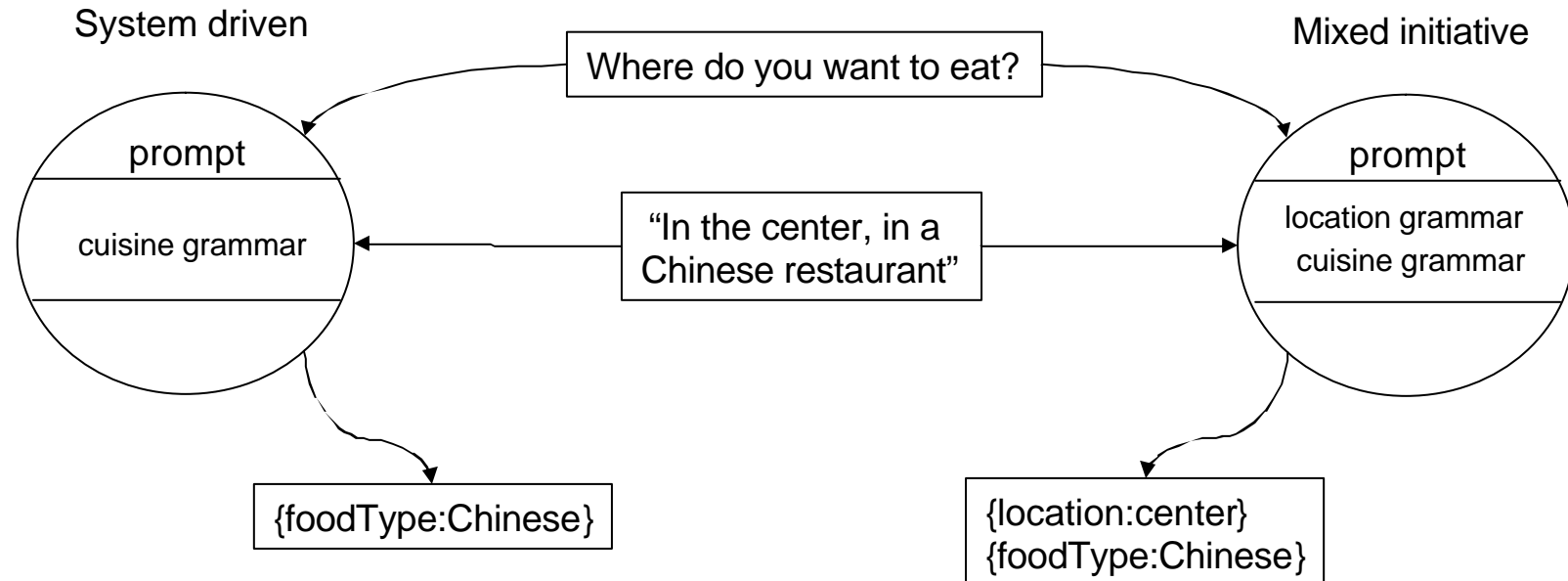
Multiple constraints: “Give me the meetings where
David answers to Alex on the budget”

\longrightarrow

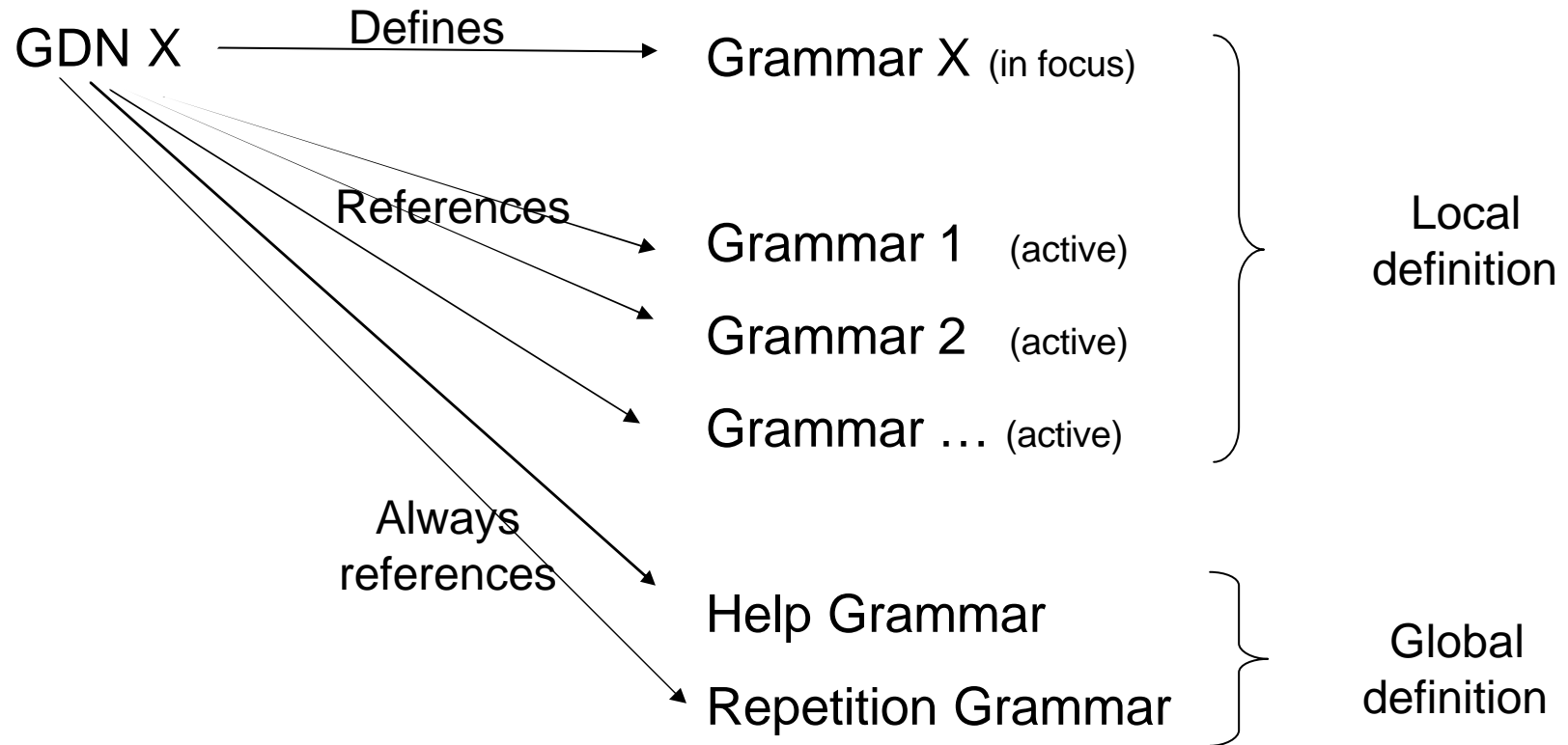
```
{
  dialogueAct( utterance1): question,
  dialogueAct( utterance2): answer,
  speaker( utterance1): Alex,
  speaker( utterance2): David,
  topic( utterance1): budget,
  topic( utterance2): budget
}
```

Level of mixed initiative

The level of mixed initiative is implemented through the notion of active grammar. The grammar of active GDN is active by definition, but, if required by the dialogue model designer, grammars of other GDNs can be simultaneously active as well.

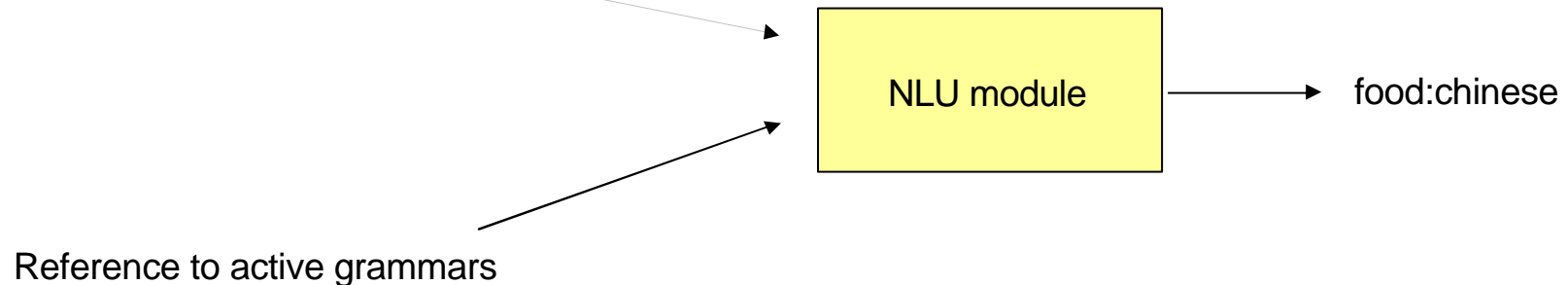


Different types of grammars



The NL understanding strategy

Utterance: "I would like to eat some
Chinese"

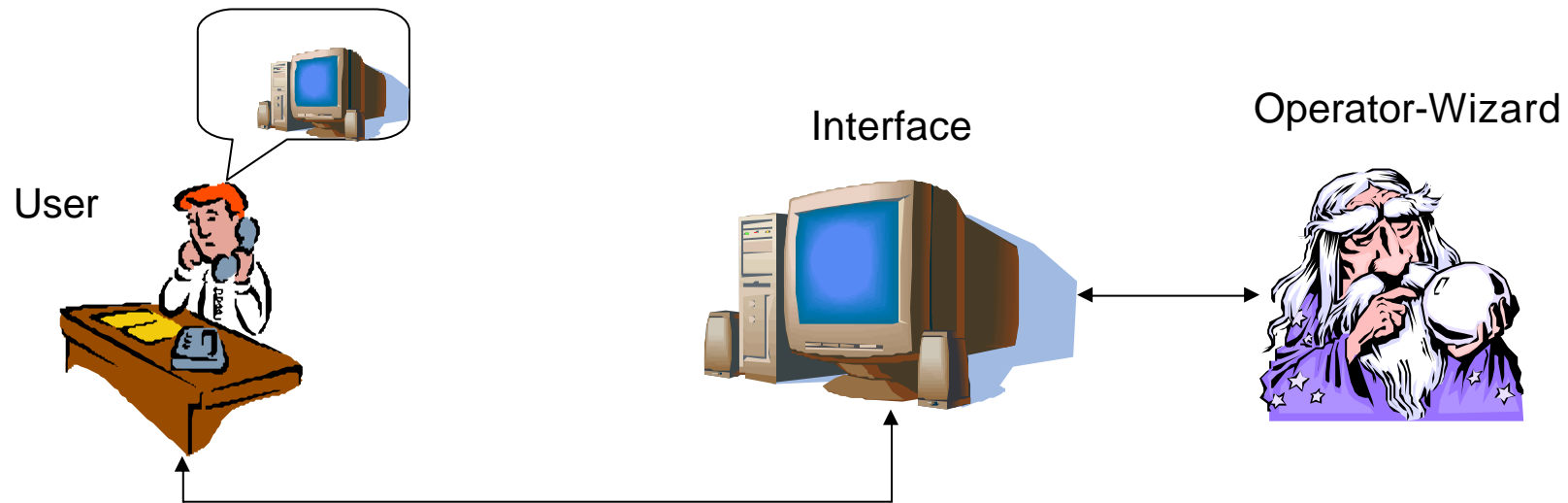


- A strategy needs to be defined to deal with utterances that are compatible with several active grammars:
 - Leftmost longest match is used (guaranties a unique segmentation of the input)
 - A priority rule is used for resolving the remaining ambiguities (first the active grammar in focus, then help grammar, repetition grammar, then the others grammars)

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system; WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Wizard-of-Oz experiment



Useful to:

1. simulate an automated system by a human operator;
2. get “real” data about “true” behavior of potential users;
3. evaluate the underlying dialogue model.

Wizard-of-Oz experiment

- Allows (in early steps of the design) the acquisition of experimental data about the behavior of the users when interacting with the system.
- Not yet implemented functionalities are simulated by the Wizard.
- The Wizard uses a Wizard's Control Interface (WCI) to fulfill his task.
 - The interface is generated automatically;
 - Dialogue data is visualized so that the Wizard can check or modify it, or even create new pieces of data if necessary.
- Currently, the WCI allows the simulation or supervision of the following modules:
 - automatic speech recognition, natural language understanding, and (re)starting or stopping the entire system.

WoZ experiment: the interface

The screenshot displays the 'WoZ interface' window with the following components:

- Control Buttons:** Start, Restart, Stop.
- Output Component:**
 - *XML: `<multiMediaPrompt>I understood SwitchOn as Operation . What is the device you want to operate? </multiMediaPrompt>`
 - *Prompt: "I understood SwitchOn as Operation . What is the device you want to operate?"
- DialogStateInfo:**
 - State: running
 - CurrentGDN: DeviceName
 - Incoherence: false
 - Dead-End: false
 - DeviceName: (Fan, Light, TV)
 - Location: (TableLamp, YellowL...)
 - ActionName: SwitchOn
 - Day: 0
 - TVShowName: 0
 - Channel: 0
 - TVShowAction: 0
- Input Simulator - active (inputSource has finished):**
 - Radio buttons: Use Semantic Pairs, No Match, No Input
 - Fields: DeviceName (Light), Location (All), ActionName, Day, Channel, TVShowAction, _LIST, _BOOL, _HELP, _REPEAT, _FAIL, _NEWFOCUS
 - Buttons: Submit, AutoSubmit, Clear
- InputSRE simulator - inactive:**
 - Text transcription: all the lights
 - Buttons: Submit, NoMatch, NoInput

A concrete example

- The video presents the experiments with the German prototype of the Smart Home system. Philips, Eindhoven, Netherland, May 2004.
- [Play the video](#)

Practical session

- Part 1.4: a better understanding
- Part 1.5: prompts and grammars

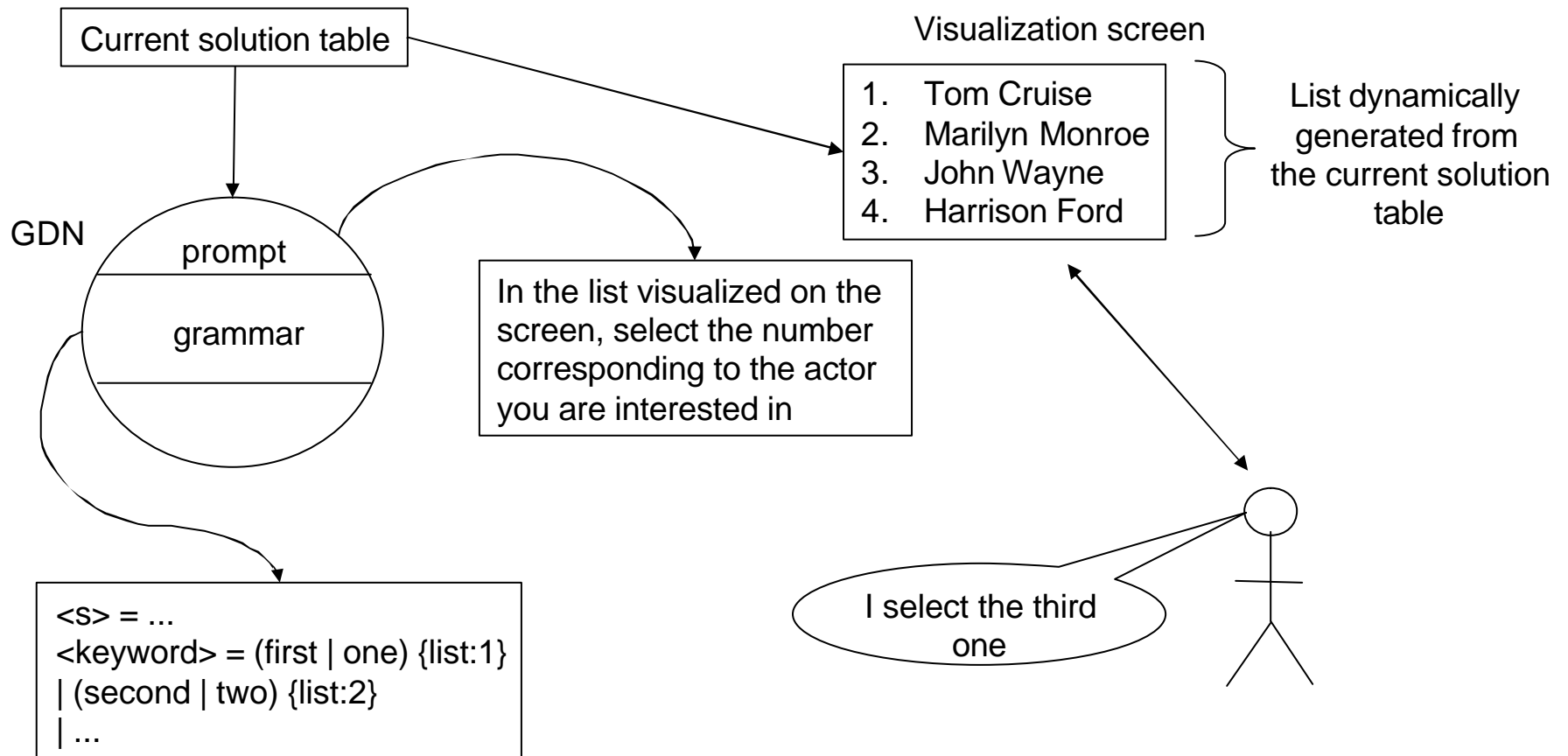
Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system; WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Dealing with dynamic information: Different types of GDNs

- Simple GDNs (also called static GDNs) associated with *static fields*, i.e. fields the values of which do not change in time, or change only very slowly; for example the price range for a given restaurant;
- List processing GDNs (also called dynamic GDNs) associated with *dynamic fields*, i.e. fields the values of which quickly change in time; for example the types of food in a selected restaurant or names of movies;
- Internal GDNs that are used to perform the interactions that are required by various special functions implemented in the dialogue manager (e.g. confirm a selected solution, start/reset the dialogue, select solution from the list of solutions, etc.).

Dynamic information (example)



Practical session

- Part 1.6: List Processing GDNs
- Part 1.7: The role of the Wizard

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Local dialogue management strategies

5 types of possible situations in any GDN:

- **OK:** the user answers the question in a usable way.
- **Repetition:** the user asks for the repetition of the question.
- **Help:** the user does not know how to answer the question.
- **NoInput:** the user didn't say anything.
- **NoMatch:** nothing can be extracted from the answer.

Local dialogue management strategies (cont.)

- In the case of the **OK** situation, control is handed back to the global dialogue manager which applies the global dialogue flow management strategy for the activation of the next GDN.
- In the other 4 situations, control remains at the GDN level.
- A strategy needs also to be defined for the situations where an OK value and a “specific” situation(s) are simultaneously detected. Example: “I would like to operate the fan. Help me.”

Several possible strategies:

- the help grammar has priority: help request is treated, the rest is ignored.
- all the keywords are passed to the global dialogue manager;
- the GDN grammar has priority on the help grammar (the request for help is ignored);

Processing the “problematic” dialogue situations

(dialogue repair)

- **Help:** the current GDN remains active, with the *help prompt* instead of the main prompt;
- **Repetition:** the current GDN remains active and the *last prompt* is played again;
- **NoMatch/NoInput:** the current GDN is reactivated with the *NoMatch/NoInput prompt* prefix preceding the main prompt.

Notice that, in all cases, there is an upper limit for the number of consecutive times that a given GDN can be active. If this limit is exceeded, the control is handed back to the global dialogue manager with the appropriate error message.

Processing the “problematic” dialogue situations (2)

(dialogue repair)

- **Help:**

System: What device would you like to activate?

User: I don't know. What are the possibilities?

System: You can operate the lights, the fan, the blinds, the TV or the answering machine.

User: OK, switch on the light.

- **Repetition:** (using conditional prompts)

System: Which light do you want to switch on?

User: Sorry?

System: Which light do you want to switch on.

User: I didn't understand. Could you repeat?

System: For the light you want to switch on, what location?

- **NoInput:**

User: <>

System: I couldn't understand you. What type of cuisine do you want?

User: Chinese restaurant please.

- **NoMatch:**

User: Well, you know, I hesitate.

System: There was no interpretable value in your request. What location for Chinese restaurant?

User: Well, what are the possibilities?

Practical session

- Part 2.1: Local dialogue management strategy
 1. Help
 2. Repetition
 3. No input
 4. No match

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Global dialogue management strategies

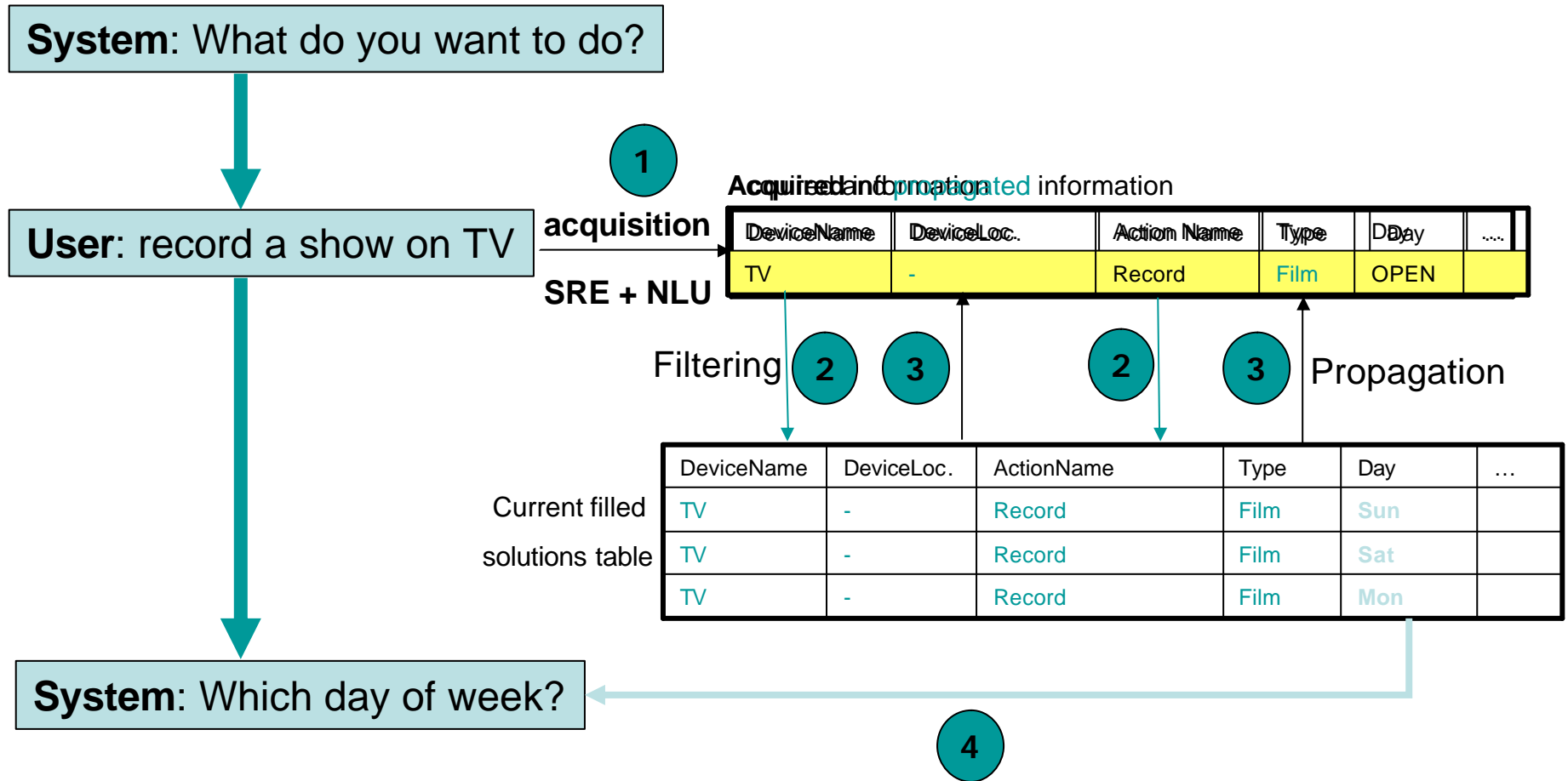
- a branching logic defining the next GDN to be activated;
- a dialogue dead end management strategy to deal with dialogue situations corresponding to zero solution;
- an explicit confirmation strategy for irreversible actions
- a strategy to deal with incoherencies
- a dialogue termination strategy (e.g. visualize the current solutions as soon as their number is lower than a pre-defined threshold)
- A strategy allowing an explicit focus (GDN) change on user's request

Branching logic: the acquire-filter-(propagate)-activate cycle

The proposed branching logic is independent from the application. It only relies on the fact that the task model is expressed in the form of a relational table.

- **Acquire**: some concepts are obtained from the user through the interaction with the current GDN;
- **Filter**: the obtained values are added to the set of already acquired ones and the current solution table is filtered in order to only contain the solutions that are compatible with the current set of values;
- **(Propagate)**: for the attributes for which all the current solutions have the same concepts, the value is propagated, i.e. considered as "implicitly" acquired for the attribute;
- **Activate**: the next "open" attribute (i.e. the first attribute in the current solution table still associated with a heterogeneous set of values) is identified, and the associated GDN is activated.

Branching logic: Example



Practical session

- Part 2.2: Global dialogue management strategy
 1. Branching logic

Dialogue termination strategy

When the interaction with the user should be terminated?

1. The number of the solutions is exactly 1. In this case, the solution is simply executed.
2. There are no solution. This happens when the user's request is over constrained. The dead-end strategy is triggered (see next slides).
3. The number of the solutions is higher than 1 AND the branching strategy cannot provide a GDN to select (i.e. all the slots are filled-in or their value was propagated). This typically happens when user provided "ANY" as a value for some of the attributes.
 - a. If (the number of solutions) \leq (a predefined threshold): provide list of the solutions to the user and ask him to select one.
 - b. Otherwise: the interaction fails and an automatic restart is triggered

Practical session

- Part 2.2: Global dialogue management strategy
2. Dialogue termination

Dialogue dead end management

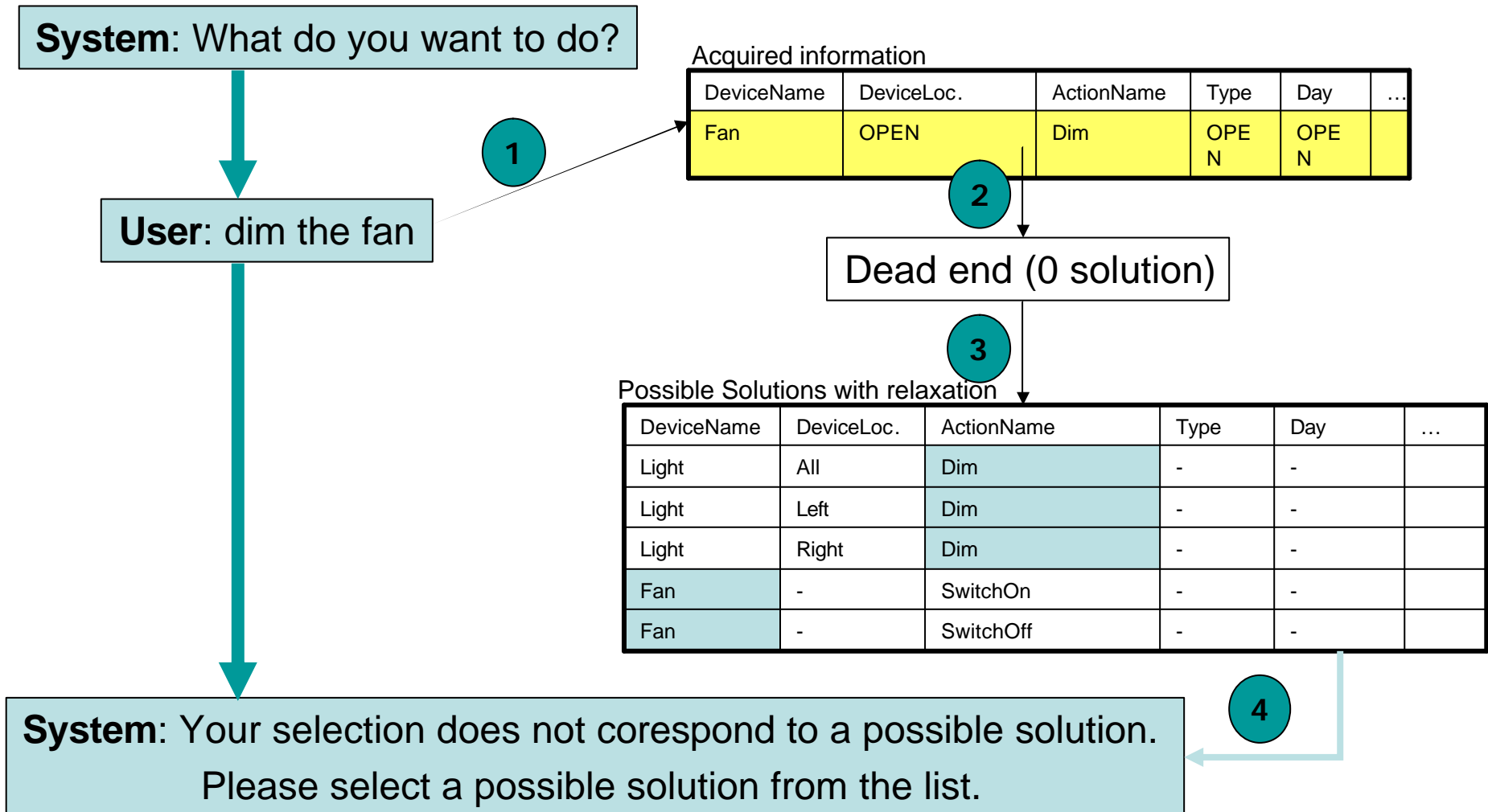
This management is required to deal with cases where the current solution set is empty (over constrained user request).

1. “Close solutions” are solutions that are compatible with all values that have been explicitly provided by user, but one.
 - If (the number of close solutions) == 0: reset the dialogue.
 - If (the number of close solutions) == 1: ask user for confirmation of this solution and select it if confirmed.
 - If (the number of close solutions) <= (a predefined threshold): provide list of the solutions to the user and ask him to select one.

Otherwise ((the number of close solutions) > (a predefined threshold)):

1. Generate a list of “relaxable” attributes. An attribute is relaxable if removing its value from the current constraints yields more than one solution.
2. For each relaxable attribute, ask the user if he agrees to change its value. As soon as the user agrees with a relaxation, activate the given GDN
3. If the user rejects all relaxation possibilities, reset the dialogue.

Dialogue dead end management: Example



Practical session

- Part 2.2: Global dialogue management strategy
- ## 3. Dead End management

Confirmation strategies

The procedure used during the dialogue to obtain user's confirmation of the values that have been recognized by the system. 2 possible approaches :

- Explicit confirmation: the confirmation is obtained by explicitly asking the user;
- Implicit confirmation: the confirmation is induced from the reaction of the user to a confirmation information associated with a standard question.

Implicit confirmation example:

System: What can I do for you ?

User: Turn on the light.

System: What light do you want to turn on?

User: On my left.

Explicit confirmation example:

System: What can I do for you ?

User: Turn on the light.

System: I understood that you request light as device and switch on as action. Is it correct ?

User: Yes.

Practical session

- Part 2.2: Global dialogue management strategy
- ## 4. Explicit Confirmation

Incoherence processing

- What to do when there are at least two different values for the same slot provided by the user?
- In our approach: the system asks the user to choose between them.
 - If there are more incoherent values (possibly in several slots), the system removes all other incoherent values.

Practical session

- Part 2.2: Global dialogue management strategy
5. Incoherencies

Explicit focus (GDN) change on user's request

- The user can ask the system to change the focus, e.g. “Let’s talk about the device I want to operate”.
- In our approach: Explicit focus change simply generates special attribute:value pair (`changeFocus:device`) that is taken into account by the branching logic when selecting the next GDN.

Practical session

- Part 2.2: Global dialogue management strategy
6. Change Focus

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Practical session

- Part 3: Improving the system

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Internal Field-Test

The aim is to improve the dialogue model (reformulation of system messages, improved feedback, ...) and to validate the evaluation procedure (coherence, understandability).

Set up:

1. Description of the system and of the evaluation procedure (3 minutes);
2. The user is put in a specific applicative context with a scenario (3 minutes);
3. The user is connected to the system (5-10 minutes);
4. A satisfaction questionnaire is submitted to the user (10 minutes).

Notice also that the users are friendly users (systems designers, colleagues, friends, relatives) that are not necessarily representative for the users of the targeted application

External Field-Test

The aim is to evaluate the final dialogue model according to the evaluation procedure defined during the Internal Field-test.

Users are external in the sense they are randomly chosen in the set of relevant users for the targeted application

External Field-Test

Input data:

- The answers to the satisfaction questionnaire;
=> **subjective indicators**: average scores obtained for the various closed question
- The log files produced by the system during the interactions with the users (interaction duration, number of turns, number of repair subdialogs, ...);
=> **objective indicators**: average values measured for various system characteristics

External Field-Test

3 different kinds of analyses were performed:

1. Retrospective trend analysis: the aim is to provide a synthetic view the opinion of the users about the system (identification of predominant modalities – "trends");
2. Retrospective correlation analysis: identify dependencies between pairs of subjective indicators;
3. Prospective correlation analysis: identify dependencies between objective and subjective indicators.

Practical session

- Part 4: Evaluation of the system

Outline

- Introduction
- Task model
- Dialogue model
 - Dialogue model: GDNs
 - A better understanding of the system: WOz experiments
 - Different types of GDNs
 - Local dialogue management strategies
 - Global dialogue management strategies
- Improving the system
- Evaluation
- Conclusions & References

Key points

- Five steps of the methodology for rapid dialogue prototyping;
- Choice of adequate task model;
- Importance of evaluation.

References

- 1) Rajman, M., Bui, T.H., Rajman, A., Seydoux, F., Trutnev, A., Quarteroni, S.: Assessing the usability of a dialogue management system designed in the framework of a rapid dialogue prototyping methodology. Acta Acustica united with Acustica, the journal of the European Acoustics Association (EAA) 90 (2004) 1096-1111
- 2) Bui, T.H., Rajman, M., Melichar, M.: Rapid dialogue prototyping methodology. In: Proc. of TSD 2004, Brno, Czech Republic, Springer-Verlag (2004) 579-586
- 3) Cenek, P., Melichar, M., Rajman, M.: A Framework for Rapid Multimodal Application Design. In: Proc. of TSD 2005, Karlovy Vary, Czech Republic, to appear.