Linköping Studies in Science and Technology

Dissertation No. 1113

Dialogue Behavior Management in Conversational Recommender Systems

by

Pontus Wärnestål

Linköping University
INSTITUTE OF TECHNOLOGY

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2007

Cover Design: “Preferences” Pontus Wärnestål

© Pontus Wärnestål, 2007

ISBN 978-91-85831-47-0
ISSN 0345-7524

Printed by LiU-Tryck, Linköping 2007

Till Cecilia

Abstract

This thesis examines recommendation dialogue, in the context of dialogue strategy design for conversational recommender systems. The purpose of a recommender system is to produce personalized recommendations of potentially useful items from a large space of possible options. In a *conversational* recommender system, this task is approached by utilizing natural language recommendation dialogue for detecting user preferences, as well as for providing recommendations. The fundamental idea of a conversational recommender system is that it relies on dialogue sessions to detect, continuously update, and utilize the user's preferences in order to predict potential interest in domain items modeled in a system. Designing the dialogue strategy management is thus one of the most important tasks for such systems.

Based on empirical studies as well as design and implementation of conversational recommender systems, a behavior-based dialogue model called BCORN is presented. BCORN is based on three constructs, which are presented in the thesis. It utilizes a user preference modeling framework (*preflets*) that supports and utilizes natural language dialogue, and allows for descriptive, comparative, and superlative preference statements, in various situations. Another component of BCORN is its message-passing formalism, PCQL, which is a notation used when describing preferential and factual statements and requests. BCORN is designed to be a generic recommendation dialogue strategy with conventional, information-providing, and recommendation capabilities, that each describes a natural chunk of a recommender agent's dialogue strategy, modeled in *dialogue behavior diagrams* that are run in parallel to give rise to coherent, flexible, and effective dialogue in conversational recommender systems.

Three empirical studies have been carried out in order to explore the problem space of recommendation dialogue, and to verify the solutions put forward in this work. Study I is a corpus study in the domain of movie recommendations. The result of the study is a characterization of recommendation dialogue, and forms a base for a first prototype implementation of a human-computer recommendation dialogue control strategy. Study II is an end-user evaluation of the ACORN system that implements the dialogue control strategy and results in a verification of the effectiveness and usability of the dialogue strategy. There are also implications that influence the refinement of the model that are used in the BCORN dialogue strategy model. Study III is an overhearer evaluation of a functional conversational recommender system called CoreSong, which implements the BCORN model. The result of the study is indicative of the soundness of the behavior-based approach to conversational recommender system design, as well as the informativeness, naturalness, and coherence of the individual BCORN dialogue behaviors.

Sammanfattning

I denna avhandling undersöks rekommendationsdialog med avseende på utformningen av dialogstrategier för konverserande rekommendationssystem. Syftet med ett rekommendationssystem är att generera personaliserade rekommendationer utifrån potentiellt användbara domänobjekt i stora informationsrymder. I ett *konverserande* rekommendationssystem angrips detta problem genom att utnyttja naturligt språk och dialog för att modellera användarpreferenser, liksom för att ge rekommendationer. Grundidén med konverserande rekommendationssystem är att utnyttja dialogsessioner för att upptäcka, uppdatera och utnyttja en användares preferenser för att förutsäga användarens intresse för domänobjekten som modelleras i ett system. Utformningen av dialogstrategihantering är därför en av de viktigaste uppgifterna för sådana system.

Baserat på empiriska studier, liksom på utformning och implementering av konverserande rekommendationssystem, presenteras en beteendebaserad dialogmodell som kallas BCORN. BCORNs bas utgörs av tre konstruktioner, vilka alla presenteras i denna avhandling. BCORN utnyttjar ett preferensmodelleringsramverk (*preflets*) som stöder och använder sig av naturligt språk i dialog och tillåter deskriptiva, komparativa och superlativa preferensuttryck i olika situationer. Den andra komponenten i BCORN är dess interna meddelande-formalism PCQL, som är en notation som kan beskriva preferens- och faktiska påståenden och frågor. BCORN är utformat som en generell rekommendationshanteringsstrategi med konventionella, informationsgivande och rekommenderande förmågor, som var och en beskriver naturliga delar av en rekommendationsagents dialogstrategi. Dessa delar modelleras i *dialogbeteendediagram* som exekveras parallellt för att ge upphov till koherent, flexibel och effektiv dialog i konverserande rekommendationssystem.

Tre empiriska studier har utförts för att utforska problemkomplexet som utgör rekommendationsdialog och för att verifiera de lösningar som tagits fram inom ramen för detta arbete. Studie I är en korpusstudie i filmrekommendationsdomänen. Studien resulterar i en karakteristik av rekommendationsdialog, och utgör basen för en första prototyp av dialoghanteringsstrategi för rekommendationsdialog mellan människa och dator. Studie II är en slutanvändarutvärdering av systemet ACORN som implementerar denna dialoghanteringsstrategi och resulterar i en verifiering av effektivitet och användbarhet av strategin. Studien resulterar också i implikationer som påverkar utformningen av den modell som används i BCORN. Studie III är en medhörningsutvärdering av det funktionella konverserande rekommendationssystemet CoreSong, som implementerar BCORN-modellen. Resultatet av studien indikerar att det beteendebaserade angreppssättet är funktionellt och att de olika dialogbeteendena i BCORN ger upphov till hög informationskvalitet, naturlighet och koherens i rekommendationsdialog.

Preface

Over the years that this research has been carried out, I have tried to keep in mind that research on conversational interaction with machines in the end must support real people when carrying out their tasks. Just like dialogue, this turns out to be a two-way street: The task of writing this thesis would not have been completed without the support of, and conversations with, real people.

First of all, I am indebted to Arne Jönsson, my main supervisor. He has guided and supported me over the years, and enthusiastically engaged in critical discussions about a great many topics. My secondary supervisor, Lars Degerstedt, has been a great influence and patiently discussed, and opened up my eyes for, many issues of engineering and technological aspects of software design and development. Apart from being great supervisors, you have been great collaborators and co-workers.

I would like to thank the members of the Natural Language Processing Laboratory (NLPLAB) and Human-Centered Systems (HCS) at the Department of Information and Computer Science, and everyone involved in the Swedish Graduate School of Language Technology (GSLT), for providing an active and stimulating research environment. I am also thankful to the technical and administrative staff, especially Lillemor Wallgren and Britt-Inger Karlsson for their help in the many administrative matters that surround a thesis production. A special thank you goes out to the people that participated in the user studies.

This thesis is based on several papers, which have been read and commented on by audiences and reviewers (who remain anonymous). Their valuable feedback has contributed greatly.

Teaching is an important part of my work, and the opportunity to teach at different institutions such as Linköping University, Göteborg University, and Chalmers has been fun and a great experience for me.

I would like to thank my parents and my dear brothers for their encouragement. And many thanks to my extended family, relatives, and friends. Thank you Oskar for our continuous dialogue (spanning decades now) on life, games, and thermos design. Thanks Martin for interesting discussions on many aspects of human-computer interaction and for being a great host on my many visits. Your prediction all those years ago turned out to be true!

Last, but certainly not the least, thank you Cecilia for your never failing support and encouragement.

Pontus Wärnestål, Olskroken, June 2007

Refereed Publications by the Author

- Pontus Wärnestål, Lars Degerstedt, and Arne Jönsson. Emergent conversational recommendations: A dialogue behavior approach. In *Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue*, Antwerp, Belgium, September 2007.

- Pontus Wärnestål, Lars Degerstedt, and Arne Jönsson. Interview and delivery: Dialogue strategies for conversational recommender systems. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (Nodalida)*, Tartu, Estonia, May 2007.

- Pontus Wärnestål, Lars Degerstedt, and Arne Jönsson. PCQL: A formalism for human-like preference dialogues. In *Proceedings of the 5th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Hyderabad, India, January 2007.

- Pontus Wärnestål. Modeling a dialogue strategy for personalized movie recommendations. In *Proceedings of the IUI Workshop "Beyond Personalization"*, pages 77–82, San Diego (CA), USA, 2005.

- Pontus Wärnestål. Modularized user modeling in conversational recommender systems. In L. Ardissono, P. Brna, and A. Mitrovic, editors, *Proceedings of 10th International Conference on User Modeling (UM05) LNAI 3538. Edinburgh, Scotland U.K.*, pages 545–547. Springer-Verlag, Berlin Heidelberg, 2005.

- Pontus Wärnestål. User evaluation of a conversational recommender system. In Ingrid Zukerman, Jan Alexandersson, and Arne Jönsson, editors, *Proceedings* *of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 32–39, Edinburgh, Scotland U.K., 2005.

As Pontus Johansson:

- Aseel Berglund and Pontus Johansson. Using Speech and Dialogue for Interactive TV Navigation. *Universal Access in the Information Society*, 3(3–4):224–238, October 2004.
- Pontus Johansson. *Design and Development of Recommender Dialogue Systems.* Licentiate Thesis 1079, Linköping Studies in Science and Technology, Linköping University, 2004.
- Pontus Johansson. Natural Language Interaction in Personalized EPGs. In *Proceedings of the 3rd UM Workshop "Personalization in Future TV"*, pages 27–31, Pittsburgh (PA), USA, 2003.
- Lars Degerstedt and Pontus Johansson. Evolutionary Development of Phase-Based Dialogue Systems. In *Proceedings of the 8th Scandianvian Conference on Artificial Intelligence*, pages 59–67, Bergen, Norway, November 2003.
- Pontus Johansson. MadFilm - a multimodal approach to handle search and organization in a movie recommendation system. In *Proceedings of the 1st Nordic Symposium on Multimodal Communication*, pages 53–65, Helsingör, Denmark, 2003.
- Pontus Johansson, Lars Degerstedt, and Arne Jönsson. Iterative Development of an Information-Providing Dialogue System. In *Proceedings of the 7th ERCIM Workshop "User Interfaces for All"*, pages 29–36, Paris, France, 2002.
- Aseel Ibrahim and Pontus Johansson. Multimodal Dialogue Systems: a case study for interactive TV. In *Proceedings of the 7th ERCIM Workshop "User Interfaces for All"*, pages 209–218, Paris, France, 2002.
- Aseel Ibrahim and Pontus Johansson. Multimodal Dialogue Systems for Interactive TV Applications. In *Proceedings of the 4th IEEE ICMI conference*, pages 117–222, Pittsburgh (PA), USA, 2002.

Table of Contents

1 Introduction 1

1.1 Aim and Research Questions . . . 2

1.2 Contributions . . . 4

1.3 Method . . . 5

1.4 Thesis Outline . . . 7

2 Background: Conversational Recommender Systems 11

2.1 Dialogue Systems . . . 12

2.1.1 Concepts and Definitions . . . 12

2.1.2 Dialogue Model Components . . . 14

2.1.3 Layered Control Architectures . . . 14

2.1.4 Computational Dialogue Management Formalisms and Classifications . . . 20

2.1.5 Other Dialogue Classification Schemes . . . 22

2.2 User Modeling and User Preferences . . . 23

2.2.1 User Modeling . . . 23

2.2.2 User Preferences . . . 27

2.2.3 Recommendation Models and Algorithms . . . 32

2.2.4 Recommendation Performance Factors . . . 36

2.3 Conversational Recommender Systems . . . 38

2.4 Development of Conversational Recommender Systems . . . 41

2.4.1 Design and Development Methodologies . . . 42

2.4.2 Interaction Design and Usability . . . 43

2.4.3 Evaluation . . . 46

2.5 Summary . . . 48

3 An Empirical Study of Recommendation Dialogue 49
- 3.1 Experimental Design . . . 50
  - 3.1.1 Participants . . . 50
  - 3.1.2 Apparatus . . . 51
  - 3.1.3 Procedure . . . 51
  - 3.1.4 Results . . . 52
- 3.2 Analysis . . . 52
  - 3.2.1 The Dialogue Distilling Process . . . 52
  - 3.2.2 Guideline Development . . . 53
  - 3.2.3 Guideline Application . . . 53
- 3.3 Characterization of Recommendation Dialogue . . . 60
  - 3.3.1 Roles and Dialogue Flow . . . 61
  - 3.3.2 Factual and Preference Requests and Statements . . . 63
  - 3.3.3 Dialogue Acts . . . 64
  - 3.3.4 Delivery and Interview Dialogue Behaviors . . . 68
- 3.4 Summary . . . 72

4 Design and Verification of a Recommendation Dialogue Strategy Model 73
- 4.1 Movie Recommendation Dialogue Control . . . 74
  - 4.1.1 System Initiative . . . 74
  - 4.1.2 User Initiative . . . 81
- 4.2 Evaluation . . . 82
  - 4.2.1 Participants . . . 83
  - 4.2.2 Acorn . . . 83
  - 4.2.3 Procedure . . . 85
- 4.3 Results . . . 86
  - 4.3.1 Dialogue Corpus Analysis . . . 87
  - 4.3.2 User Satisfaction Analysis . . . 90
- 4.4 Discussion . . . 91
- 4.5 Summary . . . 94

5 Storing and Utilizing Situation-Based User Preferences 95
- 5.1 Defining the Preference Model Framework . . . 96
  - 5.1.1 Requirements . . . 96

5.1.2 Dimensions 98
5.1.3 Limitations 98
5.2 PCQL 99
5.2.1 FP State Formulas 100
5.2.2 Factual State Mapping 103
5.2.3 Preference State Mapping 104
5.2.4 PCQL Action Statements 106
5.3 Utilizing and Supporting Dialogue 107
5.3.1 Utilizing Dialogue 109
5.3.2 Supporting Dialogue 111
5.4 The Preflet Construct 113
5.4.1 Definitions 113
5.4.2 Preflet Descriptors 115
5.4.3 Preference Evaluation 116
5.4.4 Constraining and Relaxing Attributes 119
5.4.5 Preference Updates and Conflicts 122
5.4.6 Preflet Update Resolution Heuristics 124
5.5 Summary 127

6 BCORN: Managing Recommendation Dialogue 129
6.1 Dialogue Behavior Diagrams 130
6.2 BCORN Dialogue Behaviors 133
6.2.1 Conditions and PCQL in BCORN 133
6.2.2 PCQL Action Tags and Dialogue Acts 133
6.2.3 Conventional 134
6.2.4 Direct Delivery 136
6.2.5 Interview 137
6.2.6 Indirect Delivery 141
6.3 Emergent Dialogue Strategy 143
6.4 Implementing BCORN: CoreSong 145
6.4.1 Input: Interpretation and Streaming 146
6.4.2 Output: Weaving and Generation 148
6.4.3 Preflet Updates 152
6.5 Evaluation 152

6.5.1 Participants . . . 154
6.5.2 CoreSong Configurations . . . 154
6.5.3 Procedure . . . 158
6.5.4 Results . . . 158
6.6 Summary . . . 159

7 Conclusion 167
7.1 Summary . . . 167
7.2 Future Work . . . 171

A Corpora 189
A.1 Corpus I . . . 189
A.2 Corpus II . . . 190

B Experiment Scenarios 191
B.1 Study I . . . 191
B.1.1 Recommender's Instructions . . . 191
B.1.2 Customer's Instructions . . . 192
B.2 Study II . . . 192
B.2.1 Task 1 . . . 193
B.2.2 Task 2 . . . 193
B.2.3 Task 3 . . . 193
B.2.4 Questionnaire Statements . . . 193
B.3 Study III . . . 194
B.3.1 User Instructions . . . 194
B.3.2 Questionnaire Statements . . . 195

List of Figures

1.1 Sample recommendation dialogue 3
1.2 Dialogue system reference architecture. 4
1.3 Principal work scheme. 7

2.1 Layers in the YMIR architecture. 16
2.2 Preference strength ranges. 31

3.1 Excessive customer input 55
3.2 Distilled version of excessive customer input 56
3.3 Ambiguous reference 57
3.4 Complex preference attribute 58
3.5 Database browsing difficulties in the dialogue 59
3.6 A biased recommender 60
3.7 Sample recommendation dialogue 62
3.8 Information requests in a recommendation dialogue 64
3.9 Accept-Solution conversational circumstance 68
3.10 Constraint selection strategy 70
3.11 Constraint relaxation and delivery 71

4.1 Dialogue control strategy 75
4.2 Exhausted recommendation base 77
4.3 Failure of the RECENGINE node 80
4.4 ACORN's graphical user interface. 84
4.5 Sample movie recommendation dialogue 86
4.6 Scenario deviation and exploration 89
4.7 User satisfaction parameters 90

4.8 Recommendation realization failure 93
4.9 Sample recommendation dialogue 93

5.1 Sample recommendation dialogue 117

6.1 The Conventional DBD. 136
6.2 The Direct Delivery DBD. 137
6.3 The Interview DBD. 139
6.4 The Indirect Delivery DBD. 142
6.5 The standard CoreSong behavior configuration. 145
6.6 CoreSong's graphical user interface. 147
6.7 Sample recommendation dialogue 153
6.8 Sample dialogue for the BLUNT experiment configuration 156
6.9 Sample dialogue for the RECOMMENDER experiment configuration . . 157
6.10 Sample dialogue for the Q-A experiment configuration 157
6.11 Overhearer evaluation web page 161
6.12 Experiment result: Informativeness 162
6.13 Experiment result: Preference Management 163
6.14 Experiment result: Coherence 164
6.15 Experiment result: Naturalness 165

List of Tables

2.1 User model dimensions 25
2.2 Reliability ratings of endorsements 31

3.1 Dialogue distilling guidelines 54
3.2 Utterance content taxonomy 63
3.3 RECOMMENDATION and MOTIVATION dialogue acts. 67

4.1 User evaluation data 88

5.1 Preference model capabilities 97
5.2 Factual and preferential operators 101
5.3 FP state formula mappings (factual) 103
5.4 FP state formula mappings (descriptives and superlatives) 105
5.5 FP state formula mappings (comparatives) 106
5.6 PCQL action mappings 108
5.7 Preference utterance types 109
5.8 Conversational circumstances 110
5.9 Example evaluation scores 119

6.1 BCORN condition functions 134
6.2 System action tags and dialogue acts 135
6.3 User action tags and dialogue acts 135
6.4 Sample direct delivery in the music domain 138
6.5 Sample interview in the music domain 140
6.6 Sample preference interview 141
6.7 Sample indirect delivery dialogue in the music domain 143

6.8 Sample BCORN dialogue 149
6.9 Experiment configurations 155

1

Introduction

This chapter introduces the topic of conversational recommender systems, and presents the aim, research questions, demarcations, and contributions of this thesis.

THIS THESIS DEALS with dialogue strategy design for *conversational recommender systems.* The purpose of a *recommender system* is to produce personalized recommendations of potentially useful items from a large space of possible options that is hard to manually browse or search. In a *conversational* recommender system, this task is approached by utilizing natural language dialogue for detecting user preferences, as well as for providing recommendations. There are several reasons for pursuing the conversational approach as a viable alternative to traditional recommender system interaction. For example, natural language dialogue can allow users to express their preferences qualitatively, and in contexts where they are motivated to elicit them. Furthermore, detecting a user's preferences and using them for recommending items is a *collaborative* venture, where coherent conversation with a dialogue partner seems a natural choice of interaction style. Figure 1.1 exemplifies a recommendation dialogue between a user and a conversational music recommender system. The fundamental idea of a conversational recommender system is that it relies on dialogue sessions to detect, continuously update, and utilize the user's preferences in order to predict the user's potential interest in domain items (e.g. *songs* in the dialogue shown in Figure 1.1) modeled in the system. The design of dialogue strategy management is thus one of the most important tasks when designing conversational recommender systems. In order to place this particular work into a more complete picture, Figure 1.2 sketches a typical conversational dialogue system conceptual architecture, and shows where the focus of this thesis lies. In this work, the aim is thus to discover generic solutions for modeling conversational recommendation behavior, that are suitable for implementing in specific recommender system applications. The work is carried out in three main steps. First, we define a recommendation dialogue situation, from which we empirically study and characterize recommendation dialogue. Then, based on a verified empirical model, we construct and verify a generic model and software artifacts of recommendation dialogue that can be used in conversational recommender system implementations. Verification of these components is then done with an end-user evaluation of a conversational recommender system implementation.

1.1 Aim and Research Questions

Since a conversational recommender system relies on dialogue and incremental modification of a user's preference model, the system's *dialogue strategy* to initialize, update, and utilize the preference model in the interaction is crucial for its performance and usability. The overall aim of this thesis is therefore to describe and present a computational dialogue strategy model for personalized conversational recommender systems. In order to pursue this, the following specific research questions are posed:

1. How are recommendation dialogues characterized?
2. What is the structure of a recommender agent's dialogue strategy?
3. How are user preferences stored, utilized, and updated in recommendation dialogue?

S	Welcome. What situation should we discuss?
U	I would like a play list for work
S	Ok. Is there a particular genre you prefer when you work?
U	I like *jazz*
S	These artists and albums belong to the genre *Jazz*: `[list]` Any of these you like in particular?
U	What *rock* artists are there?
S	These artists belong to the genre *Rock*: `[list]` Any of these you like in particular?
U	*Eric Clapton* is the best!
S	These albums belong to the artist *Eric Clapton*: `[list]` Any of these you like in particular?
U	I like *Journeyman* better than *Pilgrim*
S	Ok. Are there any other genres, artists or albums you prefer at work?
U	What albums by *Elvis Presley* are there?
S	These are the albums by the artist *Elvis Presley*: `[list]` Any of these you like in particular?
U	I don't like his albums released after *Promised Land*
S	Ok. I think you would like *Tales of Brave Ulysses* by *Cream.* Have you heard it?

Figure 1.1: Sample recommendation dialogue between a conversational recommender system (S) and a user (U). `[list]` denotes a listing of database items, such as genres, artists, or songs.

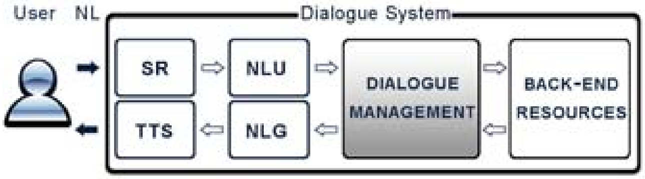


Figure 1.2: Prototypical dialogue system reference architecture. Even though complete implementations guide this work and involves all components above, the focus of this thesis' research questions is on dialogue strategy management (shaded in the figure). NL = Natural Language, SR = Speech Recognition, NLU = Natural Language Understanding, NLG = Natural Language Generation, TTS = Text-to-Speech.

4. What is required for a computational model for recommendation dialogue and user preference management?

1.2 Contributions

The thesis is restricted to single-domain, unimodal dialogue systems, that have as their purpose to deliver personalized recommendations to users. That is, the model does not cover switching between different domains in the same dialogue session, and we utilize natural language input as the only input channel[1]. Furthermore, we assume a *strict* turn-taking mechanism[2]. In particular, the focus is on conversational recommender systems in *media domains*, such as music and movie recommendations.

Given these demarcations, the contributions should be viewed as studies from a cross-disciplinary perspective. That is, the three first research questions are problem-oriented and expected to help us describe and understand the complex problem space at hand. While reaching the answers to these questions in a hypothesis-generating manner, we are ready to investigate the fourth research question, where we can expect more tangible components engineered to encapsulate the empirical findings previously

---

[1] Even though the systems described use standard graphical user interfaces, on-screen direct manipulation is not part of the dialogue management model per se.

[2] The framework as such, can be implemented as parallel Harel statecharts, effectively setting the scene for utilizing temporal constructs such as time-outs and delays, and thus more dynamic turn-taking. However, the focus and claims made within this work are made on the strict turn-taking assumption.

discovered.

In summary, this thesis makes the following contributions:

1. Characterization of recommendation dialogues based on collected human-human recommendation dialogues.
2. PCQL: A formalism for describing preferential and factual statements and requests, as well as for supporting management of preferences in conversational recommender systems.
3. Preflets: A preference model that stores situation-dependent user preferences, and supports recommendation dialogue.
4. Dialogue Behavior Diagrams: A precise description of conversational recommendation dialogue as modularized behaviors.
5. BCORN: A computational model of recommendation dialogue, based on dialogue behavior diagram instances. The model is implemented in the CoreSong conversational recommender system, and has been evaluated to verify the BCORN model.

1.3 Method

Language Technology[3] is a multi-disciplinary research field that combines expertise from the humanities, natural and behavioral sciences. Scientific approaches and practical techniques are drawn from a range of disciplines, including Linguistics, Computer Science, Human-Computer Interaction, Engineering Science, Psychology, and Mathematics.

In this thesis, the focus is to provide a computational model of recommendation dialogue. Therefore, the model is not psychological or “human-like” from an internal perspective. It is, however, “human-like” (with limitations and modifications that are clarified later) and empirically studied and verified as such, from an external, behavior-based, viewpoint. Due to the cross-disciplinary approach, and the inherent

---

[3] The related terms Computational Linguistics, Natural Language Processing, and Language Engineering (and possibly others) all refer to some aspect of the computational study of language. No distinction between them is made in the following, unless specifically noted.

properties of the phenomenon of dialogue interaction itself (i.e. human users with expectations on natural language use in human-machine dialogue situations), dialogue system research in general includes (exploratory) design methodology, formal specification, and empirical validation [Hulstijn, 2000]. It is therefore difficult to choose a single methodological position for this venture. Both exploratory and empirical studies have been carried out in this work, as well as software development work. The exploratory approach has been used to gain insights on the constitution of recommendation dialogue in a human-computer use context. This includes dialogue corpus collection, and qualitative corpus analysis in order to generate hypotheses concerning dialogue characterization and design. Empirical studies—in the behavioral, human-computer interaction (HCI) sense—has been chosen for end-user prototype evaluations, primarily to verify dialogue model designs. This includes analysis of questionnaires and dialogue session logs. The aim of software development research in general is to gain insights into the mechanisms and pre-conditions under which certain phenomena can be modeled and/or emerge. Although much progress has been made within this approach, the integration with results from other scientific disciplines often has weak explicit linkage to empirically derived knowledge. This is addressed in this work by mapping empirical findings to software component design as an application of empirical research [Jönsson, 1993]. Development is grounded in an engineering approach, where the focus is on functionality and robustness of software, work effort connected to the construction process, and reason about development methods from the perspective of usefulness and efficiency for programmers.

The viewpoint in this work is thus that the way forward is to cross-fertilize an engineering approach with empirical HCI studies in order to further advance the field of dialogue system research as a whole. By illuminating the stated research issues from this hybrid approach [Hulstijn, 2000] a number of interesting answers can be provided, and we will view the contributions of the thesis in the light of this.

More concretely, Figure 1.3 shows the work scheme that has been used in order to address the research questions of this thesis, and the results that will be described in the following chapters. Main work activities include Studies I–III, which are empirical studies that each results in various "artifacts". Study I results in a corpus and recommendation dialogue characteristics; Study II is an evaluation of a dialogue strategy, as well as an exploration with implications for refinements and

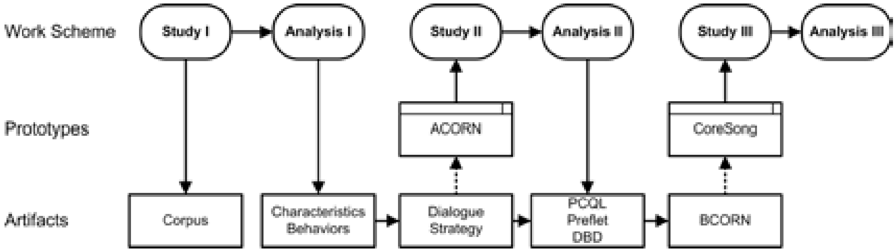


Figure 1.3: Principal work scheme of this thesis, showing important work activities and resulting contributions.

customized notations and frameworks; and Study III is an evaluation and verification of the artifacts arrived at during the course of this work. The artifacts shown in Figure 1.3 constitute the main contributions of this thesis. Prototyping suits an iterative, exploratory development process [Jacobson *et al.*, 1992], and is recommended in order to make a system that meets users' needs [Nielsen, 1993]. During the course of this work, several prototypes have been built to aid the exploration and investigation of conversational recommender system design and development in film and TV media domains (see [Johansson *et al.*, 2002; Johansson, 2003b; 2003a; Berglund and Johansson, 2004]). Some of these system prototypes are not explicitly covered in this dissertation. However, two of them—ACORN [Wärnestål, 2005] and CORESONG [Wärnestål *et al.*, 2007b]—are used in the controlled experiments Study II and Study III and are therefore included as part of the work scheme as shown in Figure 1.3.

1.4 Thesis Outline

The rest of the thesis' chapters and content are organized as follows.

- **Chapter 2. Background: Conversational Recommender Systems**. As a starting point for the work, this chapter provides definitions of relevant concepts, classifications of (general) dialogue management approaches, user preference modeling and recommender system research. We also present aspects of the design and development of conversational recommender systems.

- **Chapter 3. An Empirical Study of Recommendation Dialogue**. The empirical basis for this thesis is a recommendation dialogue corpus, collected in a dialogue study (Study I). This results in a human-human dialogue corpus, which is systematically re-written into a human-computer recommendation dialogue through the process of *dialogue distilling*. The analysis of the material is presented as a characterization of the recommendation dialogue genre.

- **Chapter 4. Design and Verification of a Recommendation Dialogue Strategy Model**. This chapter first presents a basic dialogue strategy model for movie recommendation dialogue, based on the analysis of the previous chapter, that is implemented in a prototype conversational recommender system called ACORN. The chapter then describes an evaluation of ACORN's dialogue strategy model with end users (Study II). We assess a range of usability metrics on the recommendation dialogue strategy based on Study I. We thus verify that the distilled dialogue that forms the model is both effective, efficient, and usable.

- **Chapter 5. Storing and Utilizing Situation-Based User Preferences**. In order to provide personalized recommendations to a user, the system needs a model of the user's preferences. The topic of Chapter 5 is a situation-based user preference model called *preflet*, that supports natural language dialogue management and provides user preference data for content-based recommender engines. This chapter also introduces the PCQL data manipulation notation for expressing preferences and factual queries and statements, which will be used as a generic message-passing formalism in the BCORN model.

- **Chapter 6. BCORN: Managing Recommendation Dialogue**. In this chapter the concept of dialogue behavior diagrams (DBDs) is introduced. DBDs are used to describe a generic recommender agent's dialogue strategy, based on the dialogue behaviors defined in Chapter 3 and the implications from Chapter 4. DBD instances update and utilize preflets for managing the personalized recommendation dialogue, and utilize various external resources such as databases and recommender engines. The presented DBDs constitute the computational model BCORN (Behavior-based Conversational Recommender). The chapter also presents CoreSong, which is a functional conversational recommender system in the music domain, implemented as an application of the bcorn model. CoreSong is validated in an overhearer user study (Study III), which is also presented.

- **Chapter 7. Conclusion**. The last chapter first provides a summary and concluding discussion of the previous chapters. The results of the thesis are discussed, and pointers for future research in the area are provided.

2

Background: Conversational Recommender Systems

This chapter surveys state-of-the-art research of dialogue systems and dialogue models, user preference modeling, conversational recommender systems, and relevant software engineering methodologies.

In this introductory chapter, several research areas are briefly surveyed. The common denominator is a conversational approach to personalized recommender system design and interaction. This requires us to examine four parts. First, we look at dialogue system concepts in general (Section 2.1), where some basic definitions and components are surveyed, before moving on to (some) control architecture and formalism constructs. Second, we address user modeling and user preference management in general (Section 2.2). The usage of user models is also taken into account when we examine recommendation models and algorithms. Armed with dialogue systems, user preference models, and recommendation models we are ready to survey existing approaches to conversational recommender systems (Section 2.3). Finally, some aspects of the engineering of conversational recommender systems are covered by surveying development methodologies, interaction design issues, and the particularities of personalized dialogue system evaluation (Section 2.4).

2.1 Dialogue Systems

Advances made in the fields of language technology has provided a range of models aiming at producing “the conversational computer” [McTear, 2002]. This section describes definitions of and motivations for dialogue system concepts, a classification of dialogue modeling approaches, and the general components required for dialogue systems.

2.1.1 Concepts and Definitions

We define a *dialogue system* to be a computer system with which human users interact on a turn-by-turn basis, and in which natural language is an important interaction modality. The purpose of a dialogue system is to provide a natural-language interface between a user and a computer-based application [McTear, 2002]. The underlying goal is to enable users to interact in a natural and intuitive way using spoken or typed natural language. However, since general human-like conversational competence is still far beyond today’s models and technology, the naturalness of the interaction is questionable. Dialogue systems of today limit users in what they can say and how. Intuitive and natural interaction is then a case of design, and the human-computer dialogue interaction needs to be shaped accordingly, utilizing the weak and strong points of human-computer dialogue, which is significantly different from human-human dialogue [Jönsson and Dahlbäck, 1988].

One demarcation that has been proposed to this issue is to focus on *practical dialogues* [Allen *et al.*, 2000; 2001]. The practical dialogue hypothesis states that [Allen *et al.*, 2001, page 29]:

> The conversational competence required for practical dialogues, while still complex, is significantly simpler to achieve than general human conversational competence.

By applying the practical dialogue approach in well-defined domains, a robust and usable conversational behavior can be achieved despite the shortcomings of today's natural language understanding components and models [Pieraccini and Huerta, 2005]. The design of a proper dialogue management model is thus one of the most important tasks of dialogue system research. It is worth designing generic dialogue management solutions, following the *domain-independence hypothesis*. It states that [Allen *et al.*, 2001, page 29]:

> Within the genre of practical dialogue, the bulk of the complexity in language interpretation and dialogue management is independent of the task being performed.

To this end several generic architecture and tool box approaches that provide implementations of dialogue models have been suggested for new-domain dialogue system customization [Allen *et al.*, 2000; O'Neill and McTear, 2000; Degerstedt and Jönsson, 2004; Larsson and Traum, 2000].

The core of a typical dialogue system architecture is the *dialogue manager*. Even though there is no universal agreement of exactly what a dialogue manager is, and what capabilities it should have, there seems to be two common denominators [Pieraccini and Huerta, 2005]. A dialogue manager should: (a) keep track of session states, and (b) decide on the next system action. These aspects can then be completed with e.g. contextual interpretation, ambiguity resolution, anaphora resolution, system back-end communication, output generation, etc.

The dialogue manager is responsible for two aspects of dialogue control: management of initiative, and management of dialogue flow. Initiative may be on the system, the user, or mixed. Simple dialogue system approaches adopt completely system-driven dialogue control. That is, the user is guided through a number of pre-defined system prompts until the task is complete. Typically, such systems are implemented in a finite-state based fashion. A larger degree of user freedom exists in frame based approaches. (See Section 2.1.4 for details.) In user-driven dialogue, the system reacts to user queries and delivers the desired information. More complex dialogue system allows for mixed initiative, where dialogue control is shared. The user is free to e.g. ask questions at any time, and the system may have an agenda or plan and issue questions/prompts to the user accordingly. Plan (or agent) based systems (see Section 2.1.4) are typically mixed initiative.

2.1.2 Dialogue Model Components

Dialogue managers utilize *knowledge sources* which, according to McTear [2002], collectively is referred to as the *dialogue model*. For the purpose of this thesis, we accept this notion of a conversational system's dialogue model. Some fundamental knowledge sources include [McTear, 2002, page 96]:

- **Dialogue history.** A representation of (relevant aspects of) the dialogue in terms of mentioned domain entities and topics. This resource forms the basis for anaphora resolution, and focus management.
- **Task model.** A representation of the user's current task and the information required to be gathered in the dialogue.
- **World knowledge model.** A representation of general background information to support reasoning (e.g. temporal knowledge).
- **Domain model.** A representation of domain-specific knowledge including ontological relationships, etc.
- **Conversational competence model.** A representation of turn-taking, discourse obligations (e.g. Grice's maxims [1975]), etc.
- **User model.** A representation of user characteristics relevant to the dialogue.

All knowledge sources listed above can be more or less complex, depending on the requirements of the system application. The view in this thesis is thus that dialogue complexity exists on several levels and can be described from different perspectives.

2.1.3 Layered Control Architectures

One important question when designing and implementing dialogue systems is which architecture to use. During the past decades several architectures have been proposed; all with their own advantages and disadvantages. Most architectures are not formally specified. This means that there is a degree of freedom concerning how an architecture is interpreted. However, the need for a formalization cannot be escaped when implementing an actual system (since a verbal description cannot be compiled and run on a computer). An important issue thus becomes; given an (informal) architecture description—which formalism should be chosen for implementing the architecture? This survey will focus on layered architecture approaches, since this relates to the contributions made in this thesis.

A general characteristic of layered systems is that they are organized hierarchically. Each layer provides service or information to the layer above it and serves as a client to the layer below. The connection between layers are defined by protocols, determining how layers will interact [Garlan and Shaw, 1993]. Outside the world of Language Technology, the most widely used applications of layers are (a) layered communication protocols such as the TCP/IP stack, and (b) operating systems, where the kernel runs in one layer, and user space and processes reside in layers above.

According to Garlan and Shaw [1993, page 11], layered systems support the following properties:

1. design based on different levels of abstraction (i.e. supporting a divide-and-conquer strategy of partitioning complex problems into incremental and manageable steps).

2. enhancement (since each layer only interacts with *at most* the surrounding two layers, changes in one layer affect at most two other layers).

3. reuse (i.e. different implementations of a layer can be used interchangeably, allowing for designs of reusable layer interface standards in a framework).

There are some disadvantages one must be aware of, and not all kinds of systems are suitable for a layered approach. *Performance* considerations need to be taken, even if a system can be structured in layers. That is, a system may require close coupling between low-level and high-level layers due to computational performance, such as large search tasks etc. This may thus violate the *enhancement* property above. *Design* considerations are also important, since some systems conceptually are hard to model in this manner. Finding the right level of abstraction for the different layers is a non-trivial design issue. For example, communications layers sometimes need to bridge several layers—an unhealthy sign since this violates both the *enhancement*

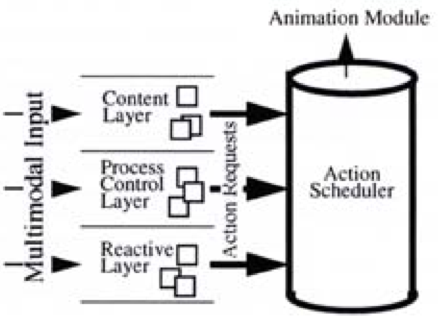


Figure 2.1: Principal layers and components of the YMIR architecture. From Thórisson [1997].

and the *reuse* advantages mentioned above. It may even violate the *design* property depending on the complexity of the problem the system aims to solve.

Layered Dialogue System Approaches

The layered and behavior-based approach to dialogue system construction is limited. This ties into the observation that the notion of layers can mean different things. Even though the general—software engineering—notion of layers have been used in several architectures adhering to the classical paradigm, implementations of a more dynamic behavior-based layer approach are rare. This section briefly describes two such approaches.

Reactive, Process Control and Content Layers The YMIR system [Thórisson, 1997] employs a layered architecture that consists of three layers as shown in Figure 2.1. It is focused on communication and models speech, intonation, body language and facial gesture. The three layers are: reactive, process control, and content. Each layer contains perception and decision modules. Response times are very important in natural language communication, as a pause or delay can change the entire meaning of an utterance depending on context. The reactive layer has very short response times (in the 150–500 ms range) in order to provide realistic gaze fixation or blinking etc. Processing time increases in the higher layers. The top-most layer—the content layer—have significantly longer response times, and consists of topic information in knowledge bases. The output of the layers is processed by a fourth main component, the Action Scheduler, which prioritizes and morphs action requests from the three layers.

Implementations of YMIR display a set of characteristics that are not usually found in dialogue systems:

- there is a non-rigid interruptible quality of the behaviors[1].
- gestures and body language (including facial gestures) are integrated with the communication content, without artificial communication protocols.
- behaviors run concurrently at the expected times and without unnatural delays.
- miscommunication and speech overlaps are handled in a "natural" way, by using stops and restarts.

Recently, the YMIR architecture has been enhanced with a more elaborate turn-taking model [Thórisson, 2002]. This work addresses the fact that each participant in an ordinary dialogue may take 2–3 communication decisions every second. One interesting aspect of this work is that it assumes no protocol, or implementation of turn-taking *rules*.

Content and Interaction Layers Another example of a multi-layer approach to spoken dialogue systems is presented by Lemon *et al.* [2003]. This work is inspired by (a) communication layers as presented by Clark [1996], and (b) robot architectures as outlined in the behavior-based AI field. As Lemon *et al.* put it [2003, page 169]:

> We view the process of natural interaction with a dialogue participant as analogous to the interaction with a dynamic environment: dialogue phenomena arise which need to be negotiated (as a new obstacle must be avoided by a robot).

The proposal is a two-layer architecture that separates interaction-level phenomena from content and context management and conversation planning. The main benefit

---

[1] Note that this is different from barge-in, which is a general way to just skip the reminder of a system utterance.

of this approach is that the interaction layer makes use of low-level signals to provide more natural and robust interactions between dialogue systems and human participants in a way that the content layer is not able to do solely. This is supported by a number of findings in psycholinguistic literature that stresses the importance of asynchronous interaction-level processing in realistic and natural dialogue. Examples of interaction-level phenomena handled by systems implementing this architecture include: realistic timing and turn-taking, immediate grounding and thus more realistic (continuous) feedback, barge-in management, and NP selection (anaphora management). Unfortunately, no user evaluation data is available yet, but the approach nevertheless seems promising for end-users since it addresses issues that are mostly not focused on with most other dialogue systems.

From an engineering point of view, the layered approach to dialogue system construction displays some desirable characteristics: According to Lemon *et al.*:

> In engineering terms, this division of labor is attractive in that the clarity and modularity of dialogue management is enhanced

However, some approaches in this strand have failed. Note that e.g. Steels' non-hierarchical, massively parallel approach [Steels, 1994] was abandoned for *engineering reasons* since it was not practical for each behavior to model all other behaviors, even though the theory seemed sound from a biological perspective [Bryson, 2001].

Subsumption Layer Control

We will end the survey of layered control architectures with the subsumption architecture, somewhat radically suggested by Rodney Brooks. By organizing behaviors as layers, Brooks [1991b; 1991a] created an architecture (and methodology) called the *subsumption* architecture for designing and implementing robots with multiple goals and robust behavior. A crucial difference with this approach compared to those of e.g. Thórisson and Lemon *et al.* is that a particular behavior[2] does not model other

---

[2] In the following, a behavior is an informal term denoting a result of system actions in a context, whereas a layer is a technical term that refers to internal structures used to generate behaviors (in the former sense).

behaviors. It can instead be described as a loosely coupled approach without centralized control. Both the YMIR model and that of Lemon *et al.* are inspired by the behavior-based philosophy—even if they do not go "all the way" as they rely on high-level planning in one centralized control layer. Close in spirit, the SESAME generic dialogue management framework, is designed for supporting dynamic multi-domain dialogue processing [Pakucs, 2003]. It's architecture is agent-based and agents communicate via a blackboard. However, none of the autonomous agents has access to a central representation of the problem, and there is no central agent responsible for planning. Furthermore, none of the agents knows about other agents' internal states.

An important feature in the subsumption approach is incremental extendability, i.e. that new layers are built on top of existing behaviors, effectively supporting evolutionary design and development of system capabilities.

Briefly, subsumption means that a system's control architecture is decomposed into sets of layers, each implementing a behavior of the system. The approach stresses sensor-motor coupling and is a rather radical reaction to the classical AI paradigm of functional decomposition, since layers run independently of each other and without a model of the environment/world. One central hypothesis is that coherent behavior can be achieved without explicit central modeling, but as an emergent result from micro behaviors [Pfeiffer and Scheier, 1999].

High-level cognition (including conceptual natural language and dialogue capabilities) in the system is not considered in Brooks' early papers (the first paper on subsumption appeared in 1986), and the approach is only validated for simplistic behavior. Despite of this, the approach is still of high interest and has been the subject of general discussions on embodied cognitive science and general artificial intelligence [Kirsh, 1991; Pfeiffer and Scheier, 1999], which has also been replied to by Brooks in his later work, e.g. [Brooks, 1997].

2.1.4 Computational Dialogue Management Formalisms and Classifications

Traditionally, dialogue systems are roughly divided into three[3] classes based on the type of dialogue management formalism employed [McTear, 2002].

1. Finite-state based systems
2. Frame based systems
3. Plan (or Agent) based systems

Note that this classification refers to the system as a whole; that is, aspects of both conceptual architecture and mathematical formalism used to implement the architecture influence how a system is classified.

Finite-State Based Systems

At the least complex end the dialogue system steps through a dialogue consisting of pre-defined states. The interaction is completely system-driven, and has been widely used in commercial (telephone-based) applications. The advantage is the simplicity of the approach. Due to the inflexibility of pre-defined states and the purely system-driven initiative, the interaction is very restricted and only suitable for very simple tasks such as long-distance dialing by voice or simple telephone banking.

Frame Based Systems

Frame (or template) based systems have more flexibility and allow users to decide the order in which information required by the task record is given. The frame is in this case viewed as a static context representation where a pre-defined set of parameters needs instantiation before a system action can be taken. In contrast to finite-state based approaches, several pieces of information can often be given in one go, if the user desires it. Examples include travel bookings, hotel reservations, etc.

---

[3] Allen *et al.* [2001] has a slightly more fine-grained division, but for the purpose of this survey, McTear's division is sufficient. Furthermore, we will not deal with probabilistic or machine learning approaches to dialogue management in this survey.

Since task representations are simple and well-understood, complete linguistic interpretation of user utterances is often not necessary. Therefore, robustness can be achieved in the natural language processing components [Allen *et al.*, 2001; McTear, 2002].

Frame based systems can be extended with multiple context representations, thus adding complexity in the dialogue management since the user can shift between topics in one dialogue session; for instance, discussing hotel location (with a frame task representation covering aspects such as room size, price, and various hotel facilities) and flight tickets (with a frame covering seating, number of stops, and ticket price) in the same travel booking dialogue session.

As is the case of any production rule system, a potential disadvantage for frame based development is that it is difficult for the designer to foresee all contexts under which rules may fire. Examples of frame based systems include the VOICEXML framework, and MIMIC [Chu-Carroll, 2000].

Plan Based Systems

At the higher end on the task complexity scale we find plan based (a.k.a. agent based) systems. These are designed to permit complex communication about tasks that are too complex to be represented as pre-defined frames. Dialogue phenomena that can be handled include collaborative negotiation dialogues, dynamically generated topic structures [Allen *et al.*, 2001], and advanced error detection and correction [McTear, 2002]. Since the user's input cannot be determined in advance, a sophisticated natural language interpretation component is required for agent based systems.

The plan based approach is rooted in classical AI and is related to the logic based approach [Hulstijn, 2000]. In this approach, dialogue management is viewed as a general inference engine and dialogue is represented in a logical formalism.

Plan and logic based approaches are often theoretical and implementations are few and to this date restricted to research systems, such as TRAINS and TRIPS [Allen *et al.*, 2001] and SUNDIAL [McGlashan *et al.*, 1992]. This approach also has potential disadvantages [Jönsson, 1997]: First, the problem of identifying the primitives needed for describing human intention and goals. Secondly, the concern of performance of plan recognizers.

2.1.5 Other Dialogue Classification Schemes

Categorizing dialogue can be done on other dimensions than the ones described above. This section will briefly summarize some attempts.

Dahlbäck's classification [1997] has a slightly different goal than that of Allen *et al.*, and lists the following dimensions (or criteria) for dialogues:

- Modality (spoken or written)
- Kinds of agents (human or computer)
- Interaction (dialogue or monologue)
- Context (spatial and/or temporal)
- Number and types of tasks
- Dialogue–Task distance
- Kinds of shared knowledge (perceptual and/or linguistic and/or cultural)

For the purpose of this thesis, it is interesting to note the *dialogue-task distance* dimension. Dahlbäck observes that the dialogue structure can bear close or distant resemblance to the task structure. A short distance example is planning and advisory dialogue, whereas information retrieval dialogue is a long distance example. Short distance dialogue benefit from intention-based methods according to Dahlbäck [1997, page 36]:

> The closer the language background task connection the more appropriate become plan or intention based models

Another classification is made by Hulstijn [2000]. Hulstijn defines *inquiry-oriented* dialogue as an exchange of information between two participants in different roles: an inquirer and an expert, where the inquirer has an information need, and the goal of asking the expert questions to satisfy that need. The expert's goal is to answer questions—as well as asking clarifying questions when needed.

Along with inquiry-oriented dialogue Hulstijn suggests *negotiative* dialogue. Negotiative dialogue refers to dialogues where participants can discuss and compare alternative solutions to the problem at hand. (Non-negotiative dialogue typically concerns simple information retrieval tasks, such as natural language interfaces to databases.) He describes five phases in negotiative dialogue in the default order *open*, *inform*, *propose*, *confirm*, and *close*. Except for the opening and closing phases, any result of a phase can, however, be undone.

This concludes the first part of the introductory overview of related research. In the following section, general user modeling research and recommender systems are surveyed, before relating it to the conversational approach to recommender systems.

2.2 User Modeling and User Preferences

User modeling is a large research field with several applications. This survey briefly reports on the fundamentals of user modeling (Section 2.2.1), before focusing on one particular aspect of user modeling: user preference modeling (Section 2.2.2).

2.2.1 User Modeling

To lay a basic foundation for this overview we start off with an intuitive definition of a user model, i.e. that a user model is knowledge about the user of a system, encoded for the purpose to improve the interaction. Kass and Finin [1988] view user models as a subclass of *agent models*. An agent model is a model of any entity, regardless of its relation to the system doing the modeling. A user model is thus a model of the agent currently interacting with the system. Furthermore, Kass and Finin note that *implicit* user models are often not interesting, since they merely represent assumptions about the user made by designers of the system (at design-time). This discussion is instead focused on *explicit* agent models, which are constructed and utilized at use-time. There are four features that characterize agent (and thus, user) models [Kass and Finin, 1988, page 6]:

1. **Separate Knowledge Base.** Information about an agent is not distributed throughout other system components.

2. **Explicit Representation.** The knowledge about the agent is encoded in an expressive language, with support for inferential services.

3. **Support for Abstraction.** The modeling system can distinguish between abstract and concrete entities, such as classes and instances of objects.

4. **Multiple Uses.** The agent model can be used for various purposes such as support dialog, or to calculate predictions of items for a user, etc.

Since the user model concept is approached from different directions, it is multi-faceted and can be categorized along the lines of several dimensions. One of the earliest works in the field identifies three dimensions [Rich, 1979], and other authors add to this list. Kass and Finin [1988] summarize these dimensions (D) as shown in Table 2.1. It is possible to imagine more dimensions of a user model. Zukerman and Litman [2001] for example, present the concept of multi-dimensional user models. While their use of "dimension" is not directly comparable to that of Kass and Finin[4], it gives rise to the concept of interaction modality, which can be viewed as an addition to the list above. By including modalities other than language, such as mouse pointing and clicking etc., in the user model the system's "understanding" of what the user is trying to accomplish can be enhanced.

Advantages

Billsus and Pazzani claim that the *information overload* problem could be overcome by user modeling in the context of information agents [Billsus and Pazzani, 2000, page 148]:

> [User modeling systems] locate and retrieve information with respect to users' individual preferences. As intelligent information agents aim to automatically adapt to individual users, the development of appropriate user modeling techniques is of central importance.

Such user modeling systems are known as *recommender systems* and make use of user models consisting of user *preferences*, detailed in Section 2.2.2.

The benefits of user modeling systems are [Sparck Jones, 1989; Kay, 2000]:

---

[4] Rather, Zukerman and Litman's use of "dimension" seems to be more related to Kass and Finin's *number of models* dimension.

Table 2.1: User model dimensions according to Kass and Finin's classification [1988].

	Dimension	Meaning
D1	**Specialization**	The user model may be *generic* or *individual*. Typically, a stereotype model [Rich, 1989] can act as a "bridge" between a generic and an individual model. For the class of personalized recommender systems, the user model is naturally individual.
D2	**Temporal extent**	The dimension of temporal extent is defined on a *short-term* – *long-term* scale. A short-term user model is discarded as soon as the interaction ends. Long-term models are stored between sessions, and are suitable for individual models.
D3	**Modifiability**	If the user model can be changed during the course of an interaction, it is *dynamic*. Otherwise, it is *static*. User models that continuously track goals and plans of the user during a session are dynamic.
D4	**Number of agents**	Some systems are not limited to a one-to-one relationship between user and system. There might be several agents involved in the interaction, such as in a medical diagnosis system where there is one doctor interacting with the system, and one patient. Both the doctor and the patient can be modeled in separate agent models. The system could also have a model of itself.
D5	**Number of models**	For each given agent, it is possible to have several models. Separate models for an individual agent corresponds to real-life situations where humans can "wear different hats" depending on if they act as a private person, or represent a company etc. Kass and Finin claim that there has to be a central model responsible for deciding which sub-model to employ in any given situation.
D6	**Method of use**	User models may be *descriptive* (i.e. described in a simple database which can be queried), or *prescriptive* (i.e. where the system simulates the user to check the user's interpretation of a system response).

- **Effectiveness.** The prime object of the user model is that the system reaches the correct decision. A correct user model is thought to help the system improve its decision-making.

- **Efficiency.** A user model can also serve to reach the correct decision in an economical way.

- **Acceptability.** The system may utilize a user model to support its decision-making in a comprehensible and agreeable way.

Disadvantages

The lack of utility evaluations of user modeling and personalized interaction points out one of the problems with user modeling and personalized systems; the tendency to non-determinism. That is, the interface and the available commands may differ depending not only on who the user is; but could also differ for the same user at different times depending on the task she is currently attending. This is really walking on the edge in terms of usability, since it is very close to violate established usability principles, such as "recognition rather than recall", and the principle of making things visible [Norman, 1988].

Höök *et al.* [1996] point out that adaptive systems run the risk of leaving the user without a sense of control. It is necessary for all software systems that they are inspectable, controllable and predictable. This can be addressed by *transparent* systems. Transparency occurs when the system is built as a "glass box" (i.e. the user is informed of the user model and its use in the system). Fischer [2001, page 14] claims that:

> it will be a major challenge to find ways to avoid misuses, either by not allowing companies to collect this information[5] at all or by finding ways that the individual users have control over these user models.

In order to avoid some of these ethical and social pitfalls, Kobsa [Kobsa, 1993] provides the following guidelines:

---

[5] i.e. "user models". (Author's note).

- Users should be aware of the fact that the system contains a user modeling component.
- Users should be instructed that computer systems make errors, and merely relies on assumptions.
- Users should be instructed that a system might pursue non-cooperative interests.
- Users should have the possibility to inspect and modify their user model.
- If technically possible, users should be able to switch off the user modeling component.
- Long-term characteristics should be modeled with caution, since misuse is more likely to have a larger effect, and because they are often more intimate/personal than short-term characteristics.
- Results in user modeling research should be made accessible to the general public, since they will eventually be affected by it.

Systems following these guidelines would have an "interactive user model", and thus be adaptable.

A related problem, which could yield negative social implications, is the issue of incorrect models. A recommending system, such as a TV guide, with an incorrect user profile could start recommending TV shows that a user is not interested in—or worse: would not want to be affiliated with. In some contexts (e.g. watching TV with friends), such recommendations could result in a social faux pas.

2.2.2 User Preferences

One may argue that preferences could be treated as goals. However, while there are situations in which the two seem related, Carberry *et al.* [1999] suggest that goals are conscious at the beginning of an interaction, whereas preferences come into play as users must evaluate alternatives and choose between them. Preferences influence the way users desire to achieve their goals. Satisfying strong preferences is more important than satisfying weak ones, and partially satisfying preferences are better than ignoring them; even if the underlying goal can be reached in both cases. Therefore, preferences are considered to be influencers on the actions that need to be taken in order to reach a goal. This has implications on how and when preferences should be requested [Carberry *et al.*, 1999, page 189]:

> *Dynamically* recognizing user preferences as they are expressed in dialogue, rather than querying the user about those preferences beforehand, may be the only way to obtain an accurate model of the user's preferences [...] The user may not have an accurate model of her preferences until she is faced with an actual decision, so querying her preferences before a decision is made will be ineffectual.

According to Kay [2000], four principal techniques for acquiring user preference data exist. The techniques listed are at a rather abstract level and says little of *how* the actual acquisition can be implemented in a system. However, they serve the purpose of providing a framework for continuing the discussion. The four techniques are:

- **Elicitation.** Elicitation is a straight-forward method of simply asking the user. The quality of the data is quite high, but the drawback is that it demands the user's attention, possibly hindering her from focus on the task at hand.

- **Monitoring.** This unobtrusive method demands no attention from the user's part, since it resides in the background. However, the data is typically of low quality and can never be better than a guess.

- **Stereotypic reasoning.** A stereotype is one the oldest and most common elements in user modeling work. This approach is mostly used to quickly assess default values about a user, which inferential reasoning can be based on. Disadvantages include that stereotypes by definition are nothing more than rough guesses about individual characteristics, and that it requires a significant amount of knowledge engineering to chart relevant stereotypes and related implications for a domain.

- **Domain- or knowledge-based reasoning.** This approach is related to stereotypic reasoning and relies on inferences drawn from some sort of domain model or ontological relations. For example, a user indicating that she knows nothing about concept *C2*, allows the system to infer that she is also ignorant of concept *C1*, if it follows from the knowledge model of domain *D* that *C1* is a prerequisite for *C2*.

Obviously, taking all possible techniques for acquiring user preference data into account is a huge task, considering e.g. the range of available interaction modalities that can be viewed as orthogonal to the techniques listed above. For the purpose of this thesis, we focus on the approach of elicitation and domain- and knowledge-based reasoning.

There are several ways of encoding preference data. Scales (e.g. 1–5, or "Bad–Ok–Good", etc.) are used in several commercial and research systems. In a natural language approach one runs into the problem of quantifying preferential statements and mapping them onto a chosen scale. This issue has been addressed by employing machine learning techniques (cf. [Pang *et al.*, 2002]), and lies beyond the scope of this thesis. An alternative approach is to differentiate between positive and negative preferences, and focus on preference strength and reliability measures. This is the approach of Carberry *et al.* [1999].

Carberry *et al.* claim that since people often weigh their preferences when making a decision, some preferences may be more important than others. Thus, *preference strength* is an important aspect of preference modeling. The model presented by Carberry *et al.* dynamically recognizes user preferences in planning dialogues. A complete preference representation contains an attribute–value pair according to a domain description, a strength of the preference, and the system's confidence in the preference (*reliability*).

Preference strength

Preference strength modeling depends on (a) semantics of the user's utterance, and (b) the dialogue context in which preferences are given. The utterance types considered are:

1. **Direct.** Directly expressed preferences (such as *"I like Language Technology"* represent the strongest type of preference.

2. **Indirect.** Some utterances contain implicit preferences. That is, they are not direct statements, but rather express a potential interest. An information query is an example of this. Indirect preferences have weaker strength than direct preferences.

3. **Hedging.** Uncertain preferences are conveyed by using "hedging" that introduces uncertainty about a preference (e.g. the utterance *"I might like to take a Language Technology course"*). Hedgings represent the weakest form of preference strength.

Carberry *et al.* propose to utilize dialogue context as a second influencer of preference strength. Four conversational circumstances are identified and ranked according to the difference in preference strength:

1. **Reject-Solution.** User gives preferences as a reason for rejecting a recommendation/solution.

2. **Volunteered-Background.** User includes preferences as part of a background problem description.

3. **Volunteered.** User volunteers preferences without prior prompting from the system.

4. **Question-and-Answer.** User provides preferences in response to a direct system query.

Preferences may also be deducted from a history of recommendations[6] as well as stereotypes or generalization. Figure 2.2 shows the preference strength ranges for conversational circumstances.

Utterance types, dialogue context, and deductions are combined into a preference strength measure in the ordinal range *weak-2, weak-1, mod-2, mod-1, str-2, str-1.* This range can be translated to an integer preference strength interval $[1, 6]$[7].

---

[6] Called a *proposal history* in the work by Carberry *et al.*.

[7] This is for positive preferences. For negative preferences the scale is inversed, and the strength interval thus ranges between $[-1, -6]$.

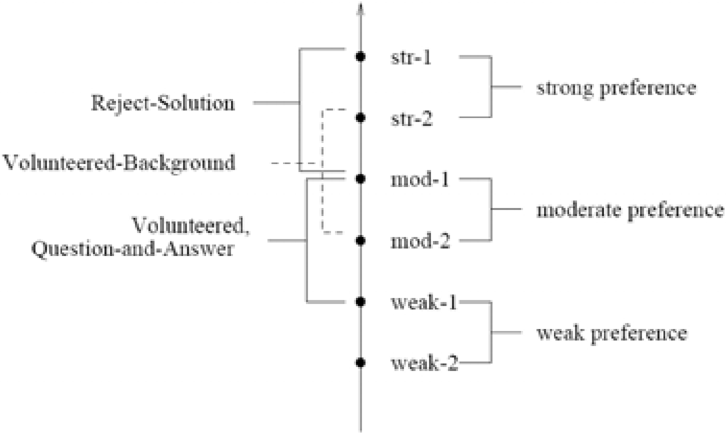


Figure 2.2: Preference strength ranges for conversational circumstances. From [Carberry *et al.*, 1999].

Reliability

Reliability is modeled by the use of *endorsements*, which are viewed as explicit factors that affect the modeling agent's certainty in a preference strength hypothesis. Carberry *et al.* utilize endorsements that reflect how a preference is detected and ranked on an ordinal scale as shown in Table 2.2. Endorsements that do not match conversational circumstances represent ways to detect implicit preferences. This is done by deducing facts from a structure that keeps track of system proposals (Proposal

Table 2.2: Reliability ratings of endorsements. From [Carberry *et al.*, 1999]. PH = Proposal History.

Value	**Reliability**	**Endorsement**
5	Very-Strong	Reject-Solution, Volunteered-Background
4	Strong	Volunteered, Question-and-Answer
3	Moderate	PH-Deduction-Strong
2	Weak	PH-Deduction-Weak, Stereotypical, Indirect-Propagation
1	Very-Weak	PH-Deduction-Initial

History), or by using stereotypical reasoning (if applicable depending on domain). The main reason for using endorsements for reliability is to allow for accumulation and combination of several (weak) pieces of evidence for preferences. Second, a motivation generation component for recommendations can justify and better explain a recommendation by taking preference reliability into account. The intricate details of the use of endorsements and reliability measures are not necessary for the current overview, and we refer to the original source for more details. We will return to reliability and preference strengths in Chapter 5.

Attributes

Carberry *et al.*'s model focuses on attribute-value type preferences, and they identify three attribute types: disjoint, scalar, and complex. *Disjoint* attributes carry values for which preferences do not conflict, such as *artist* in the music domain. A user can like or dislike artists in any combination without conflict. *Scalar* attributes are viewed as having values on a scale, where different values would conflict with each other. As an example of this, consider difficulty level of university courses. A preference toward very easy courses is viewed as being in conflict toward a preference toward very difficult courses. *Complex* attributes also have values that are scalar, but where values may fall on different points on the scale without conflicting. In the music domain, a user might like both an artist's albums released 1955–1960, as well as albums released 1968–1972 (but perhaps not the ones in between).

Carberry *et al.*'s work shows how dialogue interaction efficiently can work together to model variations in preference strength by utilizing the modality- and interaction technique-specific properties of natural language.

2.2.3 Recommendation Models and Algorithms

The purpose of recommender systems is to produce personalized recommendations of potentially useful items from a large space of possible options. To accomplish this, the system employs one or more recommendation algorithms (also known as prediction techniques), that operate on a model of user preferences. Recommender systems can be characterized in a number of ways. One taxonomy, suggested by Burke [2002], bases the categories on the used data sources and algorithms used. For a survey of various recommender systems currently used on the Internet, see [Montaner *et al.*, 2003]. The following categories are considered in Burke's taxonomy:

- Collaborative filtering
- Content-based
- Demographic
- Utility-based
- Knowledge-based

These models have their own advantages and disadvantages, and each is described below.

Collaborative filtering (CF) systems are widely used and perhaps the most familiar recommendation technique. CF systems utilize the rating information of several users (hence the term "collaborative") in order to predict item ratings for a specific user. A preference model typically consists of a vector of items rated by the user. This vector is sometimes called a "basket"[8]. The vector is compared to other users' vectors with an appropriate similarity measure and a neighborhood of similar users is identified. Recommendations then basically consist of previously unseen/unrated items in the neighborhood for each user. The ratings in the vectors can either be binary (e.g. seen or not-seen; purchased or not-purchased etc.), or valued (e.g. rated on a scale from -1 to 1, or 1 to 5). The main advantages with the CF approach are that it:

- works well for domains where the items consist of aspects that are hard to model correctly, such as music, movie, and book taste.
- always is a "correct" and relevant model of end-users' preferences, and where each user's personal preference is catered for in the community. This assumes that users' ratings do not change too often and that users keep rating items continuously.

---

[8] As in "shopping basket", due to the many commercial online shopping implementations that use CF.

- can cope with cross-genre recommendations; e.g. making confident predictions of comedy movies to a user U that never rated comedies before (as long as the neighborhood of U contains comedies).

- requires no domain knowledge.

- is adaptive, i.e. the model improves over time as more ratings are added to the preference model.

CF systems work best if the domain objects do not change too often, in which case other users' ratings become less important. Furthermore, if ratings in general are sparse it becomes hard to identify a correct and relevant neighborhood. There is also a problem if a specific user's basket is too small. This raises the question of how to "fill the basket" as quickly as possible. This issue is known as the new-user cold-start problem. A related issue is when a new object (such as a newly released movie) enters the domain, and thus contains very few (if any) ratings in the community. This issue is called the new-item cold-start problem.

Content-based (CB) systems utilize a user preference model based on the features of the objects rated by the user. Instead of deriving a user-to-item correlation and defining neighborhoods, item-to-item correlation is used. User preference models are—as in the case of CF models—long-term and improved as users rate more items in the domain. The advantages and disadvantages are basically the same as CF systems with two important exceptions. On the one hand, CB systems can not identify cross-genre items and thus tend to stick to the same *type* of recommendations, whereas CF systems can introduce new types (see above). On the other hand, the new-item cold-start problem is not apparent in CB systems, since all its features are known as soon as it is introduced and not dependent on user ratings. Another feature of CB systems is that items are limited to their initial description—or features—and this makes the technique dependent on the features that are explicitly given. CB systems naturally require a domain model; often in the form of attribute-value descriptions of the included items. Both CB and CF systems suffer from the new-user cold-start problem.

Demographic systems rely on explicit user attributes and base recommendations on the demographic cluster that a user belongs to. This kind of recommender is thus stereotypical, since they build on the assumption that all users belonging to a certain demographic group have similar taste or preference [Rich, 1989]. One of the first recommendation systems—GRUNDY—was a book recommendation system developed by Rich [1979]. The main disadvantage with demographic systems is the need to gather demographic information, with all the difficulties and privacy issues that comes with it. Both new-user and new-item cold-start problems exist in demographic systems [Burke, 2002].

Utility- and Knowledge-based systems are related to each other and for the purpose of this survey it is suitable to group them together. A utility-based system is typically short-term, and bases recommendations on utility values of each item in a domain for a specific user. Knowledge-based systems employ functional knowledge, i.e. explicit knowledge about how items in the domain meet user needs [Burke, 2002]. Knowledge-based systems do not require a utility function from the user. However, they require knowledge engineering, which is very expensive. Knowledge-based systems have the power to identify how features in an item explicitly address user preferences (or problems that the user wants to solve) and reason about how items meet needs. Knowledge engineering may take many forms, but according to Burke [2002, page 338] all knowledge systems require: (a) catalog knowledge about all objects (such as ontological relationships etc.) in the domain, (b) functional knowledge describing how user needs map to item features, and (c) knowledge about users. User knowledge can be of varying form depending on the application.

Both types share the advantage of not being prone to cold-starts. This is definitely a big advantage. However, together they share two (probably just as big) disadvantages: First, utility-based systems require the user to input the utility function which is to be satisfied. This function must cross all features of the objects. A benefit of this is on the one hand that a skilled user can express non-product specific attributes. On the other however, this demands that the user is a skilled professional who can design her utility functions efficiently, since they require the user to take all attributes of the domain into account. Second, these systems are static and cannot learn or improve their recommendations as e.g. CB and CF systems can. The inflexibility of the utility-based approach does not fit casual browsing, since moving around in the item space is cumbersome due to the fact that a new utility function must be conveyed for each such move. Finally, knowledge-based systems require knowledge engineering.

2.2.4 Recommendation Performance Factors

It is hard to state if one of the above recommender system types generally are better than the others, since they all have trade-offs. Indeed, much attention is given to combine the above techniques into *hybrid recommenders* in order to utilize the best (and eliminate the worst) characteristics of the different techniques [Burke, 2002; Carenini *et al.*, 2003]. Hybrid recommenders show promise to address the most crucial part for recommender system performance: the accuracy of item recommendations and predictions. However, the combination of algorithms is only one of the key factors to efficient and accurate recommender systems.

A second important factor is the content and density of the set of user ratings [Carenini *et al.*, 2003], or the user preference model. While this problem exist for all recommender types (except utility-based systems), the problem has received most attention in CF systems. In CF systems, the preference model ("basket") consists of ratings of items in the domain. The more ratings in the model, the better predictions (and thus recommendations) the CF algorithm can compute. Building user model content is highly related to the new-user cold-start problem.

For completeness, a third key factor can be added to algorithms and preference model density: The use of domain knowledge management and ontologies as proposed by Middleton *et al.* [2002]. They report on successful integration of the QUICKSTEP recommender system and an ontology. QUICKSTEP is a hybrid recommender system and bases its user interest profiles on an ontology of research paper topics. The construction of ontologies requires knowledge engineering and this approach thus suffers from the disadvantages from the knowledge- and utility-based system class.

Recommender system research should thus be focused on (a) developing recommendation techniques and algorithms (including combinations of existing techniques), and (b) interaction design for efficient preference data acquisition [Johansson, 2003b]. According to Carenini *et al.*, the latter aspect has been neglected to a large extent [Carenini *et al.*, 2003].

As hinted above, one problem prominent in all types of recommenders (except for the utility-based systems is the new-user cold-start (or "ramp-up" [Burke, 2002]) problem. In order to give personalized recommendations, systems have to know about the user's preferences. The process of acquiring these preferences demands time and effort from a new user. This delayed benefit is in effect the new-user cold-start problem. Users want to be able to efficiently start using the system right away, and get relevant information the minute they start using it. The cold-start problem is a serious problem, since it has been shown that most users tend to resort to non-personalized information-browsing instead of investing the effort of conveying the preferences needed by the system. Indeed, asking for preferences in advance might even be impossible in some cases as discussed in Section 2.2.2; and Baudisch and Brueckner [2002] recommend that regular information-providing behavior thus should be a necessary functionality of recommender systems in order to ensure immediate benefit.

The process of getting to know a user's preferences varies depending on the application, and the recommendation technique used. Most CF systems require the user to go through a process of explicitly rating a number of pre-defined items in the chosen domain as they are provided by the system. This is for example the approach taken in the MovieLens movie recommendation system [Rashid *et al.*, 2002]. Recommendations in such systems start out being based on the "average" user preferences. As the user rates more and more items, the recommendations gradually improve.

Another approach is to let the user give explicit content-based preferences in some sort of sign-up process. This is currently a common practice in several commercial systems, such as Amazon[9].

With sparse data in the preference model, we will always face the cold-start problem—no matter how good prediction techniques and algorithms we develop. Hence, research toward devising suitable interaction techniques for preference eliciting is important. The next section surveys some contemporary attempts to address this issue.

---

[9] `http://www.amazon.com`

2.3 Conversational Recommender Systems

A conversational recommender system utilizes natural language dialogue between the user and the system to initialize, continuously update, and put the user's preferences to use in order to calculate and present personalized item recommendations.

One underlying motivation for exploring a human-like conversational approach to recommender system interaction is voiced by Aksoy *et al.* [2006, page 297]:

> it helps consumers to use a recommendation agent that thinks like them, either in terms of attribute weights or decision strategies

This implies that preference attribute weights detected in the human-system dialogue should reflect how humans detect preferences, and that preferences and recommendations should be handled in a human-like fashion in the interaction.

A second motivation for the conversational approach is that it aims to exploit situations where the user has a high motivation to provide preference data. For this purpose Carenini *et al.* [2003] propose the *Conversational and Collaborative Model* (CC), in contrast to what they refer to as a *Standard Interaction Model.* The Standard Interaction Model preference acquisition occurs at registration time, after which user and system communicate in "an independent and asynchronous way" [Carenini *et al.*, 2003, page 13], resulting in delayed benefit for the user. Carenini *et al.* identify four situations where users have a high motivation to provide preference data:

- The user asks for a rating for an item she is interested in, but the system signals that it does not have a sufficient preference model and asks the user for more ratings.
- The system predicts an average rating (i.e. recommendation is not good enough to decide whether or not the user should be interested). The user is willing to provide more ratings to get a better supported recommendation.
- The user is puzzled by a recommendation (e.g. the user believes a prediction for an item would be significantly different).
- If the user rates items, the accuracy of *other* users can be improved. In the future, they may reciprocate. (Certain systems also implement "rewards" in the user community for this type of behavior.)

Even though the CC model aims for conversational interaction, and indeed employs something like a dialogue flow, it lacks a natural language processing component.

The conversational approach is also exemplified by Burke *et al.* [1997],who present a range of recommender systems based on the FindMe *Assisted Browsing* framework (e.g. assisting users to browse through car, video, apartments, stereo equipment, and restaurant information). Burke *et al.*'s focus lies on structuring information retrieval based on users' critique of previously retrieved items in the domain. The FindMe systems are knowledge-based and the framework can be applied to interaction situations where the user faces a large, fixed set of choices and where the domain is too complex for users to fully articulate a specific information query. However, when faced with a retrieved item, the domain is familiar enough for the user to articulate some critique of the solution. The crucial point here is that the critique can be different for different users; the critique signals what attributes are important for a specific user. This is called *tweaking* in the FindMe framework. Consider the following movie recommendation implementation tweaking: Let us say that a user U is recommended the violent science-fiction movie *Terminator II* starring Arnold Schwarzenegger. Theoretically, the user can criticize each single attribute of this movie in response to the recommendation. Responses such as (a) *"it's too violent"*[10], (b) *"I don't like science fiction"*, or (c) *"I don't like Arnold Schwarzenegger"* are all valid, and signal what attributes are important to user U. The next recommendation will be very different depending on which of the responses (a, b, or c) U chooses.

FindMe systems aim to reduce complexity, but maximize functionality. However, Burke *et al.* acknowledge that using only direct manipulation in a purely graphical user interface falls short compared to natural language when describing the movie recommendation systems Video Navigator and PickAFlick [1997, page 16]:

> Interface constraints also entered in the decision not to employ tweaking in Video Navigator and PickAFlick. There are simply too many things that the user might dislike about a movie for us to present a comprehensive set of tweak buttons. ... natural language tweaking capacity ... is the most likely candidate for a tweaking mechanism in this domain.

---

[10] Even the statement *"it's not violent enough"* is a possible ground for rejecting the recommendation; emphasizing the importance of expressing preferences qualitatively.

Another approach based on users' critique of system solutions is the *Candidate-Critique Model* (CCM) proposed by Linden *et al.* [1997]. The CCM is implemented in an automated travel assistant, and builds on the assumption that communication between the system and the user is in the form of system *candidate solutions* to a problem, and user *critiques* of those solutions. Although not implemented in the system, free-form natural language dialogue is the ultimate aim of the system, since that would [Linden *et al.*, 1997, page 69]:

> allow solution information to be communicated concisely from the system to the user and allow arbitrary information about the user's preferences to be communicated from user to system

The CC, assisted browsing, and CCM approaches all use graphical user interfaces with direct manipulation and typing of search terms as means of interaction, even though they all acknowledge the possibility and potential benefit of using natural language interaction [Carenini *et al.*, 2003; Burke *et al.*, 1997; Linden *et al.*, 1997].

As an example of an approach that utilizes spoken natural language interaction, consider the conversational recommender system ADAPTIVE PLACE ADVISOR [Thompson *et al.*, 2004]. It was first presented with a graphical user interface [Langley *et al.*, 1999], and then with a natural language interface [Göker and Thompson, 2000; Thompson *et al.*, 2004]. The latter implementation is one of the first spoken personalized conversational recommender systems. The approach contrasts against the ranked-list approach commonly used in other recommender systems. Instead, the goal of the ADAPTIVE PLACE ADVISOR is to narrow down the alternatives by having the user remove instead of re-order items. As noted by Göker and Thompson [2000], deriving preferences from on-going interaction, and gradually narrowing down choices by allowing partial descriptions of items, is a suitable recommendation strategy for conversational systems. The benefits are that (a) the user is not overwhelmed by a myriad of items, and (b) the user is aided in her understanding of the domain and her preferences by thinking about questions in the dialogue. The ADAPTIVE PLACE ADVISOR utilizes dialogue moves to modify both the current query and the user preference model. For example, if the result set size for a query is larger than four items, the user is asked to constrain the query with attributes or values of item properties. If the result set is empty, the system asks the user to relax the current constraints. Sizes in between (i.e. 1–4) are manageable by the user and thus recommended.

Finally, the AdApt system serves as an example of a multi-modal recommender system. AdApt is a dialogue system that helps the user to find apartments by asking questions and providing guidance in a dialogue [Gustafson *et al.*, 2000]. The system employs an animated talking head and presents information both graphically on maps, and auditory by the talking head. Furthermore, the system allows for both direct manipulation of the graphical user interface by means of mouse pointing, as well as speech recognition. According to Gustafson *et al.*, the apartment domain is complex enough to warrant natural language interaction as one major interaction modality. Since the research focus of AdApt is on multi-modal interaction—and not on recommendation techniques as such—issues related to recommendations in the dialogue are not thoroughly described. Indeed, it is questionable if AdApt should be categorized as a proper personalized recommendation system, since there is no explicit, individualized user preference model built. AdApt instead utilizes an implicit user model that presupposes cooperative dialogue behavior from the user. AdApt follows the Adaptive Place Advisor's notion of eliminating items from ranked lists. (AdApt tries to limit the matching number of hits to 7 or less, whereas Adaptive Place Advisor tries to limit the list to 4 or less).

Since conversational recommender systems rely heavily on dialogue and incremental modification of queries and preference models, a system's dialogue strategy to initialize, update, and utilize the preference model in the interaction is crucial for the system's performance and usability.

2.4 Development of Conversational Recommender Systems

Dialogue system development is naturally viewed as a case of software development. As a subclass, sets of general methods and evaluation criteria have evolved and is slowly maturing. For instance, the DISC project aims at defining best practice methodology regarding specification, design, usability, and evaluation of (spoken) dialogue systems [Dybkjær *et al.*, 1997].

The topic of dialogue system development is vast, and this section can only deal with a small part of it. The interested reader is encouraged to consult works such as McTear's [2002]. The rest of this section gives an overview of development methodologies (Section 2.4.1), interaction design and usability (Section 2.4.2), and evaluation (Section 2.4.3).

2.4.1 Design and Development Methodologies

As software systems tend to become more and more complex, and given the human limitations in keeping an overview of complex systems, products developed according to a methodology that requires up-front analysis, followed by design, and then implementation—such as the waterfall method as it is commonly applied—run the risk of ending up being filled with flaws [Martin, 1999]. Such heavy (or monumental) methodologies are contrasted to agile (or light-weight) methods [Fowler, 2000]. The cornerstones of agile methods are iterative and incremental software development. Iterative and incremental development [Larman and Basili, 2003], is viewed as consisting of iterative development cycles, where each iteration may consist of e.g. incremental additions of functionality or re-factoring of code. The term *evolutionary development* is used to describe an iterative and incremental methodology. Simplicity in both tools and design is crucial for agile evolutionary development in general [Beck, 2000, page 103], and in the case of dialogue systems in particular [Degerstedt and Jönsson, 2001; Johansson *et al.*, 2002].

According to Bryson [2001], the chosen architecture greatly influences the design method. It is thus, given the arguments for evolutionary development above, important that the dialogue system architecture supports evolutionary development [Degerstedt and Johansson, 2003]. That is, functionality should be easy to add, remove, and modify as requirements might change during the life-cycle of the project. Refactoring in terms of separating dialogue management, preference models, and back-end resources, etc. should be encouraged and supported by the architecture. If the architecture supports the principle of making generics as generic as possible, and specifics as specific as possible, it helps making the engineering sound.

2.4.2 Interaction Design and Usability

For a conversational recommender system interaction design needs to be approached from two aspects: the natural language interaction aspect, and the recommender system aspect. A smooth integration of both aspects is necessary for achieving a usable system.

Natural Language Interaction Design

Natural language interaction differ from traditional graphical user interface interaction, and requires its own set of methods to ensure user acceptance and usability issues [Yankelovich, 1997]. If system design methodology for dialogue system construction is a young field, usability issues and interaction design for natural language interaction have only recently started to attract attention, since there hitherto has been a strong *functional* focus on the systems. Indeed, we have previously said that it might be a good idea to pursue the goal of human-like conversational recommender systems (see Section 2.3). However, it is important to note that solely aiming for unrestricted human-human dialogue is not a guarantee for ensuring usability. This builds on the assumption that an unrestricted dialogue system is more usable than one that is not. In theory this might be true, but the significant deficiencies in current natural language understanding technology and theory is a fact that voids this assumption [Pieraccini and Huerta, 2005]. While we take inspiration from human-human dialogue when designing a conversational system, we still need to define and address explicit usability aspects in order to achieve successful interaction.

Natural language interaction is typically based on human-human dialogue analysis; *Wizard-of-Oz* (WOZ) dialogues; dialogue theory and guidelines such as Gricean maxims [Grice, 1975], or the Dialogue Evaluation Tool [Dybkjær *et al.*, 1998]; or a combination. In WOZ studies [Dahlbäck *et al.*, 1998], the aim is to collect human-machine dialogue corpora, where the machine is emulated by a human "wizard"—whose existence is supposedly unknown to the user. One advantage is that it is economical, since there is no need to build a functional system beforehand. An interaction protocol of some sort which the wizard follows is "enough". A second advantage is that it is practical, since the alternative (that of analyzing unrestricted human-human dialogue) may be too large of a step to restricted human-machine dialogue.

However, there is a viable alternative—or middle-way—to the two approaches: *dialogue distilling*, which is a corpus analysis method with a particular aim for dialogue system development [Jönsson and Dahlbäck, 2000; Larsson *et al.*, 2000]. In the distillation process, human-human dialogues are re-written to resemble human-computer dialogue. This task is systematically carried out by applying guidelines. The resulting corpus can then be used as base for dialogue modeling. The motivation for choosing distillation of human-human dialogues in the beginning of a dialogue system development project is economical (the same argument as for WOZ studies), and design-theoretically sound, since free flowing dialogue between two humans may provide data that correspond better to the relevant tasks and behaviors in the domain—without the inbuilt assumptions inherent in the design of a WOZ environment [Jönsson and Dahlbäck, 2000]. Dialogue distilling is more thoroughly described in Chapter 3.

An important aspect of successful natural language interaction design is the *prompt design* and surface realization. By carefully formulating system prompts a smoother interaction can be achieved. This is emphasized in commercial applications and much effort is devoted to customizing prompts. Within the research community, this falls under the Natural Language Generation field [Jurafsky and Martin, 2000], and includes following Gricean maxims [Dale and Reiter, 1995], and generating human-like referring expressions, e.g. [Viethen and Dale, 2006], etc.

In their studies of natural language interaction in electronic TV program guides, Berglund and Johansson [2004] and Ibrahim and Johansson [2002a] present interaction design issues and guidelines for natural language dialogue-based information systems. In their study, users reported that they found natural language dialogue efficient and flexible since they could specify a lot of information in queries in certain situations, and rely on the mixed-initiative capability of the system to aid completion of partial user queries in others. Natural language generation of query results tailored for the user was appreciated by the participants of the study; indicating that careful attention should be paid to the design surface realization of system output.

Ibrahim and Johansson [2002b] propose a separation of the user, agent, and back-end resources in their *three-entity interaction model*. The separation is done on interaction modality, as well as concretely communicating it in the graphical presentation on-screen. Ibrahim and Johansson found that this separation communicates the dialogue capabilities of the system more efficiently, and encourages the users to take advantage of them.

Recommender System Interaction Design

We now turn to the other aspect of conversational recommender system interaction design: recommender system-specific interaction issues. Since recommender systems constitute a subclass of user-adaptive systems (i.e. systems that adapts its behavior at use-time, depending on detected characteristics of an individual user), we are presented with specific problems such as non-determinism and the difficulties of predicting system behavior at design-time (see Section 2.2.1). Furthermore, there are specifics concerning the delivery of recommended items that need to be considered. The research in this area is limited to date, and what exists is targeted toward traditional recommender systems with graphical user interfaces and direct manipulation interaction (e.g. [Swearingen and Sinha, 2002]). Therefore, some care has to be taken when applying the guidelines to conversational recommender systems.

In short, the following guidelines should be observed in recommender system interaction design:

- Users' *trust* in the system increases if the system logic is transparent; if there is familiarity of both the recommended items and the process of receiving them; and if previously liked items is included in the recommendation delivery. Indeed, Buczak *et al.* [2002] found that users thought a recommender system was broken when it recommended TV shows unknown to the user, without an accompanying explanation relating to the users' preferences.

- Users dislike bad recommendations more than they dislike providing a few additional ratings or preferences. This needs to be carefully approached however, due to the cold-start problem of delayed benefit (see Section 2.2.4).

- The process of eliciting preferences should be easy and engaging (e.g. a mix of different type of questions, and continuous feedback during the input phase).

- The system's categorization of e.g. genres should map to users' mental models. Filtering over genres should be easy and self-explanatory.

- Users should not perceive the recommendation set as a dead end, but rather as a dynamic set that changes due to additionally elicited preferences or item ratings.
- Additional subjective and objective item information should be easily available. Reviews and ratings by other users are important, as well as clear paths to detailed item information.

2.4.3 Evaluation

In the area of user-adaptive systems (such as personalized recommender systems), evaluation has gradually become more and more critical, due to their dynamic nature. Requiring evaluation and verification for scientific results is far from a new and bold statement, but according to a recent survey, empirical evaluations in user-adaptive systems research are still quite rare [Chin, 2001].

For researchers and developers working on adaptive conversational systems several evaluation frameworks are available depending on the focus. If the focus is on Language Technology research (i.e. focusing on *dialogue* more than adaptive functionality), one of the most well-known and widely accepted frameworks available for dialogue system evaluation is the PARADISE evaluation framework [Walker *et al.*, 1998; Litman and Pan, 1999]. PARADISE provides elegant metrics for assessing dialogue strategy comparison, with a strong focus on *task-oriented* (spoken) dialogue systems. PARADISE is typically applied on fairly mature systems and relies on a definition of an ideal “key” to resolve specific tasks in the system’s domain.

If, on the other hand, the focus is on *adaptive functionality* a number of general evaluation frameworks are available. Examples include Gupta and Grover’s layered approach for adaptive hypermedia systems [2003], and Paramythis *et al.*’s modular approach [2001]. The modular and layered approaches to adaptive system development are in general sound, since they cater for the fact that comparing adaptive functionality with a non-adaptive version of the system can be an absurd approach, since the adaptive functionality often is an integral part of the system [Höök, 2000; Weibelzahl and Weber, 2002]. However, these evaluation methods are not optimal when considering systems that uses natural language and dialogue interaction, since they have been designed for traditional user interfaces such as web and desktop applications. The problem is the sometimes overlooked fact that there are no established standards for dialogue system interaction as in the case of visually oriented interaction, such as the dominating Window-Icon-Menu-Pointer (WIMP) interaction metaphor. Only in certain application types is there a gradually maturing standard and end-user deployment, such as information-providing services in the travel reservation domain. The lack of dialogue system interaction standards can be attributed to the incomplete operationalization of pragmatics and different dialogue genres. It is therefore important to adopt an exploratory approach in the evaluation and focus on issues not necessarily covered by the general methods in order to gradually reach sound dialogue system standards.

According to Hulstijn [1999], usability is one of the key quality factors for dialogue system performance, that concern both design and evaluation. Hulstijn considers the following properties for dialogue system evaluation:

- **Effectiveness**. The accuracy and completeness with which users achieve their task.
- **Efficiency**. The relative effectiveness of a system set in relation to the effort to achieve it.
- **Transparency**. A system is transparent if the user's mental model of the system capabilities and behavior coincides with the design of the system.
- **Coherence**. The coherence of the system is to what degree the system's utterances are combined into a well-formed dialogue.
- **Satisfaction**. Satisfaction is defined as they way users perceive the system and may be seen as a measure of usability.

Hulstijn [1999] notes that corpus-based methods tend to be conservative since a change in system functionality is likely to change users' behavior. As this quickly may render the corpus inadequate we want to keep corpus dependency to a minimum. Therefore, collecting and analyzing corpora should be fast and cheap, and more importantly so in the beginning of exploring the system and domain requirements. According to this view, the evaluation step has a strong implication on the initial design method, even though evaluation typically is employed at the final stages of system design.

One rapid evaluation paradigm is called the “overhearer” model [Whittaker and Walker, 2004]. The reason for using the overhearer model is to avoid natural language interpretation problems (e.g. for projects where the coverage of grammar and lexicon is not in focus). Furthermore, it allows for comparison of alternative dialogue strategies in the same dialogue context, and has been used to evaluate adaptive generation tasks for dialogue systems [Foster and White, 2005].

To gain feedback on system interaction and functionality, additional forms of user input is needed. There are several HCI methods available for this purpose, such as surveys and interviews (subjective in nature) that could be used as a complement to dialogue session logs and other more objective metrics.

2.5 Summary

Conversational recommender systems dwell in the intersection of dialogue system, user modeling, and recommender system research, tied together by general software development and evaluation methodology including empirical approaches and HCI. Setting this scene has been the topic of this background chapter. While these research efforts provide us with many pieces of the puzzle, the integration of them, as well as the quest for the missing ones, remains to be carried out.

Some dialogue genres, such as inquiry-oriented dialogue, are well-known phenomena. However, the particularities of *recommendation* dialogue deserves more attention, both in terms of an empirically based description of recommendation dialogue, as well as recommendation dialogue as it should be designed and developed in computational systems. One natural next step is thus to study human-human recommendation dialogue in detail, and then perform steps, with the existing pieces of previous research in mind, toward a computational model of human-computer recommendation dialogue.

3

An Empirical Study of Recommendation Dialogue

This chapter describes Study I, which is a human-human dialogue corpus study in the domain of movie recommendations. The analysis of the study consists of a dialogue distillation, and results in a characterization of recommendation dialogue, and forms a base for a first prototype implementation of a dialogue control strategy for movie recommendations, which will be presented and evaluated in Chapter 4.

A FIRST STEP toward understanding the dialogue genre of conversational recommendation is to study how humans carry out recommendations and gathers preferences that will influence recommendations in dialogue. It is a well-documented fact that human-human dialogues are not identical to human-computer dialogues [Jönsson and Dahlbäck, 1988; Reeves and Nass, 1996]. However, we ended the last chapter with noting the problem of the lack of dynamics when employing expensive corpus-based methods in an exploratory, evolutionary dialogue system design process, since a change in system functionality is likely to change users' behavior (rendering the original corpus inadequate). Therefore, collecting and analyzing corpora should be fast and cheap, and more importantly so in the beginning of exploring the system and domain requirements (see Section 3.2.3). Along these lines, our first data collection is a human-human dialogue corpus, intended to capture initial characteristics on recommendation dialogue.

In Section 3.1, the experimental design and procedure is described. As analysis method, the choice is (the economical) *dialogue distilling* method [Jönsson and Dahlbäck, 2000; Larsson *et al.*, 2000], which is the topic of Section 3.2. Armed with the data of the analysis we characterize recommendation dialogue informally in Section 3.3, which will form the basis for our first stab at a recommendation dialogue strategy model in Chapter 4.

3.1 Experimental Design

The study was set up to record dialogues about movie preferences and recommendations. The dialogues were between one dialogue partner playing a movie *recommender* role, and one partner acting as a *customer* looking for movie recommendations.

3.1.1 Participants

Forty-eight participants (24 female and 24 male students at Linköping University) were recruited. The participants did not know each other. More specifically and importantly, they did not know each other's movie preferences. Each dialogue session required two participants, one acting in the role of a movie recommender, and the other in the role of a customer wanting movie recommendations. In order to avoid repetition of recommendation strategies in the dialogues, each session had a new recommender. The benefit of receiving varying dialogue strategies is judged to outweigh the benefit of having a more experienced recommender (i.e. that would have acquired more "recommendation skill" due to his/her acting in several sessions). An experienced recommender would obviously be an advantage, but at this stage we prioritize variation.

All sessions were kept as varied as possible in terms of gender and roles, including male-female, male-male and female-female dialogues, as well as mixing the recommender and customer roles in order to ensure as much variation as possible in the dialogues. The participants were not paid.

3.1.2 Apparatus

The study was set in a home environment designed for usability studies. Apparatus for the study included:

- A laptop connected to an extensive movie information database[1] for the recommender to use as information source. The database contains over 540,000 movie titles with information on actor, director, genre, plot, year, award, language, and several other attributes for each title.
- A movie recommendation protocol—also called the "to-see list"—where the recommender writes down movie recommendations for the customer.
- Scratch pads and pencils for both customer and recommender to use for note-taking.

A stereo microphone connected to a digital recorder was used to record each dialogue session.

3.1.3 Procedure

One of the dialogue partners in each session was assigned the role of a professional movie recommender, and got a 15-minute tutorial on how to use the movie information database. The second dialogue partner was assigned the role of a customer, and received no training.

One scenario was specifically designed for the customer, which included a short background story. This scenario, in combination with the home environment was aimed to put the customer in the right mood; and to provide motivation for conveying preferences and be cooperative in the dialogue with the recommender.

---

[1] The Internet Movie Database (`http://us.imbd.com`).

The recommender also received a scenario prior to the tutorial, which provided her with the task of recommending in total ten movies. Five of these should have been seen by the customer and verified as “good” recommendations (in order to indicate that the recommender acquired and utilized the customer’s preferences), and five should be previously unseen. When the recommendation protocol (referred to as “the to-see list”) was completed, the session was terminated. Translated versions of the scenarios are available in Appendix B.

Both dialogue partners were allowed to use the scratch paper and pencils provided in order to keep notes throughout the session. Furthermore, even though the recommender used the online movie database, both participants were allowed to look at the information on the laptop screen.

3.1.4 Results

A total of 24 dialogues were recorded with two participants in each session, resulting in 7.5 hours of recorded material. In addition to recording, two observers were present (however residing in the background to avoid interference). The scratch pads were collected along with observation notes from the two observers. Transcription of the dialogues resulted in 2684 utterances with a mean of 112 utterances per dialogue. All dialogues are in Swedish, and excerpts below have been translated to English.

3.2 Analysis

The human-human dialogue corpus was systematically re-written to human-machine dialogues through the process of dialogue distilling. The method can be seen as a complement to WOZ studies (see Section 2.4.2).

This section gives a brief overview of the process of dialogue distilling, followed by an account of the guidelines used in the analysis phase of User Study I, and the results of the distillation.

3.2.1 The Dialogue Distilling Process

When distilling, we aim at re-writing an original human-human dialogue corpus into a plausible human-machine dialogue. In the movie recommendation corpus, we appoint the recommender participant to function as “the system”, and the customer as “the user”. In the following, this is how they are referred to. In general, dialogue distillation is a two-step process [Larsson *et al.*, 2000] consisting of:

1. Guideline development
2. Guideline application

It is important to realize that even the most thoroughly developed guidelines cannot be fully objective and exact, and in the application of them, issues will most certainly arise where the distiller may have to go back and refine or re-define the guidelines [Larsson *et al.*, 2000].

3.2.2 Guideline Development

Defining the guidelines is a complex task, because all communication characteristics and phenomena are less than perfectly understood—both concerning human-human as well as human-machine communication. In order to perform methodological changes to the corpus, Jönsson and Dahlbäck [2000] suggest two sets of guidelines, which are further developed by Larsson *et al.* [2000]. This boils down to four sets of guidelines: general guidelines, linguistic properties guidelines, functional properties guidelines, and ethical properties guidelines. Each set consists of a number of specific guidelines, as well as rules of thumbs for applying them. For the corpus collected in User Study I, several guidelines were developed based on these suggestions. They are summarized in Table 3.1.

3.2.3 Guideline Application

Several issues arises when trying to apply the guidelines. As noted by Larsson *et al.* [2000], the guideline application step boils down to “common sense”, and perhaps the most notable benefit of using the distillation method is that the distiller is forced to confront abstract principles with the concrete dialogue data, and thereby gains an understanding of the dialogue system that is to be built. As the collection of issues provided in this section shows, the distillation process provides us with a large number of observations that enhance understanding of the dialogue behavior of a conversational recommender system.

Table 3.1: Dialogue distilling guidelines used in the movie recommendation dialogue corpus analysis. (S = system, U = user). Adapted from Larsson *et al.* [2000].

Guideline	**Explanation**
Linguistic properties	
Syntax	S speaks syntactically correct in full sentences, and does not mumble or hesitate.
Turn-taking	S does not interrupt U, and always allows U to take initiative.
Focus	S presents the information in such a way that coherence with the focus of the user's utterance is maintained.
Functional properties	
Relevance/Quantity	S only presents relevant information.
Quantity	S asks only for the information it needs to complete its task; no more and no less.
Immediacy	S gives all relevant information at once.
Memory	S does not forget information and does not ask user twice about the same piece of information.
Orderliness	S follows a certain order when asking questions; S does not skip between questions.
Repetition	S does not repeat itself unless asked to.
Mapping	S is responsible for mapping the NL representation of a database request to a suitable one.
Ethical properties	
Honesty	S does not lie and does not try to cheat the user.
Politeness	S is polite.
Seriousness	S is not ironic, does not flirt, etc.
Voluntariness	S does not try to persuade the user.
User Initiative	S does not take the initiative (including the turn) from the user.
Neutrality	S does not express its own opinions.

R1	have you seen *Star Wars*?
C1	yeah the new *Star Wars* movies are quite lousy / the first / uhm / episode one / was really bad because of that computer animated clown that jumped around and squealed / of course they removed him in the second movie / which was good / but that movie didn't have any plot at all
R2	okay
C2	but then again they're good to watch / you know lots of special effects and great sound / so I'd watch them anyways
R3	right / I see / have you seen *The Matrix*?

Figure 3.1: Excessive customer input from the human-human dialogue that will be modified in the distilled version. Movie titles are in *italics*. R = Recommender, C = Customer.

Modification of Excessive User Utterances

The biggest challenge is to apply distillation guidelines on user input. On the one hand, we want to stay as true as possible to the human way of expressing herself—as in the human-human corpus—since we do not want to build a dialogue system that restricts the user. On the other, several difficulties arise as many utterances are very complex and ambiguous and thus hard to implement in a dialogue system. Furthermore, capturing the multi-faceted user preferences in a human-human dialogue with existing recommender engine technology in mind is next to impossible. The general rule is to modify user utterances as little as possible. However, the closer we get to implementing a speech-based dialogue system, the more restrictive we need to be with what sort of input we will accommodate. This is because, the longer the utterances, the harder it is to recognize and interpret them with contemporary speech recognizers.

Another issue related to both utterance length and content is when customers are very talkative and provide information that even a very advanced recommender engine will not be able to make use of. An example of this sort of user input as a response to a movie recommendation is shown in Figure 3.1. Recognizing and interpreting utterances C1 and C2 in Figure 3.1 is beyond the capabilities of the

S1	Have you seen *Star Wars*?
U1	Yeah, they are okay
S2	I see. Have you seen *The Matrix*?

Figure 3.2: Distilled version of the dialogue excerpt of Figure 3.1. Movie titles are in *italics*. S = System, U = User.

dialogue systems reported on in this thesis. Given that a speech recognizer succeeds in correctly recognizing and interpreting C1 and C2, the only really usable fact for future recommendations is that the user ratings for the episode I and II in the *Star Wars* series are less than average. Even if the recommender engine could utilize and model preferences concerning e.g. computer animation and special effects in movies, we would not be able to draw any conclusions about this from C1 and C2, since the user first states that the movie is lousy because of the "computer animated clown", but then says that the movies are good because of the "special effects". Thus, the distilled version of the dialogue in Figure 3.1 becomes quite severely changed (see Figure 3.2). This particular dialogue is even harder to accommodate correctly, since *Star Wars* is both one of currently six episodes in a series, as well as one unique movie in that series. It is not clear whether the recommender in R1 was originally referring to the movie or the series. We can view the excerpt in Figures 3.1 and 3.2 as an application of the MAPPING guideline (see Table 3.1) since neither the information nor the recommender engine of the system can handle the content of the original dialogue in Figure 3.1.

System Utterance Syntax

As the guidelines dictate, the system should not make use of ambiguous ellipses, since this may confuse the user. Repairing such misunderstandings requires complex dialogue tracking and makes the dialogue less efficient. Figure 3.3 shows an example of this from the dialogue corpus. The confusion in the dialogue in R7-C8 is avoided if we make sure that the system never uses ambiguous expressions such as R7.

R1 *Samuel L Jackson* acts in the following movies: `[list]`
C1 *Time to Kill* is good
R2 have you seen *51st State*?
C2 no
R3 do you want to put it on your to see list?
C3 what is it about?
R4 *51st State* is about `[plot]`
C4 no
R5 have you seen *Unbreakable*?
C5 is *Bruce Willis* starring in that?
R6 yes
C6 *Bruce Willis* has done many good movies / *The Sixth Sense* is great
R7 ok / he acts in *Star Wars*
C7 *Bruce Willis*!?
R8 no no / *Samuel L Jackson* / do you like science fiction?
C8 aha / yeah

Figure 3.3: Dialogue excerpt where the recommender (R) makes an ambiguous reference (R7), confusing the customer (C). Movie titles and actor names are in *italics*; domain information, such as title lists and plot information, from the database is omitted but marked with brackets.

User Initiative

Even though the recommender in the human-human dialogue generally tries to accommodate customer requests, she sometimes ignores customer requests and maintains initiative according to her plan. Since the recommender has a legitimate “expert” role, this behavior is accepted by both parts in the dialogue. However, in a human-machine dialogue this sort of behavior will probably be less easy to accept for the user. When distilling, we need to accommodate this and abide to the User Initiative guideline. That is, when the user takes initiative, the system should let her do that without question.

The typical example of this is when the user initiates an information request before responding to a pending system-initiated preference request.

R1	it seems that you like movies where the story is not chronological?
C1	yeah / like the story jumps back and forth with cool cuts
R2	right / then I think you'd like *Memento*

Figure 3.4: Dialogue excerpt with a complex preference attribute. Movie titles are in *italics.* R = Recommender, C = Customer.

"Do Not Diagnose What You Cannot Treat"

As pointed out by Höök [2000], user modeling applications should not include data that are of no relevance for the task being performed. That is, the system should not request user model properties that have no relevance for the recommendation task. This seems intuitive and clear, but when applying this guideline to the corpus we are required to know the workings of the recommender engine that we will put to use in the final conversational recommender system. Otherwise, we might remove too much from from the corpus, as well as remove too little. For example, one human recommender in the study asked for a customer's age. Demographic recommender systems (see Section 2.2.3) utilize age information in their user model, whereas e.g. CF systems do not. When distilling, we thus complement the guidelines with the relevance recommendation of Höök, as well as taking the chosen recommender engine into account.

Another example of preference conveying covering attributes that are hard to model is found in the excerpt of Figure 3.4. The preference in C1 in Figure 3.4 would require a knowledge-based recommender system with explicitly encoded information for "non-chronological storyline" and "cool cuts". Allowing for such—and other equally complex—attributes is perhaps possible, but it obviously requires a massive knowledge engineering effort.

Wasting Turns: Immediacy and Forgetfulness

Human recommenders tend to "waste turns" by not giving all relevant information at once, as the guidelines suggest. This is partly due to that it takes time for a human recommender to browse and overview the information in the movie database.

C1	do you have any Spanish movies
R1	Spanish? / uhm how do I find that //
C2	I think I've heard about a Spanish move called *Amores* something
R2	uhum / maybe
C2	supposed to be a gang movie
R3	sorry / what was the title?
C3	like *Amores Peros* I think

Figure 3.5: Dialogue excerpt showing how the dialogue can be affected by database browsing difficulties and human memory limitations. Movie titles are in *italics*. R = Recommender, C = Customer.

Figure 3.5 shows how this affects the dialogue (R1, R2). Another turn-waster is due to human memory limitations. Several dialogues contain utterances where the recommender asks the customer to remind her of a title or an actor name, as in Figure 3.5 (R3). While this dialogue behavior is natural between humans, it is not suitable in a human-machine dialogue according to the Forgetfulness guideline.

Orderliness: Is it there or not?

The Orderliness guideline has been applied with care, since it is hard to characterize the original corpus as having a quality of order. Human recommenders differ greatly in their strategy, and sudden "interruptions" from the users in form of preference volunteering and information requests cause the dialogue to take unpredicted ways. This mutual breaking of order from both recommender and customer seems to work out fine in the human-human context. The dialogue discussed here may require seemingly unordered skipping between issues in order to arrive to qualified movie suggestions. Thus, the recommended Orderliness guideline has been treated with moderation.

The Unbiased Recommender

In the excerpt in Figure 3.6 the recommender poses a leading question (R2), that makes it hard for the customer to disagree without "losing face". In R2 in Figure 3.6 the recommender indicates that the customer is "a bit childish" if she admits to

R1	have you seen *Harry Potter*?
C1	yeah /
R2	it was a bit childish right?
C2	uhm / yeah / I guess

Figure 3.6: Example of a leading recommender question unsuitable for an unbiased dialogue system. Movie titles are in *italics*. R = Recommender, C = Customer.

liking the movie *Harry Potter*. The response in C2 is very hesitant, indicating that the customer only agrees with the recommender in order not to seem childish. In effect, the inferred low "rating" of the movie in question thus might be incorrect and in turn affect future recommendations. In order to avoid such negative effects, leading questions are re-written as unbiased questions (e.g. "What did you think about it?" instead of R2).

A related issue on bias that is commonplace is when the recommender motivates her recommendations by claiming personal experience of the movie in question. This is fine in a human-human situation where the more knowledgeable recommender is the expert. Indeed, subjective explanations of this sort occur in the original corpus. However, in a human-computer situation users are probably less likely to accept a system that states *"I have seen this movie myself, and I enjoyed it very much"*. Thus, all recommendation explanations and motivations in the distilled corpus are based on objective attributes of the movie (such as genre and cast), as the NEUTRALITY guideline prescribes. This is in line with the RELEVANCE guideline, that suggests that the system should only present relevant information. In this case, this is done by using matching attributes in the current recommendation base in the explanation.

3.3 Characterization of Recommendation Dialogue

As a second step in the analysis, a number of characteristics are identified in the corpus. The characterization consists of: (a) the roles and attached initiatives, which have an impact on how the dialogue progresses (Section 3.3.1); (b) the relations between information requests and preferential statements (Section 3.3.2); (c) a list of re-occurring dialogue act types in the corpus (Section 3.3.3); and (d) a classification of two principal dialogue behaviors (Section 3.3.4).

3.3.1 Roles and Dialogue Flow

We start by defining a *recommendation dialogue* as an exchange of dialogue acts between two participants; one acting in a *recommender* role, and the other in a *customer* role (i.e. receiver of recommendations). The recommender is assumed to have extensive domain knowledge (such as access to a database of domain items), as well as a strategy for getting to know the customer's preferences, and a way of using this information in order to recommend relevant items. In a human-machine situation this translates naturally to the *system* having the recommender role, and the *user* having the customer role. Note the deliberate deviation from the roles in inquiry-oriented dialogue [Hulstijn, 2000]. Roles are slightly different since the distance between task and dialogue is closer than in traditional information retrieval dialogue [Dahlbäck, 1997]. Furthermore, *inquiry* is viewed as a part of recommendation dialogue, and *negotiation* plays a large role in recommendation dialogue. (See Section 2.1.5.)

Looking at the overall dialogue flow in a typical recommendation dialogue, we can distinguish three phases:

1. Establishing initial descriptive preferences.

2. Free exploration by query, and additional preference acquisition.

3. Refinement of preferences using comparatives and superlatives.

In phase 1, the recommender (i.e. a recommender system implementation) aims at establishing some basic preferences, preferably distributed over the majority of the domain's entity types (e.g. some preferred artists, some genres, and some album preferences in the music domain). Here, the initiative is mostly the recommender's who is guiding the customer (i.e. the user) to efficiently acquire preferences through direct questions.

The customer (or user) may then, in phase 2, take initiative and explore the domain by posing factual questions about the domain. In the dialogue corpus it is common that preference statements occur as a result of being exposed to query results. This is consistent with the observations of e.g. [Carberry *et al.*, 1999, page 187] who claim:

S1a These movies belong to the genre *Thriller*: `[list]`
S1b Any of these you like in particular?
U1 I like *The Usual Suspects* better than *The Silence of the Lambs*
S2a Ok.
S2b Are there any other genres, actors or directors you prefer?

Figure 3.7: Sample distilled dialogue excerpt from the dialogue corpus with factual statements and queries; and descriptive, comparative and superlative preferences. S = system, U = user.

> users are often unaware of their preferences at the outset of planning and only bring these preferences into play as they must evaluate alternative actions and choose among them.

When an initial set of preferences have been accumulated, preferences may be refined by introducing comparative statements in phase 3 (e.g. utterance U1 as response to S1a/S1b in Figure 3.7). It is noteworthy that preference statements in the early phases (1–2) typically deals with *classes* (entity types and values that describe attributes of domain items), whereas phase 3 typically deals with *instances* (individual domain items that are being recommended). Recommendations may occur in all three phases, but typically the recommender can only provide high-quality recommendations in phases 2 and 3 when enough preferences have been collected from the customer. Initiative in the third phase is not as clear-cut as in the previous two. The corpus indicates that about half of the recommenders re-gained more control over initiatives in phase 3 and asked customers' comparative questions. The other half simply acknowledged comparative preferences as they were stated by customers. For dialogue system strategy design, this behavior is thus an open choice using the human-human dialogue corpus as guideline.

The phases are not one-directional since they may overlap each other to a certain extent in the dialogue. Each phase may also occur several times in a longer dialogue. Furthermore, all phases are not mandatory in all preference dialogues (e.g. there may be dialogues without much exploration by query). The three phases serve as useful guidelines when designing a dialogue strategy that describe human-like preference

Table 3.2: Utterance content taxonomy. Percentage of utterances in the corpus.

Category	%	Sub-category	%	Sub-category	%
Task	79.3	Factual	28.6	Class	8.9
				Instance	19.7
		Preferential	50.7	Class	18.3
				Instance	32.4
Communication management	14.5				
Irrelevant	6.2				

dialogue behavior.

One observation on preference dialogues is that humans prefer to start out simple and then gradually refine factual queries/statements and preference statements in the on-going dialogue as opposed to construct complex utterances in one go. This should thus be supported in the dialogue strategy design.

With this bird's eye perspective of the general flow of a recommendation dialogue, we move on to utterance level to characterize requests and statements of dialogue participants.

3.3.2 Factual and Preference Requests and Statements

When examining the dialogue corpus at utterance level, it was found that 50.7% of the customer utterances in the dialogues were descriptive (32.7%), comparative (9.0%) or superlative (9.0%) *preferential utterances*. A smaller part, 28.6%, of the utterances were *factual utterances* about the domain and its entities. Preferential and factual utterances are considered to be the principal *task-related* utterances in preference dialogues. The remaining part consisted of communication management such as repeats, etc. (14.5%), and irrelevant utterances such as questions concerning the experiment situation, etc. (6.2%) [Johansson, 2003a]. Table 3.2 shows the distribution of utterances. As noted in Section 3.3.1, preference statements occur for both *classes* (e.g. *actor* or *genre* in the movie domain), and *instances* (e.g. *movie titles*).

It is interesting to note that information requests occur as part of the recommendation dialogue, often as a sub-dialogue to a preference interview. Indeed, it often drives the dialogue forward. This interleaving of factual requests and preference statements

R1 *Rene Russo* acts in these movies / any you like there? (R displays a list of movies on the screen)
C1 yeah / that one is great (C points at one of the titles on the list)
R2 I see / please name another good movie
C2 uhm / who's starring in *Ransom*
R3 here are all the actors in *Ransom*: (R shows list)
C3 so what other movies has *Mel Gibson* done?
R4 all of these (R points at Gibson's filmography list)
C4 right / oh yeah / *Braveheart* is one of my absolute favorites
R5 Oh then I think you'd like *Gladiator*

Figure 3.8: Dialogue excerpt showing how user-initiated information requests move the more general preference dialogue forward. The overall goal for the recommender is to retrieve movie preferences (R1, R2), based on movie titles from the filmography list of the actress *Rene Russo*. Subsequently, the customer initiates information requests (C2, C3), thereby retrieving another filmography list. Actions are denoted within parenthesis. Movie titles and actor names are in *italics*. R = Recommender, C = Customer (the excerpt is not distilled).

is exemplified in Figure 3.8.

3.3.3 Dialogue Acts

Task-related utterances in recommendation dialogues can be viewed in terms of traditional dialogue acts such as statements and information requests [Bunt, 1994]. Even though general and domain-independent dialogue act taxonomies have been suggested, they often seem too general for specific applications, especially as user model acquisition heuristics [Pohl *et al.*, 1995]. As hinted above, the division between factual and preferential acts is important and serves as a useful tool to categorize acts specific for the recommendation dialogue. In order to arrive at a design of a formalism specifically targeted for recommendation dialogue we identify the following acts[2]:

---

[2] Note that the focus is not on general domain-independent act types, but rather on the studied class of recommender systems. Conventional acts, such as ACKNOWLEDGE, OPENING, and CLOSING, etc. naturally occur in recommendation dialogues.

Factual-Request Requests take two distinct shapes. In the first sense, it is a question of factual nature (typically from the customer's part) about the domain. This is the information request in the traditional information retrieval dialogue system sense, where the system's task is to deliver a database result (as a FACTUAL-STATEMENT). This is a typical act found in inquiry-oriented dialogue.

Preference-Request In the second sense, the request is a preferential question from the recommender to the customer, where the goal is to acquire preferences as an answer from the customer. These PREFERENCE-REQUESTs are mostly descriptive, but occur as comparatives or superlatives in some dialogues. PREFERENCE-REQUESTs typically take three forms, ranging from specific to generic. The most specific form requests explicit entity type values (such as *"What do you think about thrillers?"*). In the second form, preferred entity types are requested in a more open-ended fashion where the user is free to give the preferred value, given an entity type (such as *"Please state a genre that you like"*). Third, open PREFERENCE-REQUESTs encourage the user to elicit both entity type as well as value. The open PREFERENCE-REQUEST may contain valid entity type suggestions (e.g. *"What other genre, actor, or director preferences do you have?"*, or be completely open (e.g. *"What else do you like?"*.

Answer As in the case of requests there are both factual and preferential ANSWERS. These are responses from the customer to PREFERENCE-REQUESTs from the recommender. Answering is an abstract act that can take several shapes: FACTUAL-STATEMENT, PREFERENCE-STATEMENT, and the simple YES/NO answer. FACTUAL-STATEMENTs as answer is most common for the recommender and PREFERENCE-STATEMENT is mostly a customer act. YES/NO answers exist for both roles.

Factual-Statement The FACTUAL-STATEMENT is a fundamental characteristic of information retrieval dialogue and is the standard response to a FACTUAL-REQUEST, typically carried out by the recommender. In an implementation, providing an answer from a database or other domain description is naturally the task of the system.

Preference-Statement *Comparative* PREFERENCE-STATEMENTs naturally refer to two entity types or entity values (arity 2), whereas *descriptive* and *superlative* statements refer to one entity type or value (arity 1). Naturally, this act is reserved for the customer in the studied recommendation situations. However, it does occur that human recommenders provide their own preferences as statements, e.g. before providing a recommendation. This is unsuitable for human-computer dialogues and has been removed in the distillation process. The reason PREFERENCE-STATEMENT is separate from the ANSWER category is that PREFERENCE-STATEMENTs also occur as volunteerings, i.e. without a preceding PREFERENCE-REQUEST. This is an important feature of preference detection in Carberry *et al.*'s theory [1999].

Agreement The abstract agreement act can be of two types: (ACCEPT or REJECT). These two are common in this domain as ANSWERs to RECOMMENDATIONs, and take the form of two acts in sequence[3]. The REJECT act is viewed as a NO combined with a PREFERENCE-STATEMENT (e.g. *"No. I don't like thrillers"*). The ACCEPT act is a YES (or ACKNOWLEDGMENT in some schemes), combined with a PREFERENCE-STATEMENT.

Recommendation The recommendation act is central to preference dialogues in recommendation situations, and is the goal of a recommender system. A RECOMMENDATION is invoked when the recommender has gathered "enough" preferences from the customer in order to present an entity that she believes the customer will like. However, RECOMMENDATION is an abstract act, since it can be realized as a QUESTION (*"Have you seen film x?"*), as a STATEMENT (*"You will probably like film x"*), or even as a sequence of the two (*"Have you seen film x? I think you will like it"*).

Motivation A recommendation is often accompanied with an explanation of why the recommendation was made. Motivations can help build trust between participants, since it helps the customer understand what preferences the recommender has picked up on. It also naturally relates to *transparency* in terms of human-computer interaction (see Section 2.2.1). This act is called MOTIVATION, and typically relates previously collected preferences to the recommended item. Table 3.3 exemplifies a

---

[3] They may also be viewed as a compound act, depending on how the encoding formalism of the act is designed.

Table 3.3: Dialogue excerpt with a RECOMMENDATION and a MOTIVATION dialogue act. The actor and genre preferences used in the MOTIVATION were previously detected in the dialogue. Movie titles and actor names are in *italics*. U = User, S = System.

	Utterance	**Dialogue Act**
S1	Have you seen *Entrapment (1999)*?	RECOMMENDATION
U1	No	ANSWER(YN)
S2	I think you will like it since it is a *thriller* starring *Sean Connery.*	MOTIVATION

RECOMMENDATION and a MOTIVATION expressed with regard to this particular user's previously detected preferences.

Compound Acts Certain utterances can best be described as being compounds of the dialogue acts listed above. The two AGREEMENT acts (ACCEPT and REJECT), can be viewed as either a compound, or a sequence of two acts. Another common compound is a merge of a preferential and a factual statement, as in *"I like comedies when I want to relax"*. The preferential part concerns the genre *comedy*, and the factual part concerns the situation *relax*.

As described in Chapter 2, the model presented by Carberry *et al.* [1999] describes three utterance types in which preferences are conveyed: **direct** (e.g. *"I like Bruce Willis"*), **indirect** (e.g. as part of queries; *"What thrillers are there?"*), and **hedging**, which signals an uncertain preference (e.g. *"I might like Pulp Fiction"*). Direct statements and hedgings falls into the descriptive PREFERENCE-STATEMENT category, whereas indirect statements belongs to the FACTUAL-QUESTION category. Carberry *et al.* focus on descriptive preferences and do not mention comparatives and superlatives in their model. However, we feel they should naturally be included in the direct PREFERENCE-STATEMENT category.

Connecting the findings in the corpus analysis with the theory of Carberry *et al.*, we provide an extension to the conversational circumstances category (see Section 2.2.2):

S1 *Star Wars: A New Hope* is a *Science Fiction* movie.
U1 Yes, I like *Science Fiction.*

Figure 3.9: Dialogue excerpt exemplifying the Accept-Solution conversational circumstance. S = System, U = User.

Accept-Solution: The user gives a preference as part of accepting a recommendation. Figure 3.9 shows an example. The preference strength is comparable to that of Volunteered-Background, and the Accept-Solution circumstance is affected by the utterance type in the same fashion, ranging from *mod-2* (3) to *str-2* (5)[4]. As endorsement, it has a reliability rating of *Very-Strong* (5)[5].

3.3.4 Delivery and Interview Dialogue Behaviors

With the completion of the characterization of recommendation dialogues, we have an empirical ground to start developing a computational model for recommendations in natural language dialogue. The first step is to define recommendation *dialogue strategies.* This is done by clustering the distilled corpus into re-occurring patterns. Two principal dialogue patterns were identified: **delivery**, and **interview** [Wärnestål *et al.*, 2007b]. The hypothesis is that a recommendation dialogue strategy model suitable for implementing conversational recommender systems can be seen as a combination of interview and delivery strategy instances of varying complexity.

The Delivery

The goal of a *delivery* is to present information. We identify two kinds of delivery: (a) direct delivery, and (b) indirect delivery. In the former case, a delivery simply consists of presenting a solution as a response to an explicit request (such as a traditional information retrieval dialogue system). In the latter, the delivery is due to an implicit request, or a long-term goal, influenced by preferences detected in the on-going dialogue. For instance, a recommender system user has the long-term goal of getting recommendations. However, it is in collaboration with the recommender

---

[4] See Figure 2.2.
[5] See Table 2.2.

agent that this is achieved, and the agent cannot be certain that the next recommendation is previously unknown to the user. This requires caution in terms of the delivery. Furthermore, the agent could motivate its choice of recommendation with a *motivation*, as well as follow-up questions regarding the quality of the given recommendation. Such motivations are central for building trust [Swearingen and Sinha, 2002] and help explaining the inner workings of the recommendation algorithm [Höök, 2000]. They are frequently used in recommendation situations and therefore desirable in conversational recommender system interaction. Asking the user for feedback on the delivered item is also considered part of the indirect delivery strategy.

At certain points, a delivery request may result in *exceptional results*. Exceptional results arise when the user has provided too little, ambiguous, or erroneous information; and needs guidance in order to achieve her goal. The amount and quality of such information depends on what background resources the deliverer has at her disposal. A dialogue system that employs a complete ontological model of the domain can, for instance, “know” that certain concepts are not covered by an underlying database (out-of-domain concepts). Other examples of exceptional results are empty result sets, or under-specified queries.

In our view of recommendation and inquiry-oriented dialogue, exceptional results is what drives the interview strategy. There is a symmetry between under-specified queries and resulting clarification dialogues, and preference acquisition for indirect delivery (such as recommendations). This similar structure makes it possible to use the interview strategy for both user preference interviews, as well as for sub-dialogue clarification for poorly understood requests. The generality of the interview and delivery strategies makes them suitable to use for both phenomena, as exemplified in Chapter 6.

It is important to note that the occurrences of exceptional results increases in human-computer dialogue compared to human-human ditto. A typical example is information about out-of-domain concepts which is a direct consequence of the limited domain descriptions in information systems to date.

S1 Is there a particular genre you like?
U1 I like *thrillers*
S2a These are the actors and directors that belong to the genre *Thriller*: `[list]`
S2b Any of these you like in particular?
U2 I like the actor *Kevin Spacey*
S3a These movies star the actor *Kevin Spacey*: `[list]`
S3b Any of these you like in particular?
U4 The first one is good
S4a Ok. Do you have any other preferences?
U4 What *drama* movies are there?
S4a These movies belong to the genre *Drama*: `[list]`
S4b Any of these you like in particular?

Figure 3.10: Preference constraint selection strategy in an interview. Example from the music domain. Entity values are in *italics*. S = system, U = user.

The Interview

The purpose of the *interview* is to collect information about opinions on domain entity types (e.g. genres, directors or actors in the movie domain) and specific items (i.e. specific movies). We refer to this information as *preferences*, which form the base for deliveries (e.g. product recommendations, or database search results). The user (interviewee) is assumed to respond cooperatively, but may also volunteer preferences not explicitly asked for. The question-selection strategy (i.e. the order in which entity type and value preferences are requested by the system) follows a certain default order, often ranked by a pre-defined importance in the domain and usually ends in "open" preference requests (e.g. utterance S4b in Figure 3.10). Our corpus analysis suggests a question-selection strategy that moves from generics to specifics (i.e. asking about genre preferences before asking about actor preferences). Note however, that the default order can be revised since interviewees may volunteer preferences in a different order and inform the interviewer of specific importance of certain attributes. Recommender systems that implement this kind of interview strategy is said to have a *dynamic* question-selection strategy (cf. [Bridge, 2002]), since the questions are chosen at run-time depending on what preferences have been given by the user. As the dialogue progresses it may become impossible to provide more deliveries based

S1a There are no more movies matching the current criteria.
S1b Would you like to ignore any director preferences?
U2 Yes, skip all directors
S2a Ok, I have a recommendation ready for you.
S2b I think you will like the movie *The Usual Suspects*.

Figure 3.11: Relaxing a specific constraint in a preference interview (S1b), which is followed by a delivery (S2b). Example from the movie domain. S = system, U = user.

on the current preference model. The system then takes on the interview strategy, but tries to relax the constraints. When asking for relaxations, the system uses the inverse order in which attributes were requested in the constrain strategy. Figure 3.11 shows a sample relaxation interview from the movie domain. When the preference requests on the interviewer's agenda has been fulfilled and the *resource* responsible for reaching a solution, a delivery can be made. This depends on the task and the nature of the resource. In some cases the interviewer has a fixed agenda stating which attributes that need values supplied by the interviewee. The dialogue then progresses with repeated constrain requests in a slot-filling fashion. When all slots have been filled a delivery (typically in the form of a database result set) can be made. This strategy is standard for information retrieval dialogue systems.

In other cases the requests for constraints are more dynamic. For instance, in recommender systems the number and nature of the "slots" that need to be filled depends on the interviewee's preferences. Consider a conversational movie recommender system. For one interviewee (the user) it might be enough for the interviewer (the system) to ask for a handful constraints[6] if her preferences are narrow enough to quickly reach high-quality movie predictions. For another interviewee (e.g. one with "too normal" preferences that does not make her easy to place in a *collaborative filtering neighborhood*) the interviewer might have to keep constraining for several turns until the recommendation engine is ready to provide a movie prediction.

---

[6] This depends on the nature of the recommender engine. It might mean a dozen or more movie titles in a collaborative filtering engine; or perhaps one or two genre preferences and a few actor preferences if the engine has a content-based flavor.

3.4 Summary

This chapter described Study I, where a human-human corpus was collected and analyzed using a systematic method to re-write the corpus into human-computer recommendation dialogues. The analysis revealed three phases of recommendation with different uses of descriptive and comparative preference statements, a dialogue act classification, as well as two clusters of behavior: interview and delivery. This forms the base for designing and implementing a dialogue strategy model that can be implemented and evaluated in a prototype conversational recommender system—which is the topic of the next chapter.

4

Design and Verification of a Recommendation Dialogue Strategy Model

First, we describe a dialogue strategy implementation based on the analysis of the previous chapter. Second, an end-user evaluation of the implementation is described. The result is a verification of the effectiveness and usability of the dialogue strategy design. Furthermore, we find implications for "conversational impetus", variation of motivation strategy, and domain exploration support, that will influence the refinement of the recommendation dialogue strategy model put forward in the following chapters.

THIS CHAPTER DESCRIBES a basic recommendation dialogue control strategy in the movie domain, based on the characteristics of the analyzed corpus from Study I in Section 4.1. Second, and in line with an iterative development approach, we are interested in quickly getting a running system which can be used in an end-user evaluation, which is the topic of Section 4.2. The results of the study are then discussed in Section 4.3.

4.1 Movie Recommendation Dialogue Control

As detailed in Chapter 3, the recommendation dialogues in the corpus can be viewed as interviews and deliveries consisting of a combination of (a) system-driven preference requests, (b) user-driven information requests, (c) user deliveries of preferences, and (d) system deliveries of information and recommendations. The mixed-initiative character of the dialogue can be said to correspond to a seamless integration of these initiative types. Based on this assumption, we can define a basic dialogue control structure for managing recommendation dialogues in the movie domain.

4.1.1 System Initiative

The system-driven preference interview and indirect recommendation delivery strategy is implemented as a hierarchical Harel statechart [Harel, 1987], where black dots denote entry nodes, circled black dots denote exit nodes, rounded squares denote super- and sub-states, and the circled H denotes a "history" node that keeps track of the encapsulating node's current state. Each node corresponds to a system prompt (with either canned text, or a template that is filled by back-end resources), and transitions correspond to influences of various types. Figure 4.1 shows a graph that corresponds to initiative types (a), (c), and (d) above.

Initiating the Recommendation Dialogue

When starting the statechart execution, a canned welcome message is produced before traversing to the InitRecBase node. In InitRecBase, a "recommendation base" is established, which is the principal attribute set that future recommendations will be based on. There are several possible responses to the InitRecBase depending on what attribute the user prefers. Most users want to base their recommendations on genre (e.g. a *drama*, *comedy*, or *action* movie), whereas some users aim for movies starring their favorite actor (e.g. *"I would like a movie starring Cary Grant please"*).

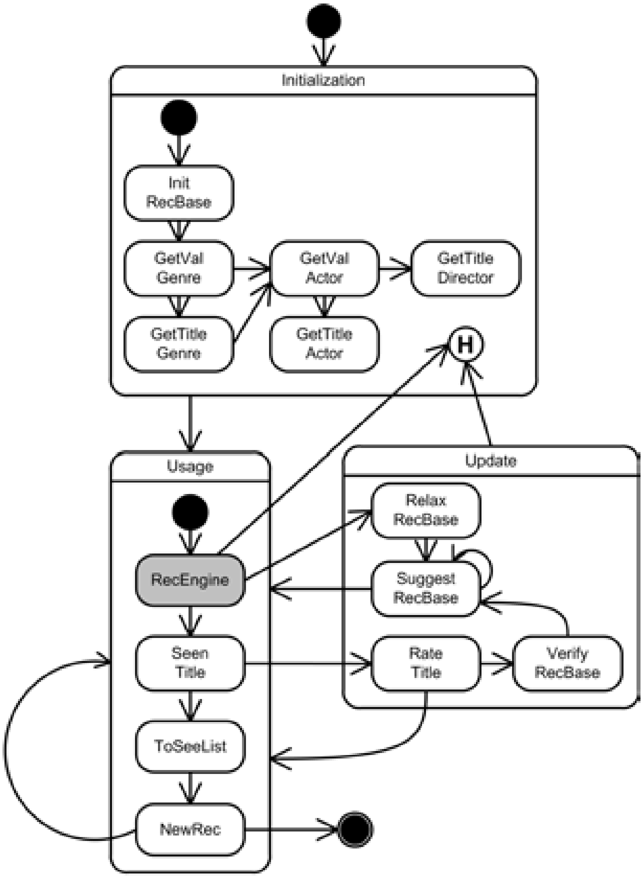


Figure 4.1: A recommendation dialogue statechart graph covering dialogue flows of the 24 distilled dialogues from the movie recommendation dialogue corpus. Sub-states correspond to system utterances (with the exception of the shaded RecEngine node which "silently" queries the recommender engine to determine the next system move), whereas super-states correspond to user model initialization, user model usage, and user model update from a dialogue point of view. Transitions are influenced by user utterances, previously recorded user preferences, and database content. (Transition conditions are not shown in the graph for readability reasons.)

Getting Attribute Values

GetValGenre is responsible for trying to assess what genre(s) the user is interested in. The GetValActor node functions in a similar way, asking the user for names of their favorite actors or actresses. The information retrieved by these two GetVal nodes is integrated in the recommendation base.

Acquiring Title Ratings

A central issue when utilizing recommender engines is to acquire title ratings from the user [Rashid *et al.*, 2002]. The more titles that are included in the user preference model, the better recommendations the engine can provide. Furthermore, the system needs some way of keeping track of which movies the user has seen, so the system does not recommend them again.

Thus, we have three GetTitle nodes, each based on one of the attributes *genre, actor*, and *director*. The typical GetTitle node usage is when the user has provided an attribute value (such as the name of an actor). The system then provides the user with a list of titles matching the given attribute values and asks her to identify movies that she likes. Note that this list is a non-personalized list and not a recommendation set. The GetTitle nodes typically occur *before* any requests have been passed to the recommendation engine. Interleaved information requests can influence how the lists turn out (such as the excerpt in Figure 3.8). Thus, there is no hard connection between the GetTitle node and the current recommendation base, since the titles in the list at any given moment do not need to reflect the recommendation base. This serves two purposes. First, we do not decrease the user's freedom of posing information requests, and indeed utilize these in the recommendation task. Second, it is good for the user preference profile to be as diverse as possible and not only include ratings for movies matching the current recommendation base.

RateTitle comes into function *after* a recommendation has been proposed. Its function is to extract the rating of an already seen recommended movie, so that we constructively can utilize an otherwise "useless" recommendation, while maintaining a conversational tone in the interaction.

S1	Have you seen *The Fifth Element*
U1	yeah / awesome
S2	It seems like we have covered all movies. Is there any other kind of movie you would like to watch?
U2	uhm / are there any movies directed by *Oliver Stone*?

Figure 4.2: Dialogue excerpt showing how Acorn suggests a relaxation of the recommendation base when the matching titles have been exhausted. Movie titles and director names are in *italics*. S = System, U = User.

Delivering Recommendations

SeenTitle is one of the central nodes in the usage situation, since this is where the system presents a movie suggestion to the user. The corresponding system utterance for this node is "Have you seen this movie?" along with the title of the highest ranked recommendation. All nodes that have arches leading to SeenTitle need to pass a check[1], since there are cases where it is not possible to traverse to SeenTitle (i.e. perform a recommendation). This depends on the chosen recommendation engine. The SeenTitle node is thus called only if the recommendation engine is able to deliver a suggestion. Otherwise, there is a need to continue to get ratings from the user (by returning to an appropriate GetTitle node), or to change the current recommendation base.

Handling Changes

As pointed out above, the user may change the recommendation base. A change in the recommendation base can also arise from the system's part (e.g. to relax the constraints posed by the current recommendation base). The excerpt in Figure 4.2 shows an example of how the system suggests to change the recommendation base. In terms of network traversing, S1 is an instantiation of the SeenTitle node. The response in U1 is a positive rating of the recommended title, causing the system to return to the RecEngine node to perform another suggestion based on the current recommendation base. Now, since all movies based on the current recommendation base have been considered, we traverse to the RelaxRecBase node (S2). From

---

[1] This check is represented as the RecEngine node in Figure 4.1.

this node there are several options, depending on the user's response. Since the user provides a new recommendation base (recommendations should henceforth be based on the director in U2) the system moves to the GETTITLEDIRECTOR node according to Figure 4.1.

Managing Recommendation Dialogue

In case the suggested title in a SEENTITLE node is indeed unseen by the user, we have a potential recommendation delivery. The system now needs to explain, or motivate, the recommendation objectively following the theory of building trust [Buczak *et al.*, 2002], and according to the findings in the dialogue corpus. This is done in the TOSEE node, which (a) generates an explanation by relating to the matching attributes in the current recommendation base, and (b) provides the user with the option of putting the recommended movie on the recommendation protocol. In case the user declines, the system needs to verify the current recommendation base, since this response is interpreted as negative feedback to the recommendation. On the other hand, if the user responds positively, we have a successful recommendation. The system can then add the recommended movie to the protocol and move on.

After a successful recommendation has been made the system asks if the user wants a new recommendation in the NEWREC node. A wide range of responses may follow this question. A simple *"no"* indicates that the session is terminated (moving the EXIT node), whereas a *"yes"* is equally easy to handle, since we simply test if we can go to the SEENTITLE node to perform a new recommendation (after passing the RECENGINE check). However, the user may also change the recommendation base if she decides to continue the dialogue. It is easy to assume that this is because the users want variation in a set of recommendations in a session and desires e.g. one action movie, one drama comedy starring their favorite actor, and one animated movie. Example responses to the question *"Would you like a new recommendation?"* include:

- *"Yes, something like Gladiator please."*
- *"A drama starring Mel Gibson would be nice."*
- *"Do you have any animated movies?"*

- *"Sure, give me movies directed by Ridley Scott."*

In the case of a changed recommendation base, we traverse to the appropriate GetTitle node (depending on which attribute(s) has been changed), in order to get a complete picture of any modifying attributes to the new recommendation base before moving on to a new SeenTitle node.

Influencing Transitions

As evident in Figure 4.1, several nodes have multiple arches branching to different nodes. It was discovered early in the distillation process that similar system preference queries can be responded to in very different ways.

By comparing the surrounding dialogue context and taking into account how long dialogues have progressed (i.e. how many previous preference requests had been completed), and the available information from the database, three ways of influencing the network node transition are identified:

1. User utterances
2. User preference model
3. Database content and recommendation base

User Utterances The first—and most obvious—way to guide transitions is to take the content of the user's utterance to a system query into account. This is done by having each node check the interpreted utterance and decide which node to traverse to next. The content of the user utterance is thus the most important as well as straight-forward way to influence dialogue node transitions. However, while this is the default and most common transition influence, there are cases where the content of a user utterance may yield two (or more) equally valid system responses. We then need to consider other parameters.

User Preference Model One alternative parameter is what the recommender agent believes to be true about the user's movie preferences. This reflects that the recommender needs to know a number of preferences (ideally covering both positive and negative preferences about the bulk of all available attributes) before a qualified

U1	I would like a drama please
S1	Can you please say a drama movie that you like?
U2	yes / *Schindler's List*
S2	Ok. Do you have any other favorite movies?
U3	what movies have *Luc Besson* directed?
S3	*Luc Besson* is the director of these movies [displays a list of movies]
U4	*The Big Blue* is very good
S4	Ok. Do you have any other favorite movies?
U5	I liked that one as well [selects another movie from the list]
S5	Ok. Have you seen *Jeanne D'Arc*?

Figure 4.3: Dialogue excerpt showing how the system fails the RecEngine check twice (S2, S4) and continues to ask the user for movie ratings. In S5, we reach SeenTitle since RecEngine is passed, since the recommendation engine now has enough data to provide a recommendation. The established recommendation base consists of dramas (U1). Movie titles and director names are in *italics*. S = System, U = User.

recommendation can be issued. It seems sound to assume that the recommender utilizes previously known preferences about movies, actors, and genres to dictate his or her next utterance.

In *recommender system* terms, this relates to the *density* and *size* of the user preference model (see Section 2.2.3). Concretely, a CF recommendation system is not able to calculate any prediction scores unless the user preference model has reached a certain density and size[2]. Figure 4.3 shows how the system returns to the GetTitleGenre node after failing the RecEngine check due to an incomplete user preference model. Thus, the size and content of the user preference model serves as an input to the dialogue nodes' transition decisions. In Figure 4.1, this is shown as the dashed arch from the RecEngine node to the GetTitle node.

Database Content and Exhausted Recommendation Base A third, but still important, issue is when the recommender realizes that the user's preferences takes

---

[2] A CF system is not required to cover any other attribute than titles, which is not a strategy typically employed by a human recommender.

the form of too demanding *constraints* for the search for movies. This is true both for regular database queries, and recommendations. It then often happens that the recommender asks the user to relax these constraints. This happens both when an information query from the user is too narrow, or when all movies matching the current recommendation base have been considered.

When there are no matching movies—or when all movies matching a specific preference set have been considered—in the dialogue, the system must have ways to proceed if the user does not take initiative and starts introducing new preferences or search constraints.

An exhausted recommendation base can thus be the reason for traversing to a RelaxRecBase node instead of a new SeenTitle node (see Figures 4.2 and 4.1).

4.1.2 User Initiative

Hitherto we have focused on system-driven preference requests and recommendations. However, as noted above, a recommendation dialogue control structure will also have to accommodate user-driven information requests. Fortunately for our rapid prototyping approach, there is a fairly large body of research addressing exactly this issue. One such initiative is the phase-based PGP design pattern[3] that allows for information-providing dialogue system construction [Degerstedt and Johansson, 2003].

The dialogue strategy presented above has been implemented in the Acorn movie recommender system[4] by adopting the PGP pattern and integrating the finite-state recommendation dialogue network with the information-providing capabilities [Johansson, 2004]. Each node in the graph in Figure 4.1 thus holds the same basic phase-based information-providing machinery, so that users can issue information requests at any time in the underlying system-driven dialogue, as the empirical corpus findings dictate.

---

[3] PGP is hosted at the NlpFarm open source initiative (`http://herd.ida.liu.se/nlpfarm/`) as part of the MOLINC component.

[4] Acorn is described in more detail in Section 4.2.2.

4.2 Evaluation

The evaluation presented in this section has as its purpose to verify the dialogue strategy model described above. The evaluation is inspired by the PARADISE framework, as well as Hulstijn's evaluation properties [1999] (see Section 2.4.3).

The following nine aspects of user satisfaction with ACORN was measured:

- Task ease: How well the user feels that a particular task can be carried out with the system.
- Adaptation: How well the system adapts to an individual user's preferences.
- System response time and pace: How fast the system responds, and whether the interaction pace feels satisfactory for the user.
- Domain coverage: Measures whether there are enough items in the domain to solve a task, and whether there is enough information about each item.
- Interpretation performance: The user's experience of how well the system understands her input.
- Generation performance: How well the system performs when generating linguistic responses (phrase choice, clarity, and verbosity).
- Expected behavior: Measures how intuitive and natural the dialogue interaction is, in terms of initiative and grounding etc.
- Entertainment value: Assesses how entertaining and interesting it is to engage in a dialogue with the system.
- Future use: Whether it is likely that the user will use the system in the future or not.

These factors are assessed by analyzing (a) dialogues from the user sessions, and (b) a post-study questionnaire filled out by each of the participants.

In this study the set of user satisfaction aspects—and corresponding questions in the questionnaire—were enhanced in order to address e.g. entertainment value and adaptation assessment.

4.2.1 Participants

Twenty participants of varying age, gender, and background were recruited as users. None of them had any special knowledge of dialogue systems[5], but were all proficient computer users. They were not paid.

4.2.2 Acorn

Acorn is a text-based dialogue system built specifically to implement and evaluate a recommendation dialogue strategy in the movie domain. It is programmed in Java[6] with a MySql[7] database back-end.

The user interface (see Figure 4.4) consists of a chat-style panel where the dialogue between Acorn and the user takes place and a text field where the user types her input. To the right of the chat panel is a result presentation panel where movie information and other pieces of domain information is displayed.

Acorn's architecture is phase-based [Degerstedt and Johansson, 2003] and builds on the molinc component[8]. The main components of Acorn are: a **dialogue manager** (implementing the recommendation dialogue control strategy described in Section 4.1), a **domain knowledge manager** (including a hybrid collaborative filtering and content-based recommendation engine and a movie database), and a **preference manager** (described below). The **Linguistic Analysis** phase uses a parser module that produces a task representation of the user utterance. In the **Pragmatic Interpretation** phase a refined interpretation based on dialogue context is carried out by using a dialogue memory to add/change information in the task representation. The Pragmatic Interpretation phase is required for simple sub-dialogue capabilities, such as asking for clarifications or additional information, or refinements if the database returns too many or no hits. The **Task Handling** phase executes the task by retrieving information from the database. The result set is transformed to suitable output in the **Generation** phase with slot-filling templates.

Whenever an information request has been addressed by Acorn, the preference and recommendation dialogue continues to gather preferences and provide recommen-

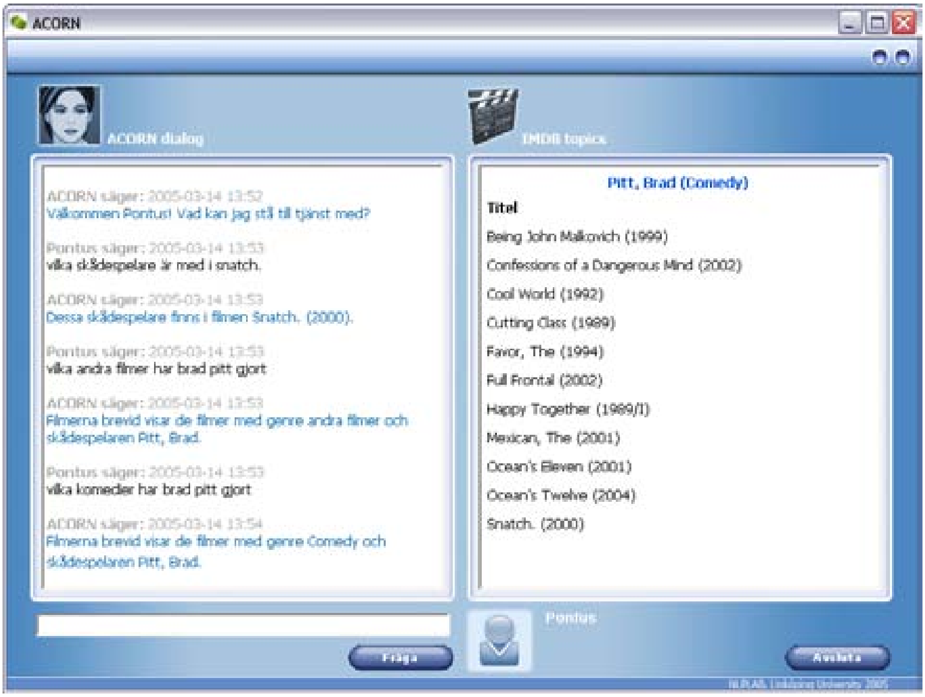
---

[5] None of the participants had previously engaged in Study I.

[6] JDK 1.4

[7] MySql 4.3

[8] Available at `http://herd.ida.liu.se/nlpfarm/`.



ACORN säger: 2005-03-14 13:52
Välkommen Pontus! Vad kan jag stå till tjänst med?

Pontus säger: 2005-03-14 13:53
vilka skådespelare är med i snatch.

ACORN säger: 2005-03-14 13:53
Dessa skådespelare finns i filmen Snatch. (2000).

Pontus säger: 2005-03-14 13:53
vilka andra filmer har brad pitt gjort

ACORN säger: 2005-03-14 13:53
Filmerna bredvid visar de filmer med genre andra filmer och skådespelaren Pitt, Brad.

Pontus säger: 2005-03-14 13:53
vilka komedier har brad pitt gjort

ACORN säger: 2005-03-14 13:54
Filmerna bredvid visar de filmer med genre Comedy och skådespelaren Pitt, Brad.

Pitt, Brad (Comedy)

Titel

Being John Malkovich (1999)
Confessions of a Dangerous Mind (2002)
Cool World (1992)
Cutting Class (1989)
Favor, The (1994)
Full Frontal (2002)
Happy Together (1989/I)
Mexican, The (2001)
Ocean's Eleven (2001)
Ocean's Twelve (2004)
Snatch. (2000)

Figure 4.4: Acorn's graphical user interface.

dations, until a new information request is detected.

Acorn's back-end part consists of a hybrid CF server[9], and a movie information database holding information on actors, genres, directors, and plot information. The database is used both to accommodate information requests, as well as providing attributes for the recommendation base. The recommendation engine is thus a hybrid engine [Burke, 2002], since it utilizes both the CF server as well as the domain-dependent database.

Assessing ratings is a tricky issue, as we need to convert natural language judgments about movies to some sort of ordinal scale. Acorn utilizes a rather simple approach. The grammar allows users to use a variety of judgmental descriptions of movies, ranging from single-word utterances (e.g. *"good"*, *"excellent"*, *"dreadful"*, etc.) to multi-word expressions (e.g. *"I really like this"*, *"*Gone With the Wind *is fantastic"*, etc.). These words and phrases are based on the utterances in the distilled dialogue corpus. As noted in Chapter 2, the problem of mapping natural language statements to numerical sentiment is naturally the "fuzziness" of language.

The approach adopted in Acorn is a vast simplification of the preference-detection theory of Carberry *et al.* [1999]. ACORN records only direct preference statements (that is, hedgings and indirect statements leave no trace in the preference model). Furthermore, reliability measures and closeness-of-match is ignored (see Section 2.2.2). This approach does not accommodate comparative statements, nor does it model factual questions as indirect preference statements. This basic approach is clearly not scalable for long-term end-usage; but given the purpose of Acorn as a prototype for this user evaluation study, it is sufficient and quick to implement. Figure 4.5 exemplifies dialogue interaction in Acorn.

4.2.3 Procedure

Each participant received a quick tutorial explaining rudimentary facts about Acorn (e.g. it being a movie information and recommendation system, that the interaction is typed, and that the language of interaction is Swedish, etc.). Next, the participant was presented with a short scenario consisting of three sub-tasks (see Appendix B).

---

[9] The user rating matrix is provided by the GroupLens (`http://www.grouplens.org`) research group.

U1	I would like a movie recommendation please
S1	Okay. What kind of movie do you feel like?
U2	I feel like watching a thriller
S2	Can you tell me one of your favorite thriller movies?
U3	Uhm, I'm not sure
S3	Okay. Can you tell me one of your favorite actors or actresses?
U4	I think *Bruce Willis* is good
S4	I see. Have you seen *Pulp Fiction*?
U5	Yes
S5	What did you think about it?

Figure 4.5: Sample recommendation dialogue about movies. Movie titles and actor names are in *italics*. U = User, S = System.

The tasks ranged from strict (tasks 1 and 3) to a bit more open-ended (task 2) in order to ensure that the sessions are comparable (cf. [Walker *et al.*, 1998]), but at the same time allow for some variety in the solutions. In addition to the comparable quality, strict tasks are also a way to determine efficiency (i.e. that a specific task is resolved), since users have no reason to persist in artificial constructed open-ended tasks and may settle for almost anything if there is no "real" or personal motivation. During the dialogue session, the participants noted their solutions/results of each task on a protocol. After completing the scenario, they were asked to fill out the questionnaire. The questionnaire consists of 23 questions regarding user attitudes toward task solution, system performance, and dialogue interaction. Response values are encoded on an ordinal scale of 1–4 corresponding to the statements *I strongly disagree* (1), *I somewhat disagree* (2), *I somewhat agree* (3), and *I strongly agree* (4). The sessions were also logged during the interaction and time-stamped and saved on file.

4.3 Results

The study yielded two kinds of data. First, the dialogue logs constitute data for a dialogue corpus analysis. Second, the questionnaire responses provides data for a user satisfaction analysis.

4.3.1 Dialogue Corpus Analysis

Session logs of the interactions resulted in a corpus with a total of 226 complete turns, and a total elapsed time of 4 hours (mean 12 minutes per dialogue). The dialogues are in Swedish, and the excerpts presented herein have been translated to English. The corpus was annotated manually with the number of system interpretation failures, and the number of system restarts. Furthermore, each dialogue was compared to an "optimal" scenario solution which represent the shortest number of turns that are required to solve all sub-tasks in the scenario. The scenario can be completed in seven turns, which is the "key" to an efficient dialogue. However, since the scenario can be resolved in a variety of ways, an additional turn or two may still feel both optimal and natural for a particular user. All twenty subjects accomplished all tasks in the scenario. The average number of turns for completing the tasks is 11.3, and 10 subjects accomplished the scenario within the optimal range (7–9 turns, depending on their strategy, and personal choice in the open-ended task).

Dialogues longer than 7–9 turns are the result of (i) system interpretation failures (due to a variety of factors, such as lack of linguistic coverage on the system's part, or on uncooperative behavior or misspellings on the user's part, etc.); (ii) domain exploration (e.g. asking for more recommendations, or additional information not required by the scenario); or (iii) miscellaneous turn types, such as clarification sub-dialogues due to too many database hits, etc. Table 4.1 shows some interesting aspects of the data, which will be discussed below.

Interpretation failure turns (FAL) are defined as turns that were not successfully parsed by the system, i.e. not covered by ACORN's linguistic resources. The total interpretation failure rate for the complete corpus is 1.2 turns per dialogue, with twelve completely error-free dialogues.

Domain exploring turns (EXP) are requests that are "not necessary" in order to complete scenarios. Examples include asking for plot information even though no task in the scenario requires that kind of information, or asking for more recommendations than required. Nine of the users used one or more turns for such additional domain exploring. The dialogue excerpt in Figure 4.6 is an example of when the user asks for more information and recommendations than required by the scenario.

Table 4.1: Total number of dialogue turns (TRN), number of interpretation failure turns (FAL), domain exploring turns (EXP), miscellaneous turns (MSC), and mean user satisfaction over the nine categories from the questionnaire (SAT).

User	**TRN**	**FAL**	**EXP**	**MSC**	**SAT**
01	14	6	0	1	2.5
02	7	0	0	0	3.1
03	9	1	1	0	2.7
04	14	0	2	5	3.2
05	9	1	0	1	3.3
06	12	1	0	4	3.2
07	15	0	5	3	3.3
08	17	1	5	4	2.7
09	17	0	6	4	3.2
10	7	0	0	0	3.3
11	7	0	0	0	2.9
12	7	0	0	0	2.9
13	7	0	0	0	3.3
14	8	0	0	1	3.0
15	23	7	1	8	2.6
16	10	0	0	3	2.8
17	9	2	0	0	3.0
18	9	0	1	1	3.0
19	12	2	2	1	2.8
20	13	3	3	0	3.0
Total	226	24	26	36	N/A
Mean	11.3	1.2	1.3	1.8	3.0

U1	please recommend a drama with *Sean Connery*
S1	Have you seen the movie *Action of the Tiger (1957)*?
U2	no
S2	*Action of the Tiger (1957)* is a drama starring *Sean Connery*. Do you want another recommendation?
U3	yes please
S3	Have you seen the movie *Anderson Tapes, The (1971)*?
U4	what is it about?
S4	Here is the plot for *Anderson Tapes, The (1971)*. Have you seen it?
U5	no
S5	*Anderson Tapes, The (1971)* is a drama starring *Sean Connery*. Do you want another recommendation?
U6	yes a comedy with *Adam Sandler*

Figure 4.6: Excerpt where the user deviates from the scenario requirements in order to explore the domain. The turns U3–S5 are clearly outside the scope of the scenario since it only requires one recommendation based on the attributes in U1, and no additional information requests such as U4. The dialogue has been translated from Swedish. S = System, U = User.

Miscellaneous turns (MSC) include a variety of turns, and have deliberately been merged together for the purpose of this analysis since they are not in focus here. These turns include e.g. sub-dialogue clarifications when the database returns none or too many hits. Another example of turns in this category arises due to the constructed and artificial nature of the scenario: A user may for example not like the actor Adam Sandler, and may dislike his comedies even more; and this has an impact on the dialogue of Task 3 (see Appendix B), since users may respond negatively to the resulting recommendation. This causes Acorn to spend turns trying to find out what is wrong with its preference model of the user. This phenomenon would not arise in "real" situations because users disliking Adam Sandler's comedies would not ask for such recommendations.

User satisfaction (SAT) is a metric that consists of mean values of the questionnaire responses (graded 1–4) for each of the nine aspects. The mean values should naturally be interpreted with care, since the questionnaire provides an ordinal scale.

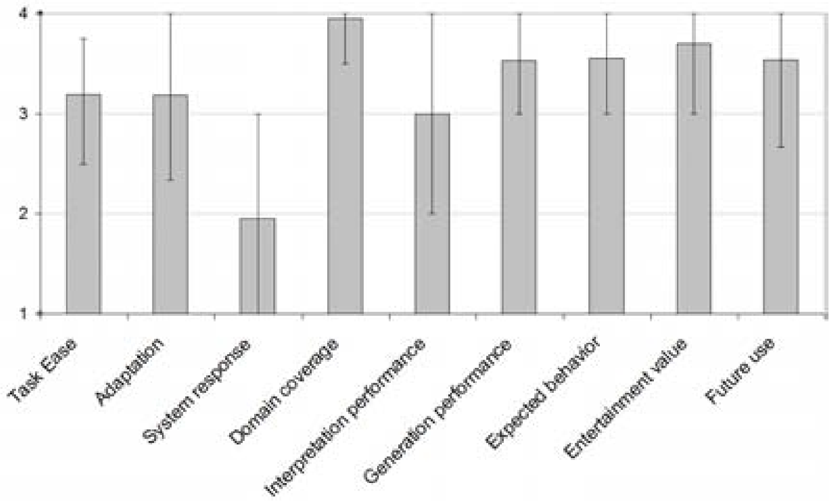


Figure 4.7: User satisfaction based on the subjects' own assessment split into nine categories based on the post-study questionnaire. Response values are: *I strongly disagree* (1), *I somewhat disagree* (2), *I somewhat agree* (3), and *I strongly agree* (4).

However, they are still useful indicators for the upcoming user satisfaction analysis.

4.3.2 User Satisfaction Analysis

The 23 questionnaire items were grouped into the nine categories, and the responses were weighed together. For example, for the category Adaptation users responded to the following questionnaire:

1. ACORN's responses were relevant and helped me solve the tasks.
2. ACORN's recommendations were effective and matched the preferences I had put in.
3. ACORN adapted continuously to my preferences.

Figure 4.7 shows the results of each of the nine user satisfaction categories for all twenty users.

It is noteworthy that nine out of twenty users engage in domain exploration dialogues. This should be considered rather high, since the users were *not* instructed or even encouraged to engage in such dialogues. Domain Coverage (mean score 3.9) and Entertainment Value (3.7) are the two highest-ranking aspects, and users engaging in exploration turns give the highest entertainment value ratings.

In the higher range we also find Expected Behavior (3.6), Generation Performance (3.5) and Future Use (3.5). Adaptation (3.2) is slightly lower, and could be affected by that the given scenario contains tasks that do not fit certain users, such as the given choice of actors and genres in task 3. Another important factor to consider is that adaptation typically requires more long-term use than just one session.

System Response Time (2.0) is significantly lower than the other rankings and is due to the slow response-time because of the large database.

4.4 Discussion

The study shows that ACORN's dialogue strategy allows for *efficient* dialogues, since all users accomplished the scenario, and that several even completed the tasks within the optimal number of turns. This capability may be seen as a prerequisite for conversational recommender systems, not to say for dialogue systems in general.

The low interpretation error rate would suggest that the user satisfaction rates are indeed measuring the desired aspects, without them being influenced by a general dissatisfaction with interpretation performance of the system.

One interesting observation is that the dialogue interaction has an *entertaining* quality. The number of domain exploring initiatives suggests that users finds the interaction interesting enough to deviate from the scenario, and engage in the dialogue out of personal interest. Exploratory behavior seems to happen toward the middle or end of the dialogue sessions, which indicate that such behavior is not only an attempt to familiarize with the system. Users engaging in exploration turns also seem to give the highest Entertainment Value ratings. This suggests that exploring the large domain space is an entertaining feature of interacting with ACORN. The questionnaire analysis shows that this is one of the most high-ranked satisfaction aspects. It is thus clear that dialogues longer than the “optimal efficiency” may have a high user satisfaction value attached. This is especially true in exploratory dialogues where it is a quality in itself for the user to be able to learn more about the domain than initially planned. Obviously, this comes with the domain and the purpose of the system: Exploring the movie domain and receiving personalized recommendations is different from e.g. finding train table information.

As pointed out in Chapter 2, previous research suggests that an important reason for investigating and developing conversational recommender systems is that they allow the system to capture user preferences when users are as motivated as possible to give them. Since most users are not explicitly aware of all their preferences at the outset of a dialogue session, the system should trigger preference volunteering. It is mostly when exposed to information that the user brings her preferences into play [Carberry *et al.*, 1999].

This triggering by information exposure we call *conversational impetus* since this could help drive the preference dialogue forward [Wärnestål, 2005]. An implication of this study is thus to integrate a mechanism for such conversational impetus in a recommender agent's dialogue strategy.

An interesting fact is that the users' opinion on Interpretation Performance according to the questionnaire seems rather low compared to the factual number of interpretation failures (FAL rates in Table 4.1). When examining these dialogues in detail it seems that users that encountered an interpretation failure in the *beginning* of their session, did not read ACORN's responses as carefully in the remainder of the session. An example of this is found in Figure 4.8 where utterance U1 was not understood by ACORN, resulting in the help message S1. After the additional help message the user does not seem to care much about ACORN's prompts, and fails to realize that her request has been addressed correctly. This stresses the importance of interpretation performance since a lack thereof takes away the feeling of interacting with a competent conversational partner, and this causes the user to pay less attention to the system's utterances and dialogue. Another possible explanation for this behavior is concerned with *when* to issue explanations. Objective explanations of recommendations serve two key points in the interaction: to provide transparency of the system logic, and inspire trust between the user and system [Swearingen and Sinha, 2002]. In the original human-human corpus—on which ACORN's dialogue strategy is based (see Chapter 3)—utterance sequences such as S2-U3-S3 in Figure 4.9 are standard. That is, the recommender provides an explanation of the recommendation

U1	I want to know about *Brad Pitt*
S1	To be able to provide recommendations I need to know about your movie preferences. Say for example: “I would like a drama”.
U2	a movie starring *Brad Pitt*
S2	Ok. I need more information in order to give you a recommendation. You can for example provide a genre you’re interested in.
U3	comedy
S3	Have you seen the movie *Snatch (2000)*?
U4	Just give me a comedy starring *Brad Pitt*, can you do that?

Figure 4.8: Excerpt showing a user’s failure to realize that a correct recommendation has been given. (The recommended movie *Snatch (2000)* is a comedy starring *Brad Pitt*, which is what the user asked for). U = User, S = System.

U1	Who is acting in the movie *Entrapment*?
S1	*Entrapment (1999)* has the following cast. [Acorn displays a list of actors: *Sean Connery, Catherine Zeta-Jones, ...*].
U2	Can you recommend a drama with *Sean Connery*?
S2	OK. Have you seen *Finding Forrester (2000)*?
U3	No
S3	*Finding Forrester (2000)* is a drama starring *Sean Connery.*
U4	Who else is starring in it?

Figure 4.9: Sample recommendation dialogue about movies. U = User, S = System.

only if the respondent has *not* seen the recommended title (compare to S4-U5-S5 in Figure 4.5). However, providing the explanation of the recommendation afterward seems not to be suitable for the dialogue in Figure 4.8. Here, it would perhaps be better if the explanation that *Snatch (2000)* is in fact a comedy starring *Brad Pitt* could be provided as part of utterance S3. A dialogue strategy could thus present recommendation explanations earlier, or better yet; provide *adaptive* recommendation explanations that depends on e.g. the number of previous successful transactions and recommendations.

4.5 Summary

This chapter presented the recommender system ACORN, which implements a dialogue strategy model based on the empirical investigations of human-human recommendation dialogues. The properties of ACORN's dialogue model was investigated in a study with end-users. The study verified that an efficient and effective dialogue management model with high usability measures can be achieved using the dialogue distillation of the human-human recommendation dialogue corpus. The study also results in implications for improvements of the model, including explicit support for conversational impetus for driving the dialogue forward and supporting domain exploration in the dialogue; and the importance of generating explanations for recommendations at the right time in the dialogue.

5

Storing and Utilizing Situation-Based User Preferences

This chapter describes a user preference modeling framework that supports and utilizes natural language dialogue, and allows for descriptive, comparative, and superlative preference statements, in various situations. The chapter also covers the PCQL *notation used for describing factual and preferential statements and requests.*

IN THE PREVIOUS chapter, it was shown that we could design and customize a domain-dependent conversational recommender system based on the investigation reported on in Chapter 3. In order to explore dialogue strategies for other domains and applications, we now aim for mechanisms to manage preferences in the dialogue in a more elaborate and generic way, as well as laying the ground work for easier modification of a conversational recommender system's dialogue behavior.

Before defining a recommendation dialogue strategy model based on the findings of the previous chapters, we will use this chapter to provide a user preference model framework that supports and utilizes natural language dialogue tailored for recommendation dialogue. In this matter, the work is based on—and extends—the preference-detection model of Carberry *et al.* [1999]. Two components are involved in this matter. Therefore, we first identify the desired characteristics and requirements of the preference model framework by relating it to the user modeling classifications reported on in Chapter 2. This is done in Section 5.1.

Second, a data manipulation notation called PCQL for representing factual and preferential statements and requests in a compact and unambiguous form is presented in Section 5.2. PCQL accommodates the preferential dialogue theory of Carberry *et al.* [1999], and the notation will also be used as the message-passing format in the recommendation dialogue strategy model in Chapter 6.

In Section 5.3, we outline how recommendation dialogue and a preference model could cross-fertilize each other. All these aspects are taken together in the specification of the *preflet* model, which is the topic of Section 5.4. PCQL and preflets form two important corner-stones of the BCORN model, which is the topic of the next chapter.

5.1 Defining the Preference Model Framework

The aim for this chapter is to design a user preference modeling framework suitable for conversational recommender systems. The characteristics of the user preference model is captured by placing them in the dimensions suggested by Kass and Finin [1988] (see Table 2.1, Section 2.2.1). Table 5.1 summarizes the requirements, dimension characteristics, and limitations of the user preference modeling framework about to be constructed in this chapter.

5.1.1 Requirements

The first overall requirement (R1) is that the model is directly compatible with natural language dialogue; i.e. it should both support dialogue modeling and management, and be correctly updated by the dialogue. Concretely, we build on the investigation by Carberry *et al.* [1999]. Thus, the user preference model should support preference strength variation of the utterance types *direct*, *indirect*, and *hedging* (R1a), as

Table 5.1: Characterization of capabilities, dimensions, and limitations for the proposed user preference model (compare Table 2.1). Q-A = Question-and-Answer, Vol = Volunteered, Vol-Bg = Volunteered-Background, Rej-Sol = Reject-Solution, Acc-Sol = Accept-Solution, CB = content-based, CF = collaborative filtering.

	Requirements	
Requirement		**Value(s)**
R1a	Utterance types	direct, indirect, hedging
R1b	Conversational circumstance	Q-A, Vol, Vol-Bg, Rej-Sol, Acc-Sol
R1c	Preferential statement types	descriptive, comparative, superlative
R2	Situations	multiple
	Dimensions	
Dimension		**Value**
D1	Specialization	individual
D2	Temporal extent	long-term
D3	Modifiability	dynamic
D4	Number of agents	one-to-one
D5	Number of models	multiple
D6	Method of use	descriptive
	Limitations	
Limitation		**Value**
L1	domain complexity	independent attributes
L2	attribute types	disjoint, scalar, complex
L3	recommender engine support	CB engine

well as support the conversational circumstances *question-and-answer*, *volunteered*, *volunteered-background*, and *reject-solution* (R1b). As discovered in the corpus analysis, there is also an *accept-solution* circumstance that has a preference strength comparable to *volunteered-background*. This circumstance is included as requirement R1b. As argued in Chapter 3, supporting human-like preferential statements requires not only descriptives, but also comparatives and superlatives. Our preference model should thus be able to handle relative preferences expressed by comparative statements, as well as superlatives, along with basic descriptives (R1c).

Preferences are often situation-dependent and it is therefore important that a recommender system can handle that. For example, even though a user may have a strong preference toward mellow jazz while casually driving her car, she may prefer upbeat dance music when working out. The second overall requirement (R2) is thus to provide a model that captures user preferences for multiple situations.

5.1.2 Dimensions

Following the agent model dimensions (D) found in Section 2.2.1, we pin point the proposed user preference model as follows: For the application type we have in mind it is desirable to capture (D1) *individual* and (D2) *long-term* characteristics of the user. Furthermore, user preferences are built up incrementally and evolves through new discoveries in the dialogue interaction depending on what, how, and when, a user elicits a preference; thus being an instance of a (D3) *dynamic* user model. We limit the model to model a one-to-one relationship between user and system, and therefore model one user in the system interaction (D4). As dictated by R2, we allow each user to have multiple models (D5), one for each situation. To keep the solution as simple as possible, we maintain a plain database of preference data entries, which makes the model *descriptive* (D6).

5.1.3 Limitations

In addition to the dimensions listed above, we make some limitations (L) regarding domain model complexity and back-end resource suitability:

In line with Carberry *et al.* [1999], we make an *independent attribute assumption* (L1). For instance; in the music domain, a preference for an entity value for the type

Artist (e.g. Metallica) does not have any explicit implication on the preference for the type genre values (e.g. Heavy Metal) of that artist. Furthermore, the model supports attributes that are *disjoint*, *scalar*, or *complex* (L2) (see Section 2.2.2).

The preference model suggested in this chapter is targeted for content-based (L3) recommender engines, since it deals with entity types as well as entity values (see Section 2.2.3).

5.2 PCQL

Having laid out the basic requirements based on previous user modeling research (see Chapter 2) and recommendation dialogue studies (see Chapters 3 and 4), we need a data manipulation language in order to computationally model recommendation dialogue. As we have seen in the preceding investigations, there is a need for both factual and preferential statements and requests in recommendation dialogue. A notation language aiming to cover recommendation dialogue in a dialogue strategy management framework should thus allow for a compact and efficient formulation of conventional, preferential and factual statements and requests. PCQL[1] is a formalism that consists of **action statements** that represent dialogue act specifications of recommendation dialogues [Wärnestål *et al.*, 2007c]. PCQL action statements are used for representation of user and system acts as well as for communication with external resources. The formalism is targeted for human-like preferential and factual expressions and intended to be used as a message passing format for the dialogue manager module in conversational recommender systems.

Since PCQL is a conversational formalism, the PCQL action statements have a double function. On the one hand, each statement *describes* some aspects of the factual and preferential state (the FP **state**) of the dialogue system. On the other hand, each PCQL action statement expresses an *action* performed by the dialogue participant, a dialogue act, where the acting agent is doing something that will result in a response from the dialogue partner. The description is subordinate to the dialogue action, but the latter requires the first to be fully understood. In that sense, the descriptive expression is a parameter to the stated action.

---

[1] Preferential Conversation Query Language.

5.2.1 FP State Formulas

The expressions of PCQL that are used to describe (aspects of) the FP state are called FP **state formulas**. In this section, we define the syntax of this formalism[2].

The FP state formulas express relations and entities of the domain that are in focus in the dialogue. The basic constituents of this language are constraints over **entity types** and **entity values**. The entity types are predefined types of possible entity values, such as Genre, which can be enumerations of known entities or open domains such as "any string". The entity values are either atomic domain entities—such as Electronic— or sets/intervals of entity values—such as $\{\mathsf{Rock}, \mathsf{Electronic}\}$ and [1989..1999]. The domains that we aim to cover are assumed to deal with attribute type sets that are finite.

The constraints can be formed using the factual operators shown in Table 5.2. A special entity type is YN consisting of the values Yes and No. References to entities through other entities (relations or attributes) are handled with two constructs. The first is to use the π operator to mark entity types whose values are inferred from the other constraints in a formula. For example, *"Albums of The Beatles and Deep Purple"* can be described as

$$\pi\,\mathsf{Album}, \mathsf{Artist} \in \{\mathsf{The\ Beatles,\ Deep\ Purple}\}$$

Informally, we may read this as follows: $\mathsf{Artist} \in \{\mathsf{The\ Beatles,\ Deep\ Purple}\}$ specifies a set of entities (in this case two); $\pi\,\mathsf{Album}$ projects this set of entities on the albums (in this case all albums by either *The Beatles* or *Deep Purple*). The second construct is that entity values can indirectly be referred to as attributes of other entity values using dot notation on the entity type names, for example 'My Song'.Album denotes the album of which the song 'My Song' belongs.

We form constraints from atomic entity types and entities, and by augmenting atomic constraints with factual operators (see Table 5.3 for examples). From the factual constraints, we form conjunctive formulas, called **factual** FP state formulas, where comma is used as conjunction sign. Intuitively, the meaning of the factual FP

---

[2] We use an abstract syntax notation for the FP state formulas. A concrete syntax suitable for implementation exists, but it is less readable in text. For example: $\boxplus$ corresponds to ++, and $\rhd$ corresponds to >>.

Table 5.2: Factual and preferential operators of the FP state formulas. The factual operators are used to form unary and binary constraint expressions. The preferential operators are used on the factual constraints to formulate: Descriptive, comparative, and superlative ratings' polarities are either positive or negative. Note that hedges ($\diamond$) can be combined with descriptive, superlative, and comparative preferential operators.

Factual			
Operator	**Name**	**Arity**	**Meaning**
$\circ$	Operator type	1 or 2	*don't know*
$\top/\bot$	max/min	1	*newest/oldest*
π	projection	1	*entity reference*
$=/\neq$	(not) equals	2	*is/is not*
$</>$	comparison	2	*newer/older*
$\in/\notin$	(not) member	2	*one of/not one of*
Preferential			
Operator	**Name**	**Arity**	**Meaning**
$\circledcirc$	Operator type	1 or 2	*don't know*
$\odot$	Indifferent	1 or 2	*doesn't matter*
$\oplus/\ominus$	Descriptive	1	*good/bad*
$\boxplus/\boxminus$	Superlative	1	*the best/the worst*
$\rhd/\lhd$	Comparative	2	*better/worse*
$\diamond$	Hedging	-	(marks uncertainty)

state formulas can be read as specifications of sets of entities. The unary operators are really aggregate operators on such sets, where the aggregate is given implicitly by the remaining formula[3].

Given the set of factual FP state formulas, we form atomic **preference formulas** by augmenting the preference operators shown in Table 5.2. It is not allowed to nest the preference operators in the same atomic preference formula (since this would increase the complexity of the language without being useful). From the factual FP state formulas and the atomic preference formulas, we form conjunctive formulas using comma as the conjunction sign. Furthermore, each preference operator may be indexed with a *hedging* symbol ($\diamond$), that indicates uncertainty about the preference [Carberry *et al.*, 1999]. The intuitive reading of the preference formulas are as statements of *like* and *dislike* of the sets of entities described by the factual part of the formula.

Finally, the factual and preference operator symbols form two **operator types**, denoted by $\circ$ and $\circledcirc$ respectively. The type symbols $\circ$ and $\circledcirc$ can be used in any formula in place of an operator to express uncertainty or requests concerning the operator's position. For example, the question *"Is Bob Dylan an artist or not?"* can be described using $\circ$, as the FP state formula

$$(\mathsf{Artist} \circ \mathsf{Bob\ Dylan})$$

Similarly, the preference statement *"Is Elvis Presley better or worse than Deep Purple?"* can be described using $\circledcirc$, as the FP formula

$$(\mathsf{Artist}=\mathsf{Elvis\ Presley}) \circledcirc (\mathsf{Artist}=\mathsf{Deep\ Purple})$$

This forms the complete FP state formula language, for which various examples can be found in Tables 5.4 and 5.5. The format of the FP state formulas is influenced by how formulas of modal (and intentional) logic keep a clean separation between the factual level and the belief of the agents, but at the same time allows for mixing levels freely in compound formulas.

An FP state formula describes a conjunctive aspect of the total FP state that is

---

[3] The Max/Min operators have higher priority than projection, in formulas where both occur.

Table 5.3: FP state formula mappings for factual utterance types. The table shows by prototypical examples how expressions of factual state in utterances correspond to FP state formulas.

Explicit	**Utterance Example**	**FP State Formula**
Entity Type	What is genre?	Genre
	One of genre, artist and album	$\mathsf{Genre}, \mathsf{Artist}, \mathsf{Album}$
Entity	Techno	$\mathsf{Genre} = \mathsf{Techno}$
Enumeration	Both Dylan and Waits	$\mathsf{Artist} \in \{\mathsf{Dylan},\ \mathsf{Waits}\}$
Yes/No	Yes	$\mathsf{YN} = \mathsf{Yes}$
Negation	Not Dylan	$\mathsf{Artist} \neq \mathsf{Dylan}$
Interval	Album three to five	$(\mathsf{AlbumNo} \in [3..5])$
Relative	Newer than 1975	$(\mathsf{Year} > 1975)$
Aggregate	The latest	$\top\ \mathsf{Year}$
Aggregate	Most sold album of the 1970's	$\top\ \mathsf{SoldCopies}, (\mathsf{Year} \in [1970..1979])$
Referential		
Entity	An album by Dylan	$\pi\ \mathsf{Album}, \mathsf{Artist} = \mathsf{Dylan}$
Enumeration	Albums by either Dylan or Waits	$\pi\ \mathsf{Album}, \mathsf{Artist} \in \{\mathsf{Dylan}, \mathsf{Waits}\}$
Negation	All albums except The Beatles'	$\pi\ \mathsf{Album}, \mathsf{Artist} \neq \mathsf{The\ Beatles}$
Interval	Songs from the 70s	$\pi\ \mathsf{Song}, \mathsf{Year} \in [1970..1979]$
Relative	Albums older than Weathered	$\pi\ \mathsf{Album}, (\mathsf{Year} < \mathsf{Weathered.Year})$
Aggregate	The first of Dylan's albums	$\pi\ \mathsf{Album}, \perp\ \mathsf{Year}, (\mathsf{Artist} = \mathsf{Dylan})$

relevant for a particular dialogue act. We say that each FP state formula expresses an FP **state mapping** from the dialogue act to some entities of the FP state that are in focus.

5.2.2 Factual State Mapping

The *factual* FP state formulas deal with information-providing aspects of the system state. We distinguish between factual FP state formulas that concern explicitly stated entities and those that are indirectly referenced using the projection operation (π).

Table 5.3 shows the identified classes of factual descriptions in dialogue acts we have found from the examined material, as discussed in Chapter 3.

In the explicit factual FP state formulas, entities are referred to by their name (in the system). In explicit aggregates and relative statements, it is the aggregate or relative value that is explicit. For example, in *"most popular in the 70s"* the aggregate set *"the 70s"* is explicitly mentioned.

In the referential factual FP state formulas, entities are referred indirectly through properties or relations that specify them. This means that the formula must also specify of what type the referred entity is. Referential formulas are most obviously occurring in questions, but may also occur in informative statements. In particular, they may be part of the informative part of user preference utterances.

5.2.3 Preference State Mapping

Preferential user utterances are built "around" factual FP state formulas, using the preference operators. Descriptive and superlative statements are syntactically handled in the same way in FP state formula mapping schemes, as shown in Table 5.4. Both types of constructs amount to similar 1-arity formulas. However, observe that the meaning of superlatives is a form of aggregate functions operating on sets, which is more complex than the descriptive case. Since these aggregates are given implicitly by the context, this complexity is hidden from the formula. For example, the FP state mapping of the sentence *"The Beatles is the best Pop artist"* can be described by the FP state formula

$$\boxplus(\mathsf{Artist}=\mathsf{The\ Beatles}),(\mathsf{Genre}=\mathsf{Pop})$$

Most factual constructs make sense as part of a preference statement. The constructs that make little sense are: explicit and referential negation, and Yes/No. In real dialogue, some of the listed utterances are less important than others. However, recall that we want to be able to use PCQL *after* contextual interpretation. In some cases this means that the FP state formula at hand actually contains the collected information of a whole sub-dialogue. In a collected formula, more complicated constructs may have been gathered over time. Thus, PCQL covers both the collected formulas and the simpler ones in a natural way.

Compound FP is a type of formula that occur only on the preference level. This class contains utterances that separately combines one part that is expressing a preference with one part that is factual (see Table 5.4).

Comparative utterances are 2-arity constructs, and are handled differently than the 1-arity preference formulas. Table 5.5 shows how the factual classes are handled

Table 5.4: FP state formula mappings for descriptive and superlative preference utterances.

1-Arity	**Utterance**	**FP State Formula**
Explicit Entity Type	Genre and artist are important, but not album	$\oplus(\mathsf{Artist}, \mathsf{Genre}), \ominus\ \mathsf{Album}$
	The artist does not matter	$\odot\ \mathsf{Artist}$
	Artist is most important	$\boxplus\ \mathsf{Artist}$
Explicit Entity	I like The Beatles	$\oplus(\mathsf{Artist} = \mathsf{The\ Beatles})$
	Techno is not good	$\ominus(\mathsf{Genre} = \mathsf{Techno})$
	Dylan is the best artist	$\boxplus\ (\mathsf{Artist} = \mathsf{Dylan})$
Explicit Enumeration	I like Dylan, The Beatles and Deep Purple	$\oplus(\mathsf{Artist} \in \{\mathsf{Dylan,\ The\ Beatles,\ Deep\ Purple}\})$
	I like Dylan and Waits the best	$\boxplus(\mathsf{Artist} \in \{\mathsf{Dylan,\ Waits}\})$
Explicit Interval	I like Album three to five	$\oplus\ \mathsf{AlbumNo} \in [3..5]$
	I like Album three to five the best	$\boxplus\ \mathsf{AlbumNo} \in [3..5]$
Explicit Relative	I might like everything older than 1975	$\oplus_{\diamond}(\mathsf{Year} < 1975)$
	I like everything older than 1975 the best	$\boxplus(\mathsf{Year} < 1975)$
Explicit Aggregate	I like the most sold albums from the 70's	$\oplus(\top\ \mathsf{SoldCopies}, (\mathsf{Year} \in [1970..1979]))$
Referential Entity	I like all of Dylan's albums	$\oplus(\pi\ \mathsf{Album}, (\mathsf{Artist} = \mathsf{Dylan}))$
Referential Enumeration	I like songs with Creed and Bush	$\oplus(\pi\ \mathsf{Song}, (\mathsf{Artist} \in \{\mathsf{Creed,\ Bush}\}))$
	I don't like songs with Creed and Bush	$\ominus(\pi\ \mathsf{Song}, (\mathsf{Artist} \in \{\mathsf{Creed,\ Bush}\}))$
Referential Interval	I like songs from the 60's	$\oplus(\pi\ \mathsf{Song}, (\mathsf{Year} \in [1960..1969]))$
	I like songs from the 60's best	$\boxplus(\pi\ \mathsf{Song}, (\mathsf{Year} \in [1960..1969]))$
Referential Relative	I like all Moby's albums before Play	$\oplus(\pi\ \mathsf{Album}, (\mathsf{Artist} = \mathsf{Moby}), (\mathsf{Year} < \mathsf{Play.Year}))$
Referential Aggregate	I like Dylan's latest album	$\oplus(\pi\ \mathsf{Album}, \perp \mathsf{Year}, (\mathsf{Artist} = \mathsf{Dylan}))$
	Dylan's latest album is the worst	$\boxminus(\pi\ \mathsf{Album}, \perp \mathsf{Year}, (\mathsf{Artist} = \mathsf{Dylan}))$
Compound FP	I like Elvis when I am working	$\oplus(\mathsf{Artist} = \mathsf{Elvis}), (\mathsf{Situation} = \mathsf{Work})$
	Elvis is the best when I am working	$\boxplus(\mathsf{Artist} = \mathsf{Elvis}), (\mathsf{Situation} = \mathsf{Work})$

Table 5.5: FP state formula mappings for 2-arity comparatives.

2-Arity	**Utterance**	**FP State Formula**
Explicit Entity Type	Artist is more important than Album	$\mathsf{Artist} \rhd \mathsf{Album}$
Explicit Entity	Master of Puppets is better than Ride the Lightning I prefer techno to songs by Creed	$(\mathsf{Album} = \text{‘Master of Puppets’}) \rhd (\mathsf{Album} = \text{‘Ride the Lightning’})$ $(\mathsf{Genre} = \mathsf{Techno}) \rhd (\mathsf{Artist} = \mathsf{Creed})$
Explicit Enumeration	I like Dylan and Waits better than The Beatles	$(\mathsf{Artist} \in \{\mathsf{Dylan},\ \mathsf{Waits}\}) \rhd (\mathsf{Artist} = \text{‘The Beatles’})$
Explicit Interval	I like Album three to five better than the others	$\mathsf{AlbumNo} \in [3..5] \rhd \mathsf{AlbumNo} \notin [3..5]$
Explicit Relative	I prefer newer than 1975 over older	$(\mathsf{Year} > 1974) \rhd (\mathsf{Year} < 1975)$
Explicit Aggregate	I like the most sold from the 70’s better than rock	$(\top\ \mathsf{SoldCopies}, (\mathsf{Year} \in [1970..1979])) \rhd (\mathsf{Genre} = \mathsf{Rock})$
Referential Entity	I like Dylan’s genre better than Scooter’s	$(\pi\ \mathsf{Genre}, (\mathsf{Artist} = \mathsf{Dylan})) \rhd (\pi\ \mathsf{Genre}, (\mathsf{Artist} = \mathsf{Scooter}))$
Referential Interval	I like songs from the 90’s better than classical music	$(\pi\ \mathsf{Song}, (\mathsf{Year} \in [1990..1999])) \rhd (\pi\ \mathsf{Song}, (\mathsf{Genre} = \mathsf{Classical}))$
Referential Enumeration	I like albums by Dylan or Waits better than Bush	$(\pi\ \mathsf{Album}, (\mathsf{Artist} \in \{\mathsf{Dylan},\ \mathsf{Waits}\})) \rhd (\pi\ \mathsf{Album}, (\mathsf{Artist} = \mathsf{Bush}))$
Referential Relative	I like all Moby’s albums before Play better than Dylan	$(\pi\ \mathsf{Album}, (\mathsf{Artist} = \mathsf{Moby}), (\mathsf{Year} < \mathsf{Play.Year})) \rhd (\mathsf{Artist} = \mathsf{Dylan})$
Referential Aggregate	I like Dylan’s latest album better than Creed	$(\pi\ \mathsf{Album}, \perp \mathsf{Year}, (\mathsf{Artist} = \mathsf{Dylan})) \rhd (\mathsf{Artist} = \mathsf{Creed})$
Compound FP	I like Bush better than Moby when I am working	$(\mathsf{Artist} = \mathsf{Bush}) \rhd (\mathsf{Artist} = \mathsf{Moby}), (\mathsf{Situation} = \mathsf{Work})$

by FP state formulas in comparative preference contexts using infix notation.

5.2.4 PCQL Action Statements

When we use PCQL to model natural language utterances we attach **action tags** to FP state formulas. An action tag is a domain or applications-specific category that accepts specific FP state formulas as valid arguments. PCQL does not make any assumptions on action tag categories, and it is up to the designer to select a set of tags that suits her needs [Pohl *et al.*, 1995].

An action tag can be used to assert facts, give answers, preferences and/or values. Such a tag accepts one or two arguments, where the (optional) second argument is a collection of values (e.g. a database result set).

The operator $\circledcirc$ can used to request type of preference. For factual requests (such as asking questions about domain items), projection π and aggregates are normally used. However, any formula can be seen as an implicit question, which may warrant the addition of projections to all kinds of formulas. For example, the FP state formula

$$\oplus\,(\pi\,\mathsf{Album}, (\mathsf{Artist}=\mathsf{Bob\ Dylan}))$$

can be seen as the WH-question *"Which albums by Bob Dylan do you like?"*, or as the implicit yes/no question *"Do you like the albums of Bob Dylan?"*, which can be made explicit by adding of $\pi\,\mathsf{YN}$. In situations where the type of statement (e.g. WH- or yes/no-question) is important, action tags can be used. Similarly, action tags can express conventional actions. These statements usually accept one (possibly empty) argument. For example, an action tag GREET could be implemented as an empty-argument action to represent the utterance *"Hello"*, but it could also accept an FP state argument such as: GREET$\llbracket\ (\mathsf{Name}=\mathsf{Tom})\ \rrbracket$ to represent *"Hello Tom"*.

Each dialogue act may correspond to a PCQL action tag. The complete PCQL action statement (action tag and FP state formula) expresses the PCQL **action mapping** that specifies the dialogue act performed by the agent. Table 5.6 shows some of the possible mappings for the identified dialogue act types discussed in Section 3.3.3 used in the CoreSong system. We will return to the action tag set and its relation to the dialogue acts in Chapter 6. In these examples the focus is on the structure of the dialogue act and action tag. Therefore, only simple FP state descriptions are used, but any of the previously discussed mappings can be used here as well.

5.3 Utilizing and Supporting Dialogue

We look at the relationship between recommendation dialogue and preferences from two perspectives: First, we describe ways in which the recommendation dialogue is utilized in order to detect and calculate preferences. Second, we describe how the preference model can serve the recommendation dialogue. This forms the ground-

Table 5.6: A sub-set of PCQL action mappings in CoreSong for dialogue acts in the recommendation dialogue.

Act	**Utterance**	**PCQL Action Statement**
Factual Question	What electronic albums are there	$\textsc{ask} \llbracket \pi\ \mathsf{Album}, (\mathsf{Genre} = \mathsf{Electronic}) \rrbracket$
Preference Question	Is Moby better or worse than Creed?	$\textsc{ask} \llbracket (\mathsf{Artist} = \mathsf{Moby}) \circledcirc (\mathsf{Artist} = \mathsf{Creed}) \rrbracket$
	Which artists are better than Metallica?	$\textsc{ask} \llbracket (\pi\ \mathsf{Artist}) \rhd (\mathsf{Artist} = \mathsf{Metallica}) \rrbracket$
	What do you think about techno?	$\textsc{ask} \llbracket \circledcirc\ \mathsf{Genre} = \mathsf{Techno} \rrbracket$
	Which song do you like best on album Weathered?	$\textsc{ask} \llbracket \oplus (\pi\ \mathsf{Song}, (\mathsf{Album} = \mathsf{Weathered})) \rrbracket$
	Which genres or artists do you prefer?	$\textsc{ask} \llbracket \oplus (\mathsf{Value} \in \{\mathsf{Genre},\ \mathsf{Artist}\}) \rrbracket$
Factual Statement	These artists belong to the genre rock: `[X,Y,Z,...]`	$\textsc{inform} \llbracket \pi\ \mathsf{Artist}, (\mathsf{Genre} = \mathsf{Rock}) \rrbracket \text{VALUES} \llbracket \mathsf{Artist} : \{\mathsf{X},\ \mathsf{Y},\ \mathsf{Z}, \ldots\} \rrbracket$
Preference Statement	I like techno but I don't like Moby	$\textsc{inform} \llbracket \oplus (\mathsf{Genre} = \mathsf{Techno}), \ominus (\mathsf{Artist} = \mathsf{Moby}) \rrbracket$
	I like Creed when I work	$\textsc{inform} \llbracket \oplus (\mathsf{Artist} = \mathsf{Creed}), (\mathsf{Situation} = \mathsf{Work}) \rrbracket$
Recommendation	Have you heard the song Just One?	$\textsc{ask} \llbracket \pi\ \mathsf{YN}, (\mathsf{Song} = \text{‘Just One’}) \rrbracket$
Agreement (Reject)	No, I don't like Hoobastank	$\textsc{inform} \llbracket \mathsf{YN} = \mathsf{No}, \ominus (\mathsf{Artist} = \mathsf{Hoobastank}) \rrbracket$
Greet	Hello	$\textsc{greet} \llbracket\ \rrbracket$
Bye	Good bye	$\textsc{bye} \llbracket\ \rrbracket$

Table 5.7: Preference utterance types and their connection to dialogue acts. Examples from the music domain.

Type	Act	Example	PCQL FP state
direct	PREFERENCE-STMT	*I like jazz*	$\oplus\,(\mathsf{Genre} = \mathsf{Jazz})$
indirect	FACTUAL-QUESTION	*What jazz artists are there?*	$\pi\,\mathsf{Artist}, (\mathsf{Genre} = \mathsf{Jazz})$
hedging	PREFERENCE-STMT	*I think I might like jazz*	$\oplus_{\diamond}\,(\mathsf{Genre} = \mathsf{Jazz})$

work for the user preference model construct called a *preflet* put forward in detail in Section 5.4.

5.3.1 Utilizing Dialogue

This section connects the empirically discovered dialogue acts presented in Section 3.3.3, with the theory of Carberry *et al.* [1999] (see also Section 2.2.2).

According to the theory, a preference utterance type can be classified as direct, indirect, or a hedging. Table 5.7 shows correspondence between utterance types as defined by Carberry *et al.*, and the dialogue acts in the corpus analysis. Classification of utterances into types and dialogue acts are performed by examining the PCQL statements. The occurrence of a preference operator in a PCQL statement signals that the dialogue act is a PREFERENCE-STATEMENT. A preference hedging operator ($\diamond$) further classify the preference statement as a hedging. A projection operator (π), and the absence of any preference operators, indicates a FACTUAL-QUESTION. Preference and projection operators are explained in Section 5.2.1.

Second, we can modify the preference strength by moving up from utterance level and examine the *conversational circumstance* in which the utterance occurs (Question-and-Answer, Volunteered, Volunteered-Background, Reject-Solution, and Accept-Solution). Table 5.8 shows (sequences) of dialogue acts corresponding to conversational circumstances that have impact on preference detection. A prerequisite for the Question-and-Answer circumstance classification is that the topic of the PREFERENCE-QUESTION from the system and the PREFERENCE-STATEMENT from the user are the same. Otherwise the user's PREFERENCE-STATEMENT will be classified as VOLUNTEERED. Agreement acts such as REJECT and ACCEPT are con-

Table 5.8: Conversational circumstances and their connection to dialogue acts. Examples from the music domain. Q-A = Question-and-Answer, Vol = Volunteered, Bg = Background, Rej-Sol = Reject-Solution, Acc-Sol = Accept-Solution, S = system, U = user, STMT = Statement.

Circ.	**Act(s)**	**Example**
Q-A	S: PREFERENCE-QUESTION	*What genre do you like?*
	U: PREFERENCE-STMT	*I like jazz*
Vol	U: PREFERENCE-STMT	*I like Eric Clapton*
Vol-Bg	U: FACTUAL-STMT	*I want to setup a play list for work*
	U: PREFERENCE-STMT or	*I like techno*
	FACTUAL-QUESTION	*What techno artists are there?*
Rej-Sol	S: RECOMMENDATION	*Have you heard* Waterloo *by* ABBA*?*
	U: REJECT	*No. I don't like ABBA*
Acc-Sol	S: RECOMMENDATION	*Have you heard* Waterloo *by* ABBA*?*
	U: ACCEPT	*Yes. I like ABBA*

sidered as described in Section 3.3.3. Note that it is also possible to use a compound FACTUAL-STATEMENT/PREFERENCE-STATEMENT dialogue act in the case of Volunteered-Background, such as *I like Elvis Presley when I work.* This is managed with PCQL *compound statements.* The example above would have the following PCQL FP state formula:

$$\oplus\,(\mathsf{Artist} = \text{`}\mathsf{Elvis\ Presley}\text{'}),\ (\mathsf{Situation} = \mathsf{Work})$$

Preference strength and reliability are calculated according to Carberry *et al.* [1999] (see Section 2.2.2). Utterance type and circumstance are combined into a preference strength measure in the ordinal range *weak-2, weak-1, mod-2, mod-1, str-2, str-1*, translated to an integer preference strength interval $[1, 6]$ (or $[-1, -6]$ for negative preferences). As noted in Section 2.2.2, reliability is modeled by the use of endorsements, which are viewed as explicit factors that affect the modeling agent's certainty in a preference strength hypothesis. They are translated to an integer interval $[1, 5]$.

Based on the findings in the corpus study, we provide the following extensions of the original theory:

- *Superlative preference statements* are interpreted as the strongest kind of preference, with preference strength 6 or -6, and reliability measures depending on conversational circumstance (see Section 2.2.2).

- Strengths arising from *comparative preference statements* are resolved as detailed in Section 5.4.6.

- The conversational circumstance ACCEPT-SOLUTION is assigned with preference strength and reliability as outlined in Section 3.3.3.

Utterance type and circumstance identification patterns (as exemplified in Tables 5.7 and 5.8) can obviously be made more elaborate if needed. However, the examined material indicates that the patterns presented here are sufficient for the kinds of systems dealt with in this thesis.

5.3.2 Supporting Dialogue

The previous section described how the dialogue should utilize the preference modeling framework in order to detect and record preferences. In this section we take the opposite perspective, and examine how the preference structure can support a recommendation dialogue flow.

Constraining and Relaxing Values in Interviews

Generally, constraints are needed when the information space is too big. On the other hand, constraints need to be relaxed when there are no items in the space that match. A collaborative dialogue partner will inform her inquirer of the status, and suggest suitable attributes that should be modified (i.e. relaxed or constrained). Two principal cases of constraining and relaxation of entity types are considered, based on the interview dialogue behavior described in Chapter 3 (Section 3.3.4).

First, factual queries may be incomplete (such as underspecified, ambiguous, or outright erroneous) or over-specified, and thus need to be modified with additional specification before a database query can be carried out. This relates to inquiry-oriented dialogue and may consist of constraining or relaxation. Second, preferences may need to be relaxed or constrained in a similar manner. The dialogue excerpts shown in Figures 3.10 and 3.11 exemplify constraining and relaxation of preferences in dialogue. The two cases are similar. Indeed, we have already established that factual queries are considered indirect preferences. One might argue however, that the intention of the participant requesting explicit information from e.g. a database is different from the participant who elicits preferences in order to get recommendations.

There are several ways to handle the dynamics of suggesting attribute relaxation and constraining, depending on how complex the domain is, and how advanced ("intelligent") the attribute selection should be.

A straight-forward and simple way to handle the problem is to pre-define a ranked list of the available domain attributes. For example, in the movie domain, one strategy is to start with generics and go toward specifics for constraining, and reverse the order for relaxation. The "initialization" step of the movie recommendation dialogue strategy model in Figure 4.1 suggests the following constrain order for the movie domain: Genre, Actor, and Director.

More elaborate techniques are sometimes required. In particular, there should be some guarantee that e.g. a relaxation of a suggested attribute leads to a possible solution to the original inquiry. Therefore, depending on processing capability and domain description, the ranked-list approach could be augmented with a check that the suggested modification really results in a query that delivers a database result set within a desired range (e.g. larger than 0, but smaller than some pre-defined limit).

Another way to augment the selection process is to let the relaxation and constraining be guided by the content of the user's preference model. Here, the system would retain strong preferences with high reliability as long as possible and suggest modification of weaker preferences or preferences with lower reliability first. The preference modeling framework suggested below supports both the generic-to-specific approach, as well as a preference dependent approach (see Section 5.4.4).

Motivations in Indirect Deliveries

When issuing indirect deliveries, motivations are central for building trust [Swearingen and Sinha, 2002] between user and system, and help explaining the inner workings of the recommender system [Höök, 2000]. Our preference modeling framework supports this by using reliability measures and preference strengths to determine what attributes should be included when generating motivations. Note that this is a domain-dependent issue, since the chosen recommender engine type ultimately will dictate what attributes and preferences are available. For content-based engines, which naturally require a domain model in the form of attribute-value descriptions of the included items, users' preferences can be mapped directly onto these features. A recommended item's attributes are matched to the user's preferences for the current situation. Only preferences with reliability above a specified threshold (see Section 5.4.3 below) are considered, and are ranked based on the preference strength. Accordingly, motivations consist of relating features of an item to the strongest preferences with a high reliability in the user's preference model. This can be expressed in generic PCQL as follows for an attribute-value description of a domain item with a ranked list of entities derived from the preference model.

$$\text{MOTIVATE} \llbracket \left(\mathsf{Item} = \mathsf{ItemName}\right), \{e_{type_1}, e_{type_2}, \ldots, e_{type_n}\} \rrbracket$$

When generating a natural language statement from such a PCQL statement instance, the choice could be to design templates that take into account the number of allowed items on the ranked list [Buczak *et al.*, 2002]. In Chapter 6 (Section 6.4.2) examples of such a template-based approach are described for a music recommender system. Other ways to generate motivations for other types of recommender engines (e.g. CF engines) have been suggested by e.g. Herlocker *et al.* [2000]. Examples of how the preference model can assist motivation generation and attribute constraining and relaxation are found in Chapter 6.

5.4 The Preflet Construct

Equipped with the specification described in the previous sections, we are ready to implement the **preflet** user preference model.

5.4.1 Definitions

A basic assumption for our framework to be useful is that **domain objects** are described by a set of **entity types** and **entity values**, (i.e name-value pairs). For example, an object in the music domain can be described by the set of attributes {Genre=Alternative, Artist=Audioslave, Album=Revelations, Title='Wide Awake'}.

Following Carberry *et al.* [1999], **preference strength** is expressed on an integer interval $[1,6]$, or $[-1,-6]$ for negative preferences. **Reliability** is expressed on an integer interval $[1,5]$ and corresponds to the ordinal scale presented in Table 2.2. Entity types are indexed with an **importance function value**, which is a positive real number. The default value of 1.

We define a **sentinel** to be of one of three types:

- maximum: $(\top\, pstr(x) : s)$
- minimum: $(\bot\, pstr(x) : s)$
- modification: $(\ddagger\, pstr(x) : s)$

A **maximum sentinel** sets an upper-limit preference strength s for x. Usually s is a dynamic reference to the strength of another preference, but there may be cases where a static strength defines the sentinel. A **minimum sentinel** similarly sets a lower-limit preference strength. A **modification sentinel** guarantees that the preference strength of its subject increases or decreases s steps, unless the ceiling or floor of the strength interval is reached.

Given the definitions above, a complete **preference** can be formulated as:

$$e_f \longrightarrow v_{(pstr,r,<S>^?)}$$

pstr is the preference strength in an integer interval $[-6,6]$ of the entity type e with an importance function value f, and value v with a reliability measure[4] r in the five-graded ordinal interval described in Section 2.2.2, and S is an optional sentinel expression. Sentinels are used to model comparative preference statements, and are explained in Section 5.4.6.

We further define a **preflet descriptor** to consist of a mapping from a descriptor name to a value. A preflet descriptor name and value pair is unique and can therefore be used for identification.

Finally, a **preflet** is a structure that consists of (a) a collection of preferences grouped on entity types, and (b) a set of preflet descriptors.

---

[4] Unless the reliability measure is explicitly needed for the discussion at hand, we omit it for clarity in the following.

5.4.2 Preflet Descriptors

In general, the preflet uses a **finite set** of preflet descriptors, with at most one preflet descriptor for each name. Intuitively, each such set combination of preflet descriptors defines a **preflet type**. For each preflet type, the user builds up user-specific attribute preferences. Each preflet descriptor can be thought of as a dimension of that type.

For example, let us assume that preflets in a certain domain are one-dimensional and describe different situations (we simply call the descriptor name - Situation). The descriptor set is a set of four pre-defined situations (e.g. Work, Exercise, Relax, and Party). This means that each user has at most four preflets to flesh out in our implementation of this domain.

In another example the preflet descriptor set can consist of two descriptors and is thus two-dimensional. The two names are Situation and Mood. A value is here any token assigned by the user. For example, the value of Situation can be defined by the user (such as Exercise) and likewise, the value of Mood (e.g. Happy, Mellow, etc.). The set of valid preflet types in this example is infinite since the user can create as many situations-mood pairs as she wants with one preflet for each pair (e.g. Happy-Exercise, Mellow-Driving, etc.).

In theory a preflet can have any number of preflet descriptors, and each value domain may be closed with pre-defined values (as in the first example) or open for arbitrary tokens (as in the second example). Typically, the number of preflet descriptor names is fixed for a given system, but preflet instances may not use all of them, and the number of preflets connected to a user increases gradually. For example, only Situation but not Mood may have been specified for a preflet at a certain point in a dialogue from the second example domain.

We suppose that exactly one preflet is active, and under discussion, at any given time in the dialogue. The user needs to explicitly switch and/or update preflets by referring to descriptors. That is, Preflet $p1$ (with the preflet descriptor Situation = u) can be updated in the context of Preflet $p2$ (with a preflet descriptor Situation = v) explicitly, such as in the following user utterance example (as response to a recommendation of the song Song x ($p2$ is active): *"No thanks, but Song x is good in Situation u"*. This updates $p1$ with a volunteered (positive) preference for Song x, and $p2$ with a reject-solution (negative) preference for Song x. However, the following dialogue will still concern $p2$ until the user explicitly activate $p1$ by saying e.g. *"Let's talk about Situation v"*.

5.4.3 Preference Evaluation

A user's set of preflets is used for the preference evaluation of domain objects. We assume that in any given moment of the dialogue only one preflet is active, selected based on the preflet descriptors (which in turn could reflect e.g. the situation currently under discussion, or the user's mood at the moment, depending on how the preflet descriptors for the system are designed). Therefore, the preference evaluation is always performed for a domain item (e.g. movie or song title) w.r.t. *one* particular preflet—or more precisely w.r.t. the preference set of that preflet.

The recommender engine utilizes the preference model when calculating prediction scores for song entities in the database. Each song is an entity (of entity type Song) that is described by a set of attributes from the entity types Genre, Artist, Album, and Title, with associated values. When calculating the predicted evaluation score for a specific song, the engine uses the preference strengths of matching genre, artist, and album values; and the importance for each entity type[5].

More formally, consider a set $S = \mathcal{T}_0, \ldots, \mathcal{T}_n$ of entity types that is a partition of a finite entity set $\mathcal{E}$. For each entity e in $\mathcal{E}$, by the *set of attributes* $A(e)$ we understand the set of pairs of the form $\langle e_i, \mathcal{T}_i \rangle$ such that e_i is an entity of type $\mathcal{T}_i$. For each such pair $\langle e_i, \mathcal{T}_i \rangle \in A(e)$, e_i is called an *attribute* of e. Similarly, let $A(\mathcal{T})$ denote the set of all types $\mathcal{T}_i$ of the attributes of the entities of $\mathcal{T}$.

Let $I_{\mathcal{T}}$ denote an *importance function* from T to the set of positive real numbers. Let $S_{\mathcal{T}}$ denote a *preference strength function* that is a mapping from $\mathcal{T}$ to the interval of integers $[-6, 6]$. Moreover, let $C_{\mathcal{T}}$ stand for a *closeness-of-match function* that is a mapping from T to the interval of integers $[0, 3]$[6].

Let $\mathcal{T}$ be an entity type in $\mathcal{E}$ where $I_{\mathcal{T}_i}, S_{\mathcal{T}_i}, C_{\mathcal{T}_i}$ exist for each type T_i occurring in

---

[5] The reliability measure is used as a threshold. Only preferences with a reliability higher than 2 are considered. However, low reliability preferences are still useful (and therefore retained), since several indications with low reliability may be combined to preferences with high reliability [Carberry *et al.*, 1999].

[6] The summing of products of preference strength and closeness-of-match follows the model of Carberry *et al.*, which in turn corresponds to a weighted additive model in human decision-making [Carberry *et al.*, 1999].

U1 What genres are there?
S2a I found these genres in the database: `[list]`.
S2b I'd like to know more about your preferences.
S2c What else do you like?
U2 I like the genre *Rock & Roll*
S3a These are the artists that belong to the genre *Rock & Roll*: `[list]`.
S3b I'd like to know more about your preferences.
S3c What else do you like?
U3 What songs were made by *Elvis Presley*?
S4a I found these songs by the artist *Elvis Presley*: `[list]`.
S4b I'd like to know more about your preferences.
S4c What else do you like?
U4 I like the album *Live at Madison Square Garden*
S5a You might like the song *Suspicious Minds* because it is a *Rock & Roll* song by *Elvis Presley.*
S5b Have you heard it?

Figure 5.1: Sample recommendation dialogue in the music domain. `[list]` denotes lists of genres, artists, albums or songs. S = system, U = user.

$A(T)$. The *evaluation score function* $\sigma_{\mathcal{T}}$ for $\mathcal{T}$ w.r.t. $I_{\mathcal{T}_i}, S_{\mathcal{T}_i}, C_{\mathcal{T}_i}$ is then:

$$\sigma_{\mathcal{T}}(e) \ = \ \sum_{\langle e_i,\mathcal{T}_i\rangle \in A(e)} I_{\mathcal{T}_i}(e_i) \times S_{\mathcal{T}_i}(e_i) \times C_{\mathcal{T}_i}(e_i)$$

A *normalized* evaluation score mapped to a real number interval [-1,1], can be calculated by the normalization formula $F(x)/\max(\mathrm{range}(F))$.

Example 1 For instance, the following preference model is constructed from the dialogue excerpt in Figure 5.1:

$$\begin{aligned} &\mathsf{Genre}_{(2.0)} \ \longrightarrow \ \mathsf{Rock\&Roll}_{(4,4)} \\ &\mathsf{Artist}_{(1.0)} \ \longrightarrow \ \text{‘}\mathsf{Elvis\ Presley}\text{’}_{(3,3)} \\ &\mathsf{Album}_{(1.0)} \ \longrightarrow \ \text{‘}\mathsf{Live\ at\ Madison...}\text{’}_{(4,4)} \end{aligned}$$

In this example, the importance function value for the type *genre* was increased due to the indirect user interest detected in utterance U1 in Figure 5.1. In our example, all direct preferences are positive, and occur in a *question-and-answer* circumstance.

According to the model they get a preference strength and reliability of 4 [Carberry *et al.*, 1999]. The request for Elvis Presley's songs is modeled as an indirect preference, with strength and reliability 3.

Following our example dialogue, the evaluation score for the song *'Suspicious Minds'* turns out to be $(2.0 \times 4 \times 3) + (1.0 \times 3 \times 3) + (1.0 \times 4 \times 3) = 45$, resulting in a normalized score of 0.63. Since this is higher than the recommender engine's threshold (0.50), the song is recommended in utterance S5b in Figure 5.1.

Example 2 The following example describes how the preference strength evaluation function of a sample preflet is used to calculate the preference score for two objects in the music domain. First, we consider the two domain objects D1 and D2:

$$D1 : \{\,\textsf{Genre=Alternative, Artist=Audioslave, Album=Revelations, Title='Wide Awake'}\}$$

$$D2 : \{\,\textsf{Genre=Alternative, Artist='Snow Patrol', Album='Eyes Open', Title='Chasing Cars'}\}$$

Then, consider the following (partial) preflet for a particular user:

$$\begin{aligned} &\textsf{Genre}_{(1.0)} &&\rightarrow \textsf{Alternative}_{(4,4)} \\ &\textsf{Artist}_{(1.0)} &&\rightarrow \textsf{Audioslave}_{(5,3)}, \textsf{'Snow Patrol'}_{(3,2)} \\ &\textsf{Album}_{(2.0)} &&\rightarrow \textsf{Revelations}_{(3,3)}, \textsf{'Eyes Open'}_{(2,4)} \end{aligned}$$

As recommended by Carberry *et al.*, a reliability threshold can be used so that only preferences with a reliability rating of 3 (*Moderate*) or higher are considered. The artist preference for *Snow Patrol* has a reliability rating lower than the threshold. Therefore, it is not considered at this time. The preference itself is retained of course, since future interactions may accumulate evidence that increases the reliability rating to a useful level.

The maximum evaluation score for a domain item (Song) for the example preflet is 72[7]. The normalized preference evaluation score for D1 is 45/72, and 24/72 for D2. Table 5.9 shows the preference evaluation score calculations for the example preflet.

---

[7] There are three entity types in this domain, the maximum strength is 6, the maximum closeness of match is 3, and the entity type Album has importance function value 2.0—the others have 1.0.

Table 5.9: Evaluation scores for example domain objects D1 and D2, given the partial preflet. Imp-F = Importance Function, Str = Strength, C-o-M = Closeness of Match, Rel. = Reliability.

D1 (“Wide Awake” by Audioslave)					
	Imp-F	**Str**	**C-o-M**	**Product**	**Rel.**
Genre	1	4	3	12	4
Artist	1	5	3	15	3
Album	2	3	3	18	3
Evaluation score				45	
Normalized score				0.63	
D2 (“Chasing Cars” by Snow Patrol)					
	Imp-F	**Str**	**C-o-M**	**Product**	**Rel.**
Genre	1	4	3	12	4
Artist	1	3	3	0	2
Album	2	2	3	12	4
Evaluation score				24	
Normalized score				0.33	

Example 3 Our third example shows the a score evaluation for both negative and positive preferences. First, consider an addition to the user's preflet from Example 2.

$$\mathsf{Artist}_{(1.0)} \longrightarrow \mathsf{Creed}_{(-5,3)}$$

Then, consider the domain object D3.

$$D3 : \{\, \textsf{Genre=Alternative, Artist=Creed, Album=Weathered, Title='Hide'} \}$$

D3's evaluation score is $(1.0 \times 4 \times 3) + (1.0 \times -5 \times 3) + (1.0 \times 0 \times 0) = -3$ (normalized score -0.04), since the positive preference for the matching genre is slightly weaker than the negative preference for the matching artist.

5.4.4 Constraining and Relaxing Attributes

Preflets support two kinds of attribute selection for constraining and relaxing attributes in the dialogue (see Section 5.3.2): the **generic-to-specific** approach, and the **preference-based** approach. Both approaches build on the concept of an ordered list of entity types that are constrained (or relaxed) in order.

The Generic-to-Specific Approach

With this basic approach, we pre-suppose that the entity types in a domain can be ordered in terms of generality. That is, certain types have the power to describe domain objects in a more general sense than other types. In the music domain, for example, Genre is more general than Album. This information is made available to the active preflet as an ordered list, along with a record of which attributes have been requested by the relax and constrain functions. As supported by the recommendation dialogue corpus in Chapter 3, the recommender agent starts by constraining the most general entity type and move on to specifics. The relaxation strategy takes the opposite direction, and tries to eliminate specific entity types first.

The Preference-Based Approach

The ordered list of entity types that need to be constrained or relaxed can be generated based on the contents of the preflet. We utilize importance function first, and in cases where importance function values are identical, the number of reliable entity values for each type. The algorithm can informally be described as follows for Constrain operations:

1. Order entity types (descending order) on importance function value (see Section 5.4.3).

2. If two types have the same importance function value:

   (a) Eliminate entity values for each type where reliability is 3 or higher. (We use the same threshold as for preference score evaluation.)

   (b) Order types of equal importance by counting the remaining values (the more values attached to a type, the more important it is considered to be for the user).

3. Reverse the list for Relax operations.

Example Consider the following preflet for user U:

User : U
Situation : Exercise
Genre$_{(2.0)}$ $\longrightarrow$ Techno$_{(5,2)}$, Dance$_{(4,3)}$
Artist$_{(2.0)}$ $\longrightarrow$ Scooter$_{(4,3)}$
Album$_{(1.0)}$ $\longrightarrow$ BodyCombat31$_{(4,4)}$, 'A Tribute to the King'$_{(3,4)}$
Year$_{(1.0)}$ $\longrightarrow$ [1990..1999]$_{(4,4)}$, [2000..2004]$_{(5,4)}$, 2007$_{(3,4)}$

When applying the preference-dependent approach, the ordered constrain list for this user would be:

1. Genre
2. Artist
3. Year
4. Album

Since Genre and Artist have the same importance (2.0), and the same number of values (the genre Techno is removed due to its low reliability), the default generic-to-specific order is retained. For Year and Album, both with lower importance function value than Genre and Artist, Year is ranked the higher of the two since three reliable values are connected to it compared to only two for Album. If relaxation are required in the dialogue, the strategy is thus to start from the bottom of the list instead.

There are times when choosing the static generic-to-specific approach to attribute selection is desirable. First, since constrain operations in the dialogue are carried out as part of the preference modeling process, there are often times when a preference-dependent approach cannot be used since there are no preferences (yet) to guide the selection process. Thus, it can also be seen as a fall-back strategy to the preference-based approach. Second, certain domains and applications might require constraining and relaxation to be carried out in the same order, even if there are preferences available.

5.4.5 Preference Updates and Conflicts

Human preferences are hard to quantify—especially in domains based on taste or gratifications, such as music, movie, and literature preferences. In some cases, this is a result of the inbuilt vagueness of human preference expressions in natural language[8]. In other cases, a user simply changes her preferences. A model aiming at maintaining a correct[9] user preference model must therefore be able to detect and resolve such issues, so that each entity value is assigned a single strength-reliability tuple in a preflet.

This section suggests that three main classes of update issues can occur (more or less frequently) when updating a preflet model. Section 5.4.6 presents resolution heuristics for each of these update issues.

- Accumulated Evidence Modification (`AEM`)
- Incomplete Comparative (`IC`)
- Conflict (`C`)

It is natural that preferences change between sessions; and sometimes even within sessions. Furthermore, preference detection is a cumulative process, and the system needs ways to combine several pieces of evidence for the an entity type and value into one combined preference strength. The issue of *Accumulated Evidence Modification* (`AEM`) deals with the problem of deciding when to overwrite previously collected preferences, or how to modify preferences, given new acquisitions.

Since we allow for comparative preference statements, such as *x is better than y*, we need to consider cases where the comparison argument is unknown (e.g. the preference for y in the example above). And even if the comparison argument is known, it is not trivial to quantify *how much* better or worse something is compared to the comparison argument. This issue is called the issue of *Incomplete Comparative* (`IC`) and can be broken down into two cases:

- `IC1`: $x \rhd y$, $pstr(y)$ is known.

---

[8] As noted in Chapter 2, we do not try to capture the quantitative difference between *awesome*, *great*, *super*, *splendid* or *extraordinary*, *brilliant* etc.

[9] Rather, "as correct as possible". There is no evidence that even human recommenders are able to maintain a "correct" preference model of recommendation dialogue partners.

- `IC2`: $x \rhd y$, $pstr(y)$ is unknown.

Here, $pstr(y)$ is the preference strength (including polarity) for entity type or value y. The problem for resolving `IC` is to assign a correct preference strength for x.

The third case—the issue of *Conflict* (`C`)—is also divided into sub-cases[10] (`C1-5`). They involve incompatible preference statements in the same preflet. The following list provides the conflict cases. Note that the temporal order in which the preferences are detected matters:

- `C1`:
    1. a superlative $\boxplus x$ is detected.
    2. a comparative $y \rhd x$ is detected, causing $pstr(y)$ to “hit the ceiling”.
- `C2`:
    1. a comparative $y \rhd x$ is detected.
    2. a superlative $\boxplus x$ is detected, which violates (1).
- `C3`:
    1. a comparative $x \rhd y$ is detected.
    2. accumulated evidence for $\oplus y$ results in $pstr(y) > pstr(x)$, thus violating (1).
- `C4`:
    1. The calculation of multiple sentinels results in conflicting strengths.
- `C5`:
    1. Circle reference between two sentinels.

`C1-3` naturally have inverses `C1'-C3'` where the preference operators are inverted. For instance, the temporal order for `C1'` is identical to `C1`, but preference operators are inverted:

---

[10] All sub-cases dealt with here involve comparatives. Conflicts concerning descriptive preferences are identified and resolved by Carberry *et al.* [1999]

1. a superlative $\boxminus x$ is detected.
2. a comparative $y \lhd x$ is detected.

The next section suggests heuristics for resolving the described update issues.

5.4.6 Preflet Update Resolution Heuristics

Resolving Accumulated Evidence Issues (AEM)

Continuous updates of a user's preferences given the content of an on-going dialogue is detailed by Carberry *et al.* [1999]. We use their approach by utilizing endorsements as reliability measures, as briefly outlined in Chapter 2. The interested reader is encouraged to study the original work by Carberry *et al.*.

Resolving Incomplete Comparatives (IC)

Comparative preference statements state something about the relationship between two entity types or values. Section 5.2.3 details PCQL FP state formulas for comparative preference statements. If we allow for such statements and want to do something useful about them in a preference modeling framework (as R1c in Table 5.1 dictates), we need a way to maintain such comparative relations over time. The sentinel concept is used to accommodate this. Sentinels are assigned to the first of the two arguments that are part of any comparative statement.

IC1 The system simply assigns x with a modification sentinel

$$< \ddagger pstr(y) : +1 >$$

There may well occur situations where previously detected preference strengths indicate that the user thinks that x has a higher strength than y. A more recent comparative statement, such as `IC1`, that contradicts this will have precedence and overwrite the old $pstr(x)$. The modification solution is thus a direct application of the *recency* guideline.

IC2 Simple enough, `IC2` is handled in a similar fashion as `IC1`, but with a slight addition. The modification sentinel is identical, but since there is no previous record of $pstr(y)$ the system assigns it a default strength of 3, with the weakest form of reliability (1). This means that $pstr(x)$ will receive a value of 4 when the preflet is updated.

Resolving Conflicts (C)

One general approach to resolving potential conflicts in our model is to put emphasis on the most recent discovered preference. The reason for this is grounded in the definition from Chapter 2, which states that preferences are not goals that are fully specified at the outset of the interaction, but rather come into play as users must evaluate alternatives. This is echoed by the exploratory character of the interaction found in the empirical analysis in chapters 3 and 4.

With the guideline of *recency* in mind, we outline heuristics for resolving the conflicts `C1-5`. Inverses (`C1'-3'`) are handled symmetrically by reversing polarity on sentinels and strengths.

C1 Superlative preferences are treated as very strong direct preferences, independent of which conversational circumstance they occur in. Thus, a preference $\boxplus x$ is assigned as $pstr(x) = 6$. If a comparative preference $y \rhd x$ is detected later, the `C1` conflict arises. Since there is no room for a higher preference strength beyond 6 the solution is to assign $pstr(y) = 6$ and attach a sentinel $< \ddagger pstr(y) : -1 >$ to $pstr(x)$. A viable alternative would be to leave the original $pstr(x) = 6$ since it is (was) a very strong preference. However, the former solution embraces the *recency* guideline and encourages a dynamic preference model[11].

C2 `C2` has the same preference statement types as `C1`, albeit in different order. Note that we assume that the respective strengths of the comparative statement `C2(1)` has been resolved using either `IC1` or `IC2` (above). The simplest solution to `C2` is to simply treat the superlative `C2(2)` as a stand-alone superlative; i.e. $pstr(x) = 6$.

---

[11] This approach is suitable for "taste"- or gratification-based domains, but it is possible that the latter alternative would be suitable for more objective domains (such as purchasing a digital camera, where preferences concern hard requirements such as size, resolution, battery longevity, etc.).

The previously detected preference strength for y remains as is. However, strictly, this imposes problems of maintaining the *meaning* of the comparative `C2(1)`. An alternative solution would be to allow $pstr(x)$ and $pstr(y)$ to both "hit the ceiling" when the superlative preference for x is detected (independently of what $pstr(y)$ was before); thus trying to conserve the preference of `C2(1)`. This could be viewed as a problem—and violation of the recency guideline—since a user may potentially "get stuck" with an old comparative preference. We argue for the former option, since we view the `C2(1)` and `C2(2)` as a case of the user "changing her mind" regarding x.

C3 This potential conflict deals with the fact that indirect preference statements (usually factual questions) may increase the preference strength through combination [Carberry *et al.*, 1999]. A previously detected comparative `C3(1)` may thus in theory be overridden by a series of indirect preference statements. Since the system should respect any direct preferences (such as the comparative `C3(1)`), we need a heuristic to avoid this. Simply put; accumulated preferences via indirect preference utterance types never supersede direct preference statements. For instance, if a user has previously said that he likes *Jazz* better than *Rock*, but accumulates a strong positive preference toward *Rock* based on *"Rock"* queries (indirect preferences), the model does not infer that the user thinks that *Rock* is better than *Jazz*. The solution is to apply a maximum sentinel on the preference strength for (Genre = Rock) and connect it to the preference strength for (Genre = Jazz). For this example we assume that any `IC` issue has been resolved so that the preference strength for (Genre = Jazz) is known to be 4, and that the reliability is known to be 3. The resulting preflet from this example in a specific dialogue state could be written as:

$$\mathsf{Genre}_{(1.0)} \longrightarrow \mathsf{Jazz}_{(4,3)}, \mathsf{Rock}_{(3,3,<\top(\mathsf{Genre}=\mathsf{Jazz})>)}$$

If a user later uses a (direct) comparative statement that affects these preferences, such as *"I think that rock is better than jazz"*, the associated maximum sentinel is replaced with a standard comparative modification sentinel.

C4 and C5 Solving conflict types 4 and 5 could be an interesting application for *constraint (logic) programming*. However, this is beyond the scope of this thesis.

From an interaction point of view, it might also be better to *not* try and resolve conflicts "under the hood", but instead fuel the dialogue by bringing the conflicts up for discussion with the user. Awaiting either a constraint programming solution, or explicitly discussing conflicting preferences in the dialogue, we present a somewhat naive solution to the problem: In order to avoid combinatorial explosion we do not allow sentinels referring to other sentinels. The older of the two sentinels is "grounded" as a fixed strength, and the more recent sentinel is constructed as usual. In cases where the older sentinel's value cannot be determined, a medium strength and low reliability is suggested (e.g. strength 3 and reliability 2). However, these values can be tuned in the given application.

5.5 Summary

We have provided a user preference model framework that supports and utilizes natural language dialogue tailored for recommendation dialogue. The work extends the preference-detection model of Carberry *et al.* [1999] with an additional conversational circumstance, and with the addition of comparative and superlative preference statements. The framework is centered around the concept of preflets, which are collections of situation-based preferences for a particular user, allowing users to have different preferences for the same entity type or value in different situations. Preflets also allow a recommender engine to use the preference evaluation score formula to estimate predictions of domain items based on a specific preflet.

The PCQL formalism was also introduced. PCQL handles preferential and factual statements and requests by the use of FP state formulas as arguments to action tags.

PCQL and preflets form important constructs for the recommendation dialogue model BCORN presented in the next chapter.

6

BCORN: Managing Recommendation Dialogue

This chapter presents the BCORN *model for a generic recommendation dialogue strategy with conventional, information-providing, and recommendation capabilities. An implementation of* BCORN *is also described: the music conversational recommender system* CoreSong*, which is evaluated in Study III with end-users in order to verify the behavior-based conversational recommendation strategy.*

THIS CHAPTER TIES together several aspects of the previous chapters into the BCORN recommendation dialogue strategy model. First, we introduce dialogue behaviors which are expressed computationally in dialogue behavior diagrams (Section 6.1). Second, BCORN is described (Section 6.2). The BCORN model is constructed using dialogue behavior diagrams that each describes a natural chunk of an agent's dialogue strategy. The empirical base for the behaviors of BCORN was presented in Chapters 3 and 4. Since a central concept of BCORN is that recommendation dialogue is viewed as an emergent phenomenon of several dialogue behavior diagram instances that are run in parallel, Section 6.3 discusses how emergence is handled. BCORN is implemented in the CoreSong conversational music recommender system, which is covered in Section 6.4. CoreSong is used in an overhearer end-user evaluation, described in Section 6.5.

6.1 Dialogue Behavior Diagrams

The basic concept of the architecture outlined in this chapter is the *dialogue behavior*. A dialogue behavior is an informal term denoting a result of linguistic actions in a context (see Chapter 2). Some dialogue behaviors are general (e.g. a conventional dialog behavior of greeting and farewell), and some are specific (e.g. a ticket booking dialogue behavior). A dialogue agent thus needs a dialogue strategy model that includes dialogue behaviors that can co-exist but at the same time have a clear order of priority. It is also imperative that the model adjusts to the needs of different back-end resources at hand in a particular application.

Similar to the behavior-based model proposed by Brooks [1991b], BCORN is constructed using state automata organized in *strata*[1]. A stratum is a technical term that refers to internal structures used to generate dialogue behaviors. Strata express dialogue behaviors of the dialogue agent that are both natural conceptually and efficient computational mechanisms. The complete dialogue strategy of an agent is the result of running several strata in parallel in a strata machine, leading to an emergent agent behavior. A desirable characteristic of such emergent behavior is to be coherent, flexible, and effective.

Computationally, a dialogue behavior is coded into a state automaton. For example, “answering a question”, or “greeting a dialogue partner” may be considered as natural dialogue behaviors, if these notions also can be coded into a well-behaved state automaton.

The states of the automaton we use are decorated with *command* statements, that can be either atomic statements or variable assignments. The special variable `%`

---

[1] In order to avoid confusion with all implications from Brooks’ original work, which has inspired this work *in part*, we use the term *stratum* instead of layer.

always holds the return value of the previously issued command.

We define three different commands: in, out, and call.

in The in command reads tokens from a designated input stream.

out The command out takes one argument which it writes on a designated output stream. Each automaton uses one separate unique input and one output stream (which is sufficient for the needs of BCORN, but could be generalized).

call The command call is used to invoke and use results from other software modules, denoted jointly as *external resources*. The call command has the signature call $\langle$ext-resource, $\{$arg$\}^*\rangle$. Here ext-resource is the name of an external resource and $\{$arg$\}^*$ is a list of arguments sent to the ext-resource in the invocation. Each invocation of an external resource returns a value, but they are not necessarily functional modules and often carry an internal persistent state.

A *Dialogue Behavior Diagram* (DBD) describes a state automaton where each state contains (one or more) *commands* and transitions are decorated with (possibly empty) *conditions*. The DBD automaton is similar to the UML activity diagram[2]. The conditions on the state transitions are standard boolean expressions, formed using variables and primitive functions for the value data types of those variables[3].

A DBD is executed by traversing the transitions and executing the commands like a flow chart execution. The transitions are fired when the conditions of the variables are met. That is, the transitions do not consume input. Instead, it is the in command that consumes input. The DBD automaton will pause each time it reaches a state with the command in, until the next input token is available. The DBD automaton will stop indefinitely if it reaches a state where there is no outgoing transition with a satisfied condition.

The execution of the DBD automaton can be expressed with automata trajectory semantics. We assume a *state environment* as a part of each state in which the

---

[2] States are depicted as a square with rounded corners. Entry nodes are depicted as black dots and exit nodes as black dots with a surrounding circle. An exit node marks that a stop is normal.

[3] We use C-style boolean operators (==, !, >, <).

commands and conditions are executed. The state environment consists of the input and output streams, and a value assignment for the variables (including `%`).

An *execution trajectory* $\mathcal{T}$ of a DBD automaton $\mathcal{D}$ for a given input stream $\mathcal{I}$ is a sequence $s_1 \overset{c_1}{\Rightarrow} s_2 \overset{c_2}{\Rightarrow} \ldots \overset{c_{n-1}}{\Rightarrow} s_n$ such that:

- s_1 is a start state of $\mathcal{D}$ with a state environment that includes $\mathcal{I}$, an empty output stream and an empty variable assignment map.
- s_{i+1} is a state obtained following an outgoing transition with a condition c_i of the state s_i where:
    - the state environment of s_{i+1} is obtained by executing the commands of s_i;
    - the condition c_i is evaluated to true in the state environment of s_{i+1}.

By the *produced* state environment of $\mathcal{T}$ we understand a (final outgoing) state environment obtained by executing the commands of the last node s_n in the state environment of s_n. By the *produced* output stream $\mathcal{O}$ of $\mathcal{T}$ we mean the output stream $\mathcal{O}$ found in the produced state environment of $\mathcal{T}$. Let $\mathcal{D}$ be a deterministic DBD[4], and $\mathcal{T}$ the unique execution trajectory for a given input stream $\mathcal{I}$. Then we call the produced output stream of $\mathcal{T}$ for the *dialogue behavior* of $\mathcal{D}$ w.r.t. $\mathcal{I}$.

Similar to the UML activity diagram[5], the DBD can use a sub-diagram within a super-state, called a *sub-behavior* DBD. The sub-behavior diagrams use explicit entry and exit nodes, similar to activity sub-diagrams. The value of the `%` variable when exiting the exit node is the same as that of the previous node. This value is also the value of the whole diagram that is exported to the invoking super DBD after the exit point.

From an interaction perspective, the DBD design supports the three-entity interaction model [Ibrahim and Johansson, 2002b] (see Chapter 2) since the `in` and `out` commands in the DBDs implement the user–agent linguistic communication interaction, and the `call` command implements the agent's domain information presentation.

---

[4] We only consider deterministic automata, since this is all we need in practice.

[5] Since activity diagrams naturally relate to statechart diagrams, the emerging standard of W3C's State Chart XML is an option for specifying dialogue behavior diagrams.

Exactly what is, and what is not, considered a "reasonable" dialogue behavior is a design question which the DBD framework leaves open to the DBD developer. The DBD is a conceptual framework in which the actual design of the BCORN model can be expressed.

6.2 BCORN Dialogue Behaviors

This section details the four DBDs that constitute the BCORN model: **Conventional**, **Direct Delivery**, **Indirect Delivery**, and **Interview**. The dialogue behaviors of these four DBDs have been conceptualized through the empirical investigations described in Chapters 3 and 4.

In an implementation, the DBDs are *instantiated* into strata which means that the names of external resources in the DBD are connected with implementations of said external resources. Each DBD has as many DBD instances as needed for a particular application.

6.2.1 Conditions and PCQL in BCORN

Before detailing the behaviors, a description of the conditions used in the DBDs is needed. As the internal message-passing format for BCORN is PCQL, a set of specific condition functions are defined that query FP state formulas of PCQL action statements (see Chapter 5). Table 6.1 shows the available condition functions for BCORN DBD instances. These convenience functions are used directly as guards on the DBD transitions. Examples of their use is found in the upcoming sections.

6.2.2 PCQL Action Tags and Dialogue Acts

In the recommendation dialogue characterization presented in Chapter 3, a set of dialogue acts was described. Before moving on to the DBD descriptions below, we will summarize the relation between the PCQL action tags found in the diagrams, and the dialogue acts found previously. BCORN employs PCQL action tags that specify recommendation dialogue in detail for two reasons: (a) generic dialogue acts are often not specific enough to aid in the user preference modeling [Pohl *et al.*, 1995], and (b)

Table 6.1: BCORN condition functions and return types.

Function	Return	PCQL example (x)	Return
get-value(attr, x)	String	INFORM ⟦ (YN = Yes) ⟧ *Note:* (attr = YN)	Yes
get-tag(x)	String	INFORM ⟦ $\ominus_\diamond$ (Genre = 'Rock') ⟧	INFORM
no-of-values(x)	Integer	RESULT ⟦ π Genre ⟧ VALUES ⟦ {Jazz, Rock} ⟧	2
is-empty(x)	Boolean	RESULT ⟦ π Genre ⟧ VALUES ⟦ {Jazz} ⟧	*false*
is-ask(x)	Boolean	ASK ⟦ π Album, (Artist = 'The Hives') ⟧	*true*
is-neg-pref(x)	Boolean	INFORM ⟦ $\oplus_\diamond$ (Artist = 'Moby') ⟧	*false*
is-pos-pref(x)	Boolean	INFORM ⟦ $\oplus_\diamond$ (Artist = 'Moby') ⟧	*true*

the action tags specify and aid in the template-based generation process. Table 6.2 lists the action tags of BCORN's dialogue behavior diagrams, and Table 6.3 shows the available action tags attached to PCQL interpretations of user utterances.

6.2.3 Conventional

Conventions form the basis of dialogue sessions, and include opening and closing of dialogs. Figure 6.1 shows a DBD describing BCORN's conventional dialogue behavior. It starts with a welcome message as the PCQL action statement

WELCOME⟦ ⟧

in node 11 (Figure 6.1). Greetings and farewells will be returned as expected (nodes 13 and 14), and any other user input will be followed by an exhortation to try again (node 15). The transition between node 12 and 13 illustrates the use of BCORN's condition functions. Consider the case where the user utterance *"Good bye"* has been interpreted as the PCQL statement

BYE⟦ ⟧

The get-tag function takes the incoming PCQL statement as argument and returns the action tag (BYE), causing the guard on the transition to evaluate to *true*.

On its own, the dialogue behavior of the conventional DBD does not do much to

Table 6.2: System action tags and dialogue acts in CoreSong.

Action Tag	**Dialogue Act**
Conventional	
GREET	OPENING
BYE	CLOSING
TRY-AGAIN	FACTUAL-REQUEST
WELCOME	OPENING
Direct Delivery	
RESULT	FACTUAL-STATEMENT
RELATED	FACTUAL-STATEMENT
NEXT	REQUEST (pref. or fact.)
Interview	
NORESULT	FACTUAL-STATEMENT
EXCESS	FACTUAL-STATEMENT
RELAX	PREFERENCE-REQUEST
CONSTRAIN	PREFERENCE-REQUEST
Indirect Delivery	
MOTIVATE	MOTIVATION
RECOMMEND	RECOMMENDATION
ACKNOWLEDGE	ACKNOWLEDGE
ASKRATE	PREFERENCE-REQUEST

Table 6.3: User action tags and dialogue acts in CoreSong.

Action Tag	**Dialogue Act**
GREET	OPENING
BYE	CLOSING
ASK	FACTUAL-REQUEST
INFORM	FACTUAL-STATEMENT
	PREFERENCE-STATEMENT
YES	YES
	ACKNOWLEDGE
NO	NO

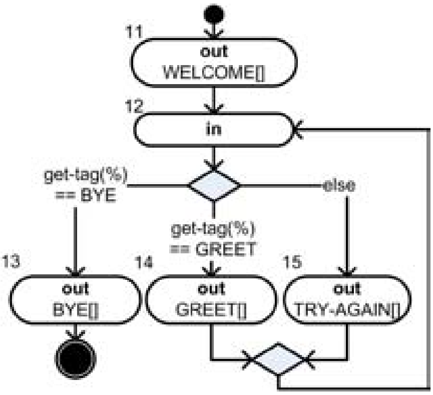


Figure 6.1: The Conventional DBD.

help a user with any task. However, it is a well-defined dialogue behavior that fits as a part of a dialogue agent's complete strategy.

A user of information-providing or recommender systems expects some sort of useful delivery. This is catered for by the dialogue behaviors of BCORN's other DBDs as detailed below.

6.2.4 Direct Delivery

On a fundamental level, the goal for a BCORN system is to provide the user with a delivery. This can, for instance, be an explicitly requested piece of information, or a recommendation of some sort. Figure 6.2 shows the direct delivery DBD. For user statements of the type ASK (requests), the DBD uses a call LookUp in node 24, typically to a database, or any information repository that the user may wish to query. In cases where a successful call LookUp has been made (that is, a non-empty result set not larger than a predefined size limit is returned), a direct delivery in the form of a PCQL statement with a RESULT tag and an FP formula with a VALUES part is passed as an argument to the out command in node 25.

In other cases, the call LookUp may not be successful and hence no delivery is made. In such cases the DBD returns to node 21 and awaits new user input. Such cases are handled by the interview behavior (see Section 6.2.5).

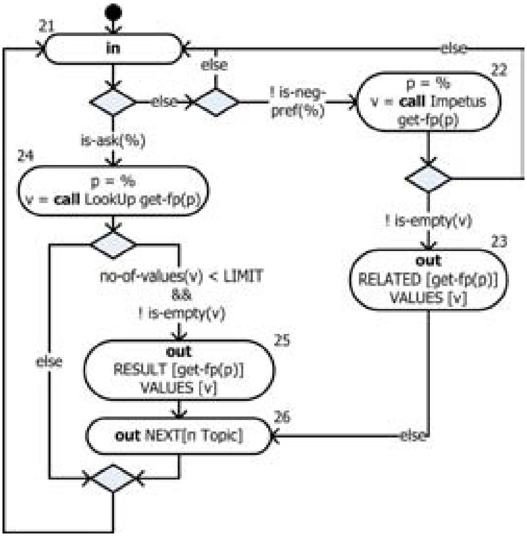


Figure 6.2: The Direct Delivery DBD.

In order to support domain exploration and to drive the dialogue forward (conversational impetus) and to support preference elicitation by presenting related information (see Chapters 3 and 4), one extra step is performed in the case of positive preference statements. Node 22 transforms such statements into tailored requests, and fetches information *related* to the preference from an external resource in node 23 (e.g. utterance S1a in Table 6.4).

Note that the Direct Delivery DBD only has the power of delivering results; and is only helpful when the user has provided a complete query. A more cooperative agent will help the user to complete erroneous or ambiguous requests. Such help is available in the *interview* dialogue behavior.

6.2.5 Interview

The purpose of the interview is to collect relevant information about domain entity types (e.g. genres, artists or albums in the music domain) or specific items (e.g. song titles). Figure 6.3 shows the Interview DBD. This is a useful dialogue behavior in

Table 6.4: Example of direct delivery in the music domain. Known entity types and values are in *italics*. `[list]` denotes domain information from the database. S = system, U = user.

	Dialogue	**in/out**
U1	*Pearl Jam* is an excellent band	21
S1a	Ok. *Pearl Jam* has made the following albums: `[list]`	23
S1b	What else can I do for you?	26
U2	What songs are on the album *Vitalogy*?	21
S2a	These are the songs on the album *Vitalogy*: `[list]`	25
S2b	What else can I do for you?	26

cases where user requests cannot be completed (perhaps due to an incomplete or ambiguous request, or if there are no matches to a valid request). It is also useful in preferential interviews used for recommendations, where the agent acquires user domain preferences in order to construct user preference data for a recommender engine.

Even though they implement the same DBD, the point of difference between the two is that the former has the characteristic of a specific goal formulated by the user, whereas the latter is dynamic and depends on the underlying external resource LookUp. That is, for preferential interviews there is not a specific number of attributes that need to be filled (as in a static slot-filling approach), but rather depends on the kind of attributes and values elicited by the user and how useful these are to the recommender engine.

If call LookUp returns an exceptional result (such as no, or too many, matches in a database) a clarification sub-dialogue is needed to sort things out. This falls under the dialogue behavior described by this DBD since the system asks the user to complete her original request. Table 6.5 shows an example of the resulting surface generation for the execution trajectory[6]

$$51 \Rightarrow 54 \Rightarrow \ldots \Rightarrow 57 \Rightarrow 51$$

in Figure 6.3. In this example, the incoming user utterance U1 is formalized as the

---

[6] We omit conditions on the transitions when these are not relevant for the ongoing discussion.

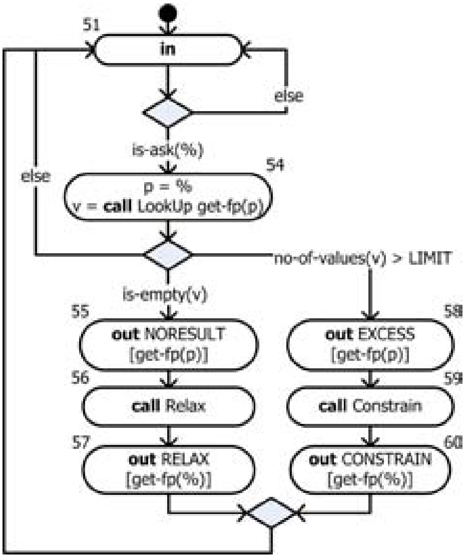


Figure 6.3: The Interview DBD.

PCQL statement

$$\text{ASK} \llbracket \pi \,\mathsf{Album}, (\mathsf{Artist} = \text{'}\mathsf{The\ Hives}\text{'}), (\mathsf{Year} < \mathsf{1985}) \rrbracket$$

Since we have an ASK tag (and a projection π in the FP state formula), the condition IS-ASK on the transition between nodes 51 and 54 evaluates to *true*. The invocation of LookUp in node 54 results in an empty result set, since the query is valid (i.e. all entity types are recognized) but no entries in the database satisfy the given constraints (*artist* and *year*). The condition is-empty on the transition between nodes 54 and 55 therefore evaluates to *true*, causing the transition to node 55, where the status is written to the out stream with the command out. A surface realizer may later generate this is as utterance S1a in Table 6.5. Following the interview dialogue behavior, node 56 invokes call Relax. This external resource's task is to return an entity type that needs to be relaxed[7]. In our example, year is the chosen type. This

---

[7] The implementation of Relax can be of varying complexity, such as a static ordered list of types, or a more elaborate algorithm that calculates the optimal type to be relaxed. The preflet solution (Chapter 5 is suitable for this example.

Table 6.5: Example of an interview with attribute relaxation in the music domain. Known entity types and values are in *italics*. S = system, U = user.

	Dialogue	**in/out**
U1	List all *albums* by *The Hives* released before 1985.	51
S1a	There are no *albums* that match.	55
S1b	Do you want to relax the *year* constraint?	57
U2	Ok	51

is incorporated into an FP state that is assigned to the % variable. Node 57 invokes the command out with the following PCQL action statement:

$$\text{RELAX} \llbracket \pi\, \mathsf{YN}, \mathsf{Year} \rrbracket$$

This statement is surface realized as S1b in Table 6.5. The DBD transits to node 51, awaiting new user input.

Queries that result in very large result sets (i.e. exceeding a defined limit number of values) need constraining instead of relaxation. Constraints are handled in a similar manner as relaxations described above.

If the Interview DBD is used in conjunction with a recommender engine and a preference model, the agent's role is to interview the user for preferences in order to be able to provide recommendations. Then the LookUp interface is implemented using a recommender engine[8] which—like the external resource for information database lookups in the previous section—returns sets of items. This kind of interview is exemplified in Table 6.6, which corresponds to the execution trajectory:

$$51 \Rightarrow 54 \Rightarrow 58 \Rightarrow 59 \Rightarrow 60$$

The recommender engine used in the external resource LookUp delivers a list whose number of values is larger than the pre-defined limit, causing the DBD to transit to

---

[8] We have experimented with content-based and hybrid engines [Burke, 2002]. A pure collaborative filtering engine does not make use of entity type preferences, but rely solely on explicit item ratings. Such a DBD 's external resources Relax and Constrain would have to operate on items rather than entity types. This does not, however, affect the design of the Interview DBD.

Table 6.6: Sample preference interview in the music domain. S = system, U = user.

	Dialogue	**in/out**
U1	I like the artist *Pearl Jam.*	51
S1a	I need more preferences in order to give you recommendations.	58
S1b	Please provide a genre you like.	60

node 58 in Figure 6.3 (corresponding to utterance S1a in Table 6.6). The external resource Constrain utilizes the user's preference model (e.g. a preflet) to find an entity type that the user has not yet given any preferences for, and returns a corresponding PCQL statement with a CONSTRAIN tag. Node 60 then invokes out with the complete PCQL action statement

$$\text{CONSTRAIN} \llbracket \oplus \mathsf{Genre} \rrbracket$$

as argument, which is later realized as utterance S1b in Table 6.6. Preferential interviews are the foundations for the second kind of delivery in BCORN: the indirect delivery.

6.2.6 Indirect Delivery

In certain situations, the direct delivery dialogue behavior is insufficient. Rather than a straight-forward lookup and black-box presentation of an explicitly requested piece of information, it is in the nature of some resources to deliver results "when they are ready". A typical case of this is conversational recommendations based on incrementally acquired user preferences detected using the interview. A delivery in this case (i.e. an item recommendation) can be made only when enough preferences have been acquired, and is not a direct response to a user-initiated query. Instead, the agent takes initiative as soon as it has a recommendation ready. In such cases the agent is usually unaware of if the delivery is previously unknown or known to the user, or if it truly satisfies the user's preferences.

Following design guidelines for recommender system interaction design, a recommender agent should (a) deliver a recommendation modestly (since we cannot know whether the recommended item is unknown and new to the user), (b) monitor the user

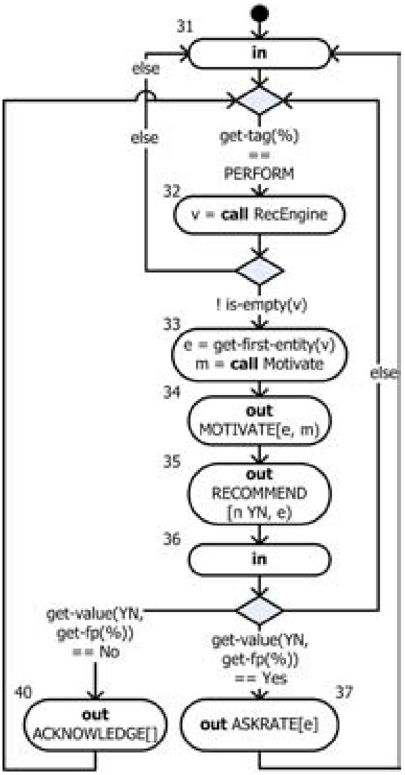


Figure 6.4: The Indirect Delivery DBD.

response to the recommendation, and (c) convey system transparency by explaining the rationale behind the recommendation (see Chapters 2, 3, and 4). Consider the dialogue excerpt in Table 6.7 as an instance of the execution trajectory

$$32 \Rightarrow \ldots \Rightarrow 37$$

in the DBD depicted in Figure 6.4. In this example, the external resource LookUp is implemented as a hybrid content-based and collaborative filtering recommender engine. In node 32 the returned result-set satisfies the condition on the outgoing transition from node 32 to 33. After necessary house-keeping (the *e* variable assign-

Table 6.7: Sample recommendation in the form of an indirect delivery in the music domain. S = system, U = user.

	Dialogue	**in/out**
S1a	You might like the song *Higher* by *Creed* in the genre *Alternative Rock* because it is liked by others who like *Pearl Jam*.	34
S1b	Have you heard it?	35
U2	Yes	36
S2	What do you think about it?	37

ment) an external resource Motivate is invoked. This resource's responsibility is to generate a motivation for the recommendation, based on this particular user's preference profile. A preceding interview has resulted in several preferences for this user's profile, including positive ratings for the genre *Alternative Rock* and the artist *Pearl Jam*, which is why these are used in the motivation (S1a)[9]. The actual delivery is then made as a Yes/No question in node 35. The DBD then stops in node 36 awaiting new input. Since the PCQL FP state formula of the incoming user PCQL statement has the entity type YN with value Yes, the DBD transits to node 37 where a follow-up question is performed (node 37) in order to get feedback on the already known recommendation.

6.3 Emergent Dialogue Strategy

Hitherto, we have considered individual dialogue behaviors implemented as DBDs. In particular, we have discussed the BCORN DBDs and explained their dialogue behaviors. A dialogue agent's *complete* behavior is described by a set of DBD instances that each runs as a DBD *strata machine*. Each stratum in a DBD strata machine consists of one DBD instance. Furthermore, the DBD strata machine streams input and merges each stratum's output into one coherent PCQL representation of the agent's turn.

Since each DBD instance in a DBD strata machine does not model any other DBD

---

[9] The collaborative filtering aspect is evident in the motivation phrasing, since one can refer to other users' music taste [Herlocker *et al.*, 2000].

instances, and runs autonomously on the same input, there is no central control of the dialogue. The dialogue systems complete behavior instead emerges from the different DBDs [Wärnestål *et al.*, 2007a]. By *emergent* functionality in a system, we understand components that operate simultaneously in order to achieve a desired behavior. This is contrasted to hierarchical systems, where sub-functions are invoked from a central component or representation. According to Steels [1990, page 454]:

> a component implements a behavior whose side effect contributes to the global functionality [...] Each behavior has a side effect and the sum of the side effects gives the desired functionality.

This approach to dialogue system design is inspired by the layered subsumption architecture (see Chapter 2) where layers correspond to behaviors that are organized hierarchically.

It is important to note that a DBD strata machine generates PCQL statements, and for most applications the raw output stream will be a case of over-generation. That is, any specific application needs to consider reduction and integration of the PCQL output. By pushing such decisions beyond the DBD strata machine we achieve a less complex and more generic dialogue management mechanism. Thus, the agent's complete dialogue strategy emerges from combinations (weaving) of the DBD instances' outputs. The dialogue manager itself does not determine surface strings to send to the natural language generator module(s). A BCORN implementation outputs PCQL statements, which are tailored to be on a sufficient level of abstraction for the generation task. Since each DBD instance operates without modeling its siblings in the DBD strata machine, the outputs from the DBDs need to be integrated (and typically reduced) so that the system utterance makes sense to the user. Combining the behaviors' output into a coherent system turn is by no means trivial, since it is important to design for the coherence, informativeness, and flexibility required for natural language dialogue. One pragmatic way that has been found sufficient in the implementations is to use two heuristic principles, which are further explained in Section 6.4.2, where the approach is exemplified with the CoreSong implementation.

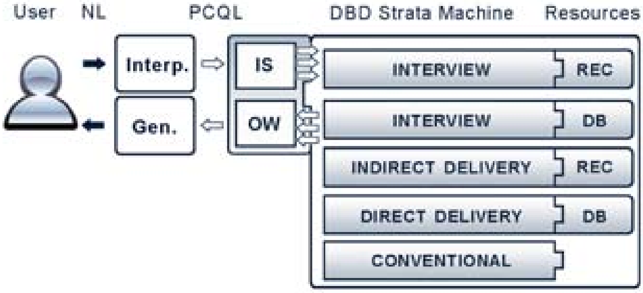


Figure 6.5: The standard CoreSong behavior configuration. Two external resources, database (DB) and recommender engine (REC), coupled with interview and different delivery behaviors. Interp = Interpretation Module, Gen = Generation Module, IS = Input Streamer, OW = Output Weaver. This constitutes configuration 4 in the experiment, see Section 6.5.

6.4 Implementing BCORN: CoreSong

Let us consider the role of BCORN in a complete conversational recommender system. This section describes such a system, called CoreSong. The architecture in Figure 6.5 shows CoreSong's DBD strata machine (with five DBD instances) and auxiliary components. CoreSong is a music conversational recommender system that allows users to engage in recommendation dialogue as outlined in the previous chapters. The interaction is typed and the language is English.

Detected user preferences are used in two ways: First, it is naturally used to generate new music recommendations. Second, it aids the user in sorting any local music files the user might possess. CoreSong thus deals with already "known" music as well as previously unknown music. Since CoreSong is deployed as a web application that runs in a web browser it requires the user's permission to upload music information in the server databases. Technically, the application is written in Ruby[10], and its back-end consists of MySql[11] databases. The user's local music collection information is taken by parsing mp3 tags for genre, artist, and album of the files and uploaded in a MySql database on the server.

---

[10] Ruby version 1.8.5. and Ruby on Rails version 1.1.6.

[11] MySql version 5.0.27

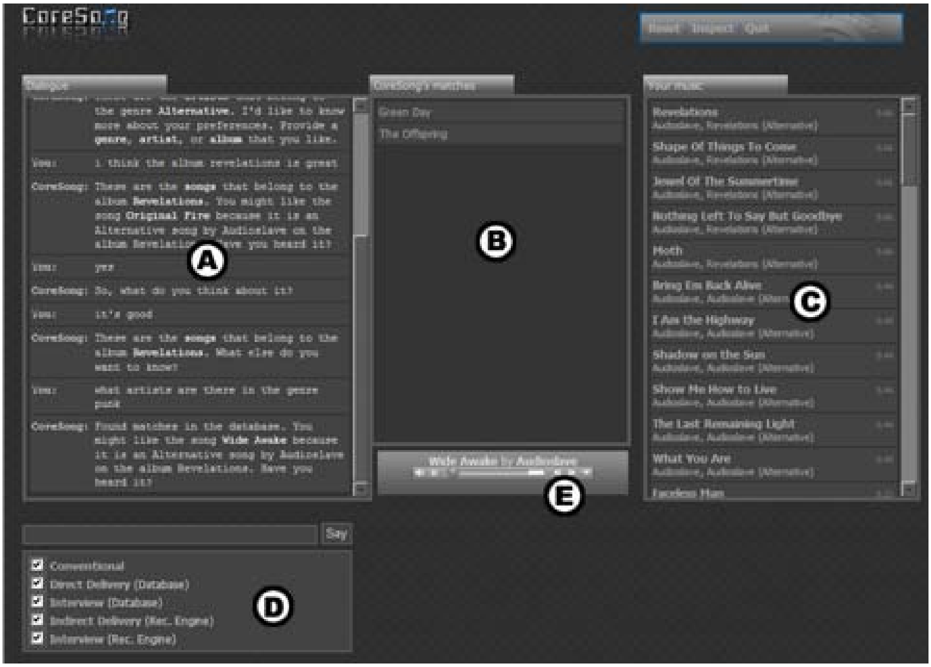
There are three external resources in the implementation. First, there is a database with music information. It encapsulates genre, artist, album, and year information for 8,500+ songs (distributed over 40+ genres, 2,000+ artists, and 650+ albums). The music database is connected to one Direct Delivery DBD instance, as well as an Interview DBD instance (see Figure 6.5). Second, there is a content-based recommender engine that utilizes the same music information as the music database. An experimental collaborative filtering hybrid recommender engine has been used in some tests, where user ratings influence the choice of recommended items, but in the version used in Study III (see Section 6.5) the content-based version was used. Third, CORESONG uses the preflet construct described in Chapter 5, and preflet information is stored in MySql tables in the back-end.

Figure 6.6 shows CORESONG' user interface. In Figure 6.6, **A** is the dialogue chat area, and **B** is the view of database results. Furthermore, **C** is the area in which the user's local music collection is displayed (and sorted according to the active preflet). **D** is the strata configuration, where DBD instances can be turned on and off. This feature is not intended to be used by end-users in a sharp version; rather, it is used in order to control the generation of dialogues in the experiment (see Section 6.5). Moreover, the current recommendation is presented in the area **E**, where standard playback controls allows the user to start and stop the recommended song. The design of the user interface follows the guidelines of separating linguistic communication (**A**) between the user and the agent from the views and direct manipulation interaction (**B**, **C**, and **E**) of information from the back-end in the three-entity model [Ibrahim and Johansson, 2002b]. The next two sections describe CORESONG's input and output functionality.

6.4.1 Input: Interpretation and Streaming

In general, the two auxiliary components Interpreter and Generator (see Figure 6.5) can be arbitrarily advanced depending on what natural language interpretation and generation capabilities a particular system is required to have.

In CORESONG, the Interpreter takes typed natural language as input and produces PCQL representations as output, using primarily the tags GREET, BYE, ASK, and INFORM. Other tag sets are obviously possible, but for the purpose of executing the



the genre **Alternative**. I'd like to know more about your preferences. Provide a **genre**, **artist**, or **album** that you like.

You: i think the album revelations is great

CoreSong: These are the **songs** that belong to the album **Revelations**. You might like the song **Original Fire** because it is an Alternative song by Audioslave on the album Revelations. Have you heard it?

You: yes

CoreSong: So, what do you think about it?

You: it's good

CoreSong: These are the **songs** that belong to the album **Revelations**. What else do you want to know?

You: what artists are there in the genre punk

CoreSong: Found matches in the database. You might like the song **Wide Awake** because it is an Alternative song by Audioslave on the album Revelations. Have you heard it?

Green Day
The Offspring

Wide Awake by Audioslave

Revelations
Audioslave, Revelations (Alternative)
Shape Of Things To Come
Audioslave, Revelations (Alternative)
Jewel Of The Summertime
Audioslave, Revelations (Alternative)
Nothing Left To Say But Goodbye
Audioslave, Revelations (Alternative)
Moth
Audioslave, Revelations (Alternative)
Bring Em Back Alive
Audioslave, Audioslave (Alternative)
I Am the Highway
Audioslave, Audioslave (Alternative)
Shadow on the Sun
Audioslave, Audioslave (Alternative)
Show Me How to Live
Audioslave, Audioslave (Alternative)
The Last Remaining Light
Audioslave, Audioslave (Alternative)
What You Are
Audioslave, Audioslave (Alternative)
Faceless Man

☑ Conventional
☑ Direct Delivery (Database)
☑ Interview (Database)
☑ Indirect Delivery (Rec. Engine)
☑ Interview (Rec. Engine)

Figure 6.6: CORESONG's graphical user interface.

DBDs presented previously, these four are sufficient. The Interpreter is implemented as a simple but effective domain-specific cascade parser that identifies words and phrases in passes.

The Interpreter also includes a basic contextual interpretation and anaphora resolution. As argued by Jönsson [1993], a plan-based approach that models goals and intentions is overkill for the kind of system dealt with here. Thus, the anaphora resolution relies on a very simple mechanism in CORESONG. A stack of entities and attributes in focus (the attentional state) is maintained by the Interpreter and is used when mapping from pronouns ("it", "they", "he", "his", etc.) to entities. This approach obviously has limits, but works for local focus management[12] in the systems

---

[12] Global focus shifts are not addressed herein, even though Johansson [2003a] suggests that multimodal interaction by pointing at previously introduced entities on-screen is a frequently used method for performing global focus shifts.

reported on here. The output of the Interpreter thus contains explicit references to domain entity types and values in the outgoing PCQL statements.

The DBD strata machine's Input Streamer feeds the incoming PCQL statement to each of the DBD instances. Each DBD thus processes the incoming token stream independently.

6.4.2 Output: Weaving and Generation

Since each DBD instance operates without modeling its siblings in the DBD strata machine, the complete output from the DBDs need to be (a) integrated, (b) reduced, and (c) surface realized from its PCQL form, so that the system utterance makes sense to the user. (a) and (b) are done in the Output Weaver, whereas (c) is performed by a **template-based** surface realizer. A template-based natural language generating system map its non-linguistic input (PCQL) with no intermediate representations to the linguistic surface structure (cf. [Reiter and Dale, 1997, pages 83–84]). Even though a more advanced generation system might be desirable, the template-based approach coupled with PCQL is sufficient for our conversational recommender systems. Indeed, as argued by van Deemter *et al.* [2005], the template-based approach is not necessarily inferior to standard (or "real") generation systems, and might thus be a suitable choice even for future expansions of the approach put forward here.

The Output Weaver in CoreSong utilizes a set of heuristics in order to address (a) and (b). The solution is pragmatic and divided into two constructs in the Output Weaver: *behavior priority* and an *order heuristic*.

Behavior Priority

We index DBDs with a priority and order the `out` PCQL statements accordingly (ascending order). The *request* with highest priority will be chosen. This hinders the occurrence of two requests back to the user which obviously could be confusing. The order of DBDs are (lowest to highest priority): Conventional, Direct Delivery, Indirect Delivery, and Interview. DBD instances connected to the recommender engine have higher priority than those of the music database. Note that interview and delivery behaviors of the same external resource are naturally designed to be mutually exclusive. The complete order of DBD instances is shown in Figure 6.5.

Order Heuristic

Table 6.8 shows an example dialogue from CoreSong where the Output Weaver has applied the heuristics.

Table 6.8: Sample BCORN dialogue in the music domain. The DBD column shows the DBD instance that won output priority. The out column contains the node that produced the output. Abbreviations used: S = system, U = user, DB = database, REC = recommender, DD = direct delivery, ID = direct delivery, I = interview.

	Dialogue	**DBD**	**out**	**PCQL**
U1	I like the artist *Pearl Jam*.			INFORM $\llbracket \oplus \ (\mathsf{Artist} = \text{'}\mathsf{Pearl\ Jam}\text{'}) \rrbracket$
S1a	These are the albums made by *Pearl Jam*: $[a_1, a_2, ..., a_n]$	DB-DD	25	RESULT $\llbracket \pi \ (\mathsf{Album}), (\mathsf{Artist} = \text{'}\mathsf{Pearl\ Jam}\text{'}) \rrbracket$ VALUES $\llbracket \{a_1, a_2, ..., a_n\} \rrbracket$
S1b	I need more preferences in order to give you recommendations.	REC-I	58	EXCESS $\llbracket \mathsf{Artist} = \text{'}\mathsf{Pearl\ Jam}\text{'} \rrbracket$
S1c	Please provide a genre you like.	REC-I	60	CONSTRAIN $\llbracket \oplus \ \mathsf{Genre} \rrbracket$
U2	What albums have the genre *alternative*?			ASK $\llbracket \pi \ \mathsf{Album}, \ (\mathsf{Genre} = \mathsf{Alternative}) \rrbracket$
S2a	These albums belong to the genre *Alternative*: $[a_1, a_2, ..., a_n]$	DB-DD	25	RESULT $\llbracket \pi \ \mathsf{Album}, \ (\mathsf{Genre} = \mathsf{Alternative}) \rrbracket$ VALUES $\llbracket \{a_1, a_2, ..., a_n\} \rrbracket$
S2b	You might like the song *Once* on the album *Ten* by the artist *Pearl Jam* in the genre *Alternative*.	REC-ID	36	MOTIVATE $\llbracket (\mathsf{Song} = \text{'}\mathsf{Once}\text{'}), (\mathsf{Album} = \text{'}\mathsf{Ten}\text{'}), \ (\mathsf{Artist} = \text{'}\mathsf{Pearl\ Jam}\text{'}), (\mathsf{Genre} = \mathsf{Alternative}) \rrbracket$
S2c	Have you heard it?	REC-ID	33	RECOMMEND $\llbracket \pi \ \mathsf{YN}, (\mathsf{Song} = \text{'}\mathsf{Once}\text{'}) \rrbracket$

Continued on next page

Table 6.8 – continued from previous page

	Dialogue	DBD	out	PCQL
U3	List all *albums* by *The Hives* released before *1985*.			ASK $[\![\ \pi\ \mathsf{Album},\ (\mathsf{Artist} = \text{'The Hives'}),\ (\mathsf{Year} < 1985)\ ]\!]$
S3a	There are no *albums* that match.	DB-I	55	EMPTY $[\![\ \pi\ \mathsf{Album},\ (\mathsf{Artist} = \text{'The Hives'}),\ (\mathsf{Year} < 1985)\ ]\!]$
S3b	Do you want to relax the *year* constraint?	DB-I	57	CONSTRAIN $[\![\ \pi\ \mathsf{YN},\ \mathsf{Year}\ ]\!]$
U4	Yes			INFORM $[\![\ \mathsf{YN} = \mathsf{Yes}\ ]\!]$
S4a	These are the *albums* released by *The Hives*: $[a_1, a_2, ..., a_n]$	DB-DD	25	RESULT $[\![\ \pi\ \mathsf{Album},\ (\mathsf{Artist} = \text{'The Hives'})\ ]\!]$ VALUES $[\![\ \{a_1, a_2, ..., a_n\}\ ]\!]$
S4b	You might like the song *Escape* on the album *The Reason* by the artist *Hoobastank* in the genre *Alternative.*	REC-ID	36	MOTIVATE $[\![\ (\mathsf{Song} = \text{'Escape'}),\ (\mathsf{Album} = \text{'The Reason'}),\ (\mathsf{Artist} = \mathsf{Hoobastank}),\ (\mathsf{Genre} = \mathsf{Alternative})\ ]\!]$
S4c	Have you heard it?	REC-ID	33	RECOMMEND $[\![\ \pi\ \mathsf{YN},\ (\mathsf{Song} = \text{'Escape'})\ ]\!]$
U5	Yes			INFORM $[\![\ \mathsf{YN} = \mathsf{Yes}\ ]\!]$
S5a	What do you think about it?	REC-ID	37	ASKRATE $[\![\ \circledcirc\ (\mathsf{Song} = \text{'Escape'})\ ]\!]$

Due to the behavior priority, there is only one request action available each turn. The order heuristic places this request at the end of the output, so that informing PCQL action statements are guaranteed to precede the request. This guarantees that the constrain request S1c (node 60) in Table 6.8 always occur after the direct delivery S1a (node 25) even though their statements origin from different DBD instances.

Surface Realization

Surface realization (issue (c) above) is handled by the Generator. It is responsible for converting the "weaved" PCQL statements into natural language and can be arbitrarily advanced[13].

The Generator in CoreSong uses a template-based approach [Jurafsky and Martin, 2000], and consists of a template collection, with one template for each `out` command in each instance. Slots in the templates are filled in by querying the appropriate external resource(s).

For example, consider the MOTIVATE template, which corresponds to node 34 in Figure 6.4. In the preceding node (33), the external resource Motivate is responsible for binding a song recommendation from the recommender engine to the variable e, as well as a list m of attributes to be used in the motivation on the form $\{e_{genre}, e_{artist}, e_{album}\}$ from the active preflet (see Section 5.3.2). The template in the Generator for node 34 has the following form:

> `You might like the song` $< e >$ `on the album` $< e_{album} >$ `by the artist` $< e_{artist} >$ `in the genre` $< e_{genre} >$`.`

An example of a filled out template can be found as utterances S2b and S4b in Table 6.8. This template could be even more dynamic considering that the prepositional phrases could be re-arranged so that the attribute with the highest preference strength attached to it would be placed first. This could for example render the template as follows for users where the artist preference in question has a higher preference strength than the album and genre:

> `You might like the song` $< e >$ `by the artist` $< e_{artist} >$ `on the album` $< e_{album} >$ `in the genre` $< e_{genre} >$`.`

Phrases corresponding to attributes for which there is no preference recorded could be removed completely. However, this should probably be done with care, since introducing previously "unknown" attributes (from the user model perspective) could work as conversational impetus (see Section 4.4) and trigger new preference elicitations.

---

[13] Note that content selection step—traditionally part of a natural language generator (cf. [Dale *et al.*, 1998; Jurafsky and Martin, 2000])—is managed by the heuristics of the Output Weaver, whereas subsequent generation steps (lexical selection, sentence structure etc.) are left to the generator component(s).

Simpler templates (typically those connected to PCQL statements with empty FP state formulas) requires no slots to be filled and can be used “as is” (e.g. all nodes in the Conventional DBD, etc.).

6.4.3 Preflet Updates

A normalized preference score for each song in the current preflet (e.g. with the descriptor Situation = Work) is calculated after each turn by the score evaluation formula (see Section 5.4.3), which multiplies detected preference strengths and closeness-of-match with the entity type weights. Recommendations can then be made by selecting songs with the highest preference scores above a pre-defined threshold (e.g. 0.50), and realized in the on-going dialogue (such as utterance S8a/S8b in Figure 6.7). Consider the following preflet based on parts of the dialogue in Figure 6.7:

$$\begin{array}{lcl} \mathsf{User} & : & \mathsf{U} \\ \mathsf{Situation} & : & \mathsf{Work} \\ \mathsf{Genre}_{(1.0)} & \rightarrow & \mathsf{Rock}_{(5)}, \mathsf{Alternative}_{(3)} \\ \mathsf{Artist}_{(2.0)} & \rightarrow & \mathsf{Creed}_{(6)} \end{array}$$

This structure contains the entity type name Genre with strength-annotated values for Rock and Alternative. The strength for Rock is 5 since it is interpreted as being part of the initial problem description (Volunteered-Background), and the strength for Alternative is 3 since it is an indirect preference statement in the form of a question. Similarly, it contains the type Artist with a value Creed with strength 6 due to the superlative preference statement in U4 in Figure 6.7. Furthermore, the importance factor for Artist is raised due to the (meta-)preference in U5. These preferences are valid for user U for play lists in the situation Work.

6.5 Evaluation

To validate the behavior based approach to dialogue management an “overhearer” experiment [Whittaker and Walker, 2004] was conducted by using four different behavior configurations of the CoreSong system. The reason for using the overhearer model is to avoid natural language interpretation problems (since the coverage of grammar

U1 I want a play list for work
S1a I need more preferences in order to give you recommendations.
S1b Is there a particular genre you prefer in that situation?
U2 I like *rock*
S2a These artists belong to the genre *Rock*: `[list]`
S2b I need more preferences in order to give you recommendations.
S2c Any of these you like in particular?
U3 What *alternative* artists are there?
S3a These artists belong to the genre *Alternative*: `[list]`
S3b Is there a particular artist you like?
U4 *Creed* is the best
S4a These albums belong to the artist *Creed*: `[list]`
S4b I need more preferences in order to give you recommendations.
S4c Are there any other genres, artists or albums you prefer at work?
U5 Artist is more important than genre
S5a I need more preferences in order to give you recommendations.
S5b Are there any other genres, artists or albums you prefer at work?
U6 What albums are there by *Audioslave*?
S6a These albums are in the artist *Audioslave.* `[list]`
S6b You might like the song *Original Fire* on the album *Revelations* by the artist *Audioslave* in the genre *Alternative.*
S6c Have you heard it?
U7 No

Figure 6.7: Sample dialogue excerpt from the CoreSong system with factual statements and queries; and descriptive, comparative and superlative preferences. `[list]` denotes a listing of items from an external resource. S = system, U = user.

and lexicon is not our focus), and letting participating users' personal music preferences, that may not be covered by our recommender engine and database, affect the participants' experience of dialogue interaction. Furthermore, it allows us to compare alternative dialogue strategies in the same dialogue context (see Section 2.4.3).

6.5.1 Participants

The experiment was conducted with 30 participants (20 male and 10 female), recruited via student email lists of the Cognitive Science program. The participants' age varied between 20 and 35, and none of the previously participated in neither Study I nor Study II.

6.5.2 CoreSong Configurations

For each of the two external resources (database and recommender engine) used by CoreSong, three DBD instances are implemented: one interview, one direct delivery, and one indirect delivery. Four different DBD instance configurations were used to generate the test dialogues, as shown in Table 6.9.

The first configuration works as a traditional information-retrieval system, where one direct delivery and one interview DBD instance connected to the database are activated. As no recommendations or explicit preference modeling is carried out, this configuration is called the "question-and-answer" configuration (Q-A).

In the second configuration we introduce the recommender engine, and connect one direct delivery and one interview DBD to it. This is done "on top of" the Q-A configuration. Note that we have two external resources, with identical dialogue behaviors attached. We call this the "blunt" configuration, since recommendations are given without motivations and follow-up questions.

Third, the database resource is dropped, and the indirect delivery is introduced along with an interview. These two instances are connected to the recommender engine. Since this configuration is expected to result in many preference questions from the system's part, but no responses to user queries, we call this the "prying" configuration.

The fourth configuration is the "default" CoreSong configuration. This is where we connect the indirect delivery behavior to the recommender engine, and the direct

Table 6.9: Experiment configurations. DD = Direct Delivery, IW = Interview, ID = Indirect Delivery, Db = Database, Rec = Recommender Engine.

Configuration	DD(Db)	IW(Db)	DD(Rec)	ID(Rec)	IW(Rec)
Q-A	x	x			
BLUNT	x	x	x		x
PRYING				x	x
RECOMMENDER	x	x		x	x

delivery to the database, and one interview DBD instance to each external resource. This configuration is called the "recommender" configuration, since this is the role that CoreSong's DBD instances were originally designed to play. Figure 6.5 shows the strata machine layout for the RECOMMENDER configuration.

For each DBD instance, the set of templates for the Generator was designed. In particular, the direct delivery and interview DBDs require customization depending on the external resource connected. For instance, this means that node 25 in the direct delivery DBD (Figure 6.2) has the following template when it is instantiated with the database external resource:

`These` $< e_x >$ `belong to the` $< e_y > < v_y >$`.`

The list of matching entities ($\{v_{x_1}, v_{x_2}, \ldots, v_{x_n}\}$) is displayed in area **B** in the graphical user interface (Figure 6.6).

In the BLUNT configuration however, a direct delivery DBD is instantiated with the recommender engine. In this case, node 25's PCQL statement is rendered as the following template, reflecting the "meaning" of a delivery with that particular external resource. (We know that the entity type for recommender engine deliveries always is Song.)

`I think you will like the song` $< e >$`.`

Following the direct delivery DBD design, no motivation is given.

The different configurations effectively modify CoreSong's complete dialogue strategy. The Q-A configuration, for example, with only the database resource, results in a question-answer information dialogue system without recommendation capabilities, whereas the PRYING configuration supports a preference interview with no

U1 What genres are there?
S2a I found these genres in the database: `[list]`.
S2b What else do you want to know?
U2 I like the genre *Rock & Roll*
S3a These are the artists that belong to the genre *Rock & Roll*: `[list]`.
S3b What else do you want to know?
U3 What songs were made by *Elvis Presley*?
S4a I found these songs by the artist *Elvis Presley* in the database: `[list]`.
S4b What else do you want to know?
U4 I like the album *Live at Madison Square Garden*
S5a These are the songs that belong to the album *Live at Madison Square Garden*: `[list]`.
S5b You might like the song *Suspicious Minds.*
S5c What else do you like?

Figure 6.8: Sample dialogue for the BLUNT experiment configuration. `[list]` denotes lists of genres, artists, albums or songs. S = system, U = user.

power to deliver answers to factual requests. The BLUNT configuration has the power to deliver both database results and recommendations; but the recommendations are not delivered with motivations and follow-up questions as the indirect delivery (found in the RECOMMENDER configuration) is designed to do. Consider the sample dialogues in Figures 6.8 and 6.9, that correspond to the BLUNT and RECOMMENDER configurations respectively. Contrast this to a conversation where there is no recommendation, as in the Q-A configuration (Figure 6.10). To illustrate the difference in behavior we use the same user utterances as in Figure 6.8.

For each of the four CoreSong configurations, a dialogue was generated and put on a web page (see Figure 6.11). The guideline for generating the dialogues was to have a similar sequence of user utterances in all four cases, but yet let the "user" be somewhat cooperative. That is, if CoreSong asks a direct question in one configuration, the "user" in the generated dialogue will answer, even if the same question would not arise in another configuration. See the dialogue excerpts in Figures 6.8, 6.9, and 6.10.

U1 What genres are there?
S2a I found these genres in the database: `[list]`.
S2b I'd like to know more about your preferences.
S2c What else do you like?
U2 I like the genre *Rock & Roll*
S3a These are the artists that belong to the genre *Rock & Roll*: `[list]`.
S3b I'd like to know more about your preferences.
S3c What else do you like?
U3 What songs were made by *Elvis Presley*?
S4a I found these songs by the artist *Elvis Presley* in the database: `[list]`.
S4b I'd like to know more about your preferences.
S4c What else do you like?
U4 I like the album *Live at Madison Square Garden*
S5a You might like the song *Suspicious Minds* because it is a *Rock & Roll* song by *Elvis Presley.*
S5b Have you heard it?
U5 Yes
S6a Ok. What do you think about it?
U6 It's good

Figure 6.9: Sample dialogue for the RECOMMENDER experiment configuration. `[list]` denotes lists of genres, artists, albums or songs. S = system, U = user.

U1-U4 (*Assuming the same user utterances as in Figure 6.8.*)
S5a These are the songs that belong to the album *Live at Madison Square Garden*: `[list]`.
S5b What else do you want to know?

Figure 6.10: Sample dialogue for the Q-A experiment configuration. `[list]` denotes lists of genres, artists, albums or songs. S = system, U = user.

6.5.3 Procedure

Each participant was presented with the four test dialogues, one at a time, displayed in a web browser. For each of the dialogues they were asked to fill out a questionnaire on a 5-point Likert-scale regarding their agreement with four statements (1 meaning total disagreement, and 5 complete agreement), before moving on to the next dialogue. Figure 6.11 shows an example of a web page used in the experiment with the dialogue, and the Likert-scale questionnaire (the complete page is not shown due to size restrictions). There was also an opportunity to comment on each dialogue in a free text field at the end of each questionnaire. A session took about 10-15 minutes to complete. Participants were not paid.

The statements are intended to determine *informativeness* (information quality), *preference modeling*, *coherence*, and *naturalness* (variation and dynamics) of the dialogue excerpts. For example, the statement: "The system's utterances are easy to understand and provide relevant information" reflects informativeness [Whittaker and Walker, 2004].

6.5.4 Results

The results of the four aspects for the four behavior configurations are shown in Figures 6.12, 6.13, 6.14, and 6.15. In general, the participants considered the Q-A and RECOMMENDER configurations to have the highest information quality (86.2% and 85.5% respectively). This is expected, since they both are equipped with the database direct delivery behavior. The PRYING configuration, lacking in database delivery functionality, received a lesser rating on informativeness. For our current work, the notion of coherence is of high importance, since this quality of the dialogue was thought to be at risk when abandoning a monolithic central dialogue strategy model. It is therefore interesting to note that the coherence measure is high for all four configurations (PRYING has the lowest of 70.3%; followed by BLUNT with 79.3%; RECOMMENDER 84.1%; and Q-A 86.2%). Furthermore, the RECOMMENDER configuration—which represents the highest complexity since it encapsulates two external resources, each with different delivery behaviors—was high-ranking in all four aspects: Information quality was high (85.5%), as well as perceived preference management (80.0%) and naturalness (79.3%) in the dialogue, without losing coherence

(84.1%).

The data for the configurations over the parameters were compared using a one-way analysis of variance (ANOVA)[14]. As expected, preference management was perceived as significantly lower in the Q-A configuration compared to the other three configurations, where preferences indeed were modeled and de facto influenced the dialogue. Information quality was perceived as significantly lower in the PRYING configuration compared to the other three (which all included delivery of database results). PRYING also received significantly lower ratings on coherence compared to the other three configurations. This is most likely due to that factual user queries were only used as indicators of indirect preferences, and were not responded to in the way that configurations with delivery behaviors did. Still, its average rating of coherence is 70.3%, which is relatively high considering that some user utterances (e.g. factual requests) are in fact "ignored"! The RECOMMENDER configuration received a significantly higher rating on naturalness compared to the other three configurations. There was no significant difference between the BLUNT and the PRYING configurations in terms of naturalness and variation. Not surprisingly, the behavior of the Q-A configuration was perceived as rather unnatural (46.9%).

The results show that BCORN's non-centralized approach that views dialogue strategy modeling as an emergent phenomenon is feasible, and encourages future development of the approach. They also imply that the individual DBDs of BCORN are soundly designed, and that natural and coherent recommendation dialogue can be explained in terms of the suggested dialogue behaviors.

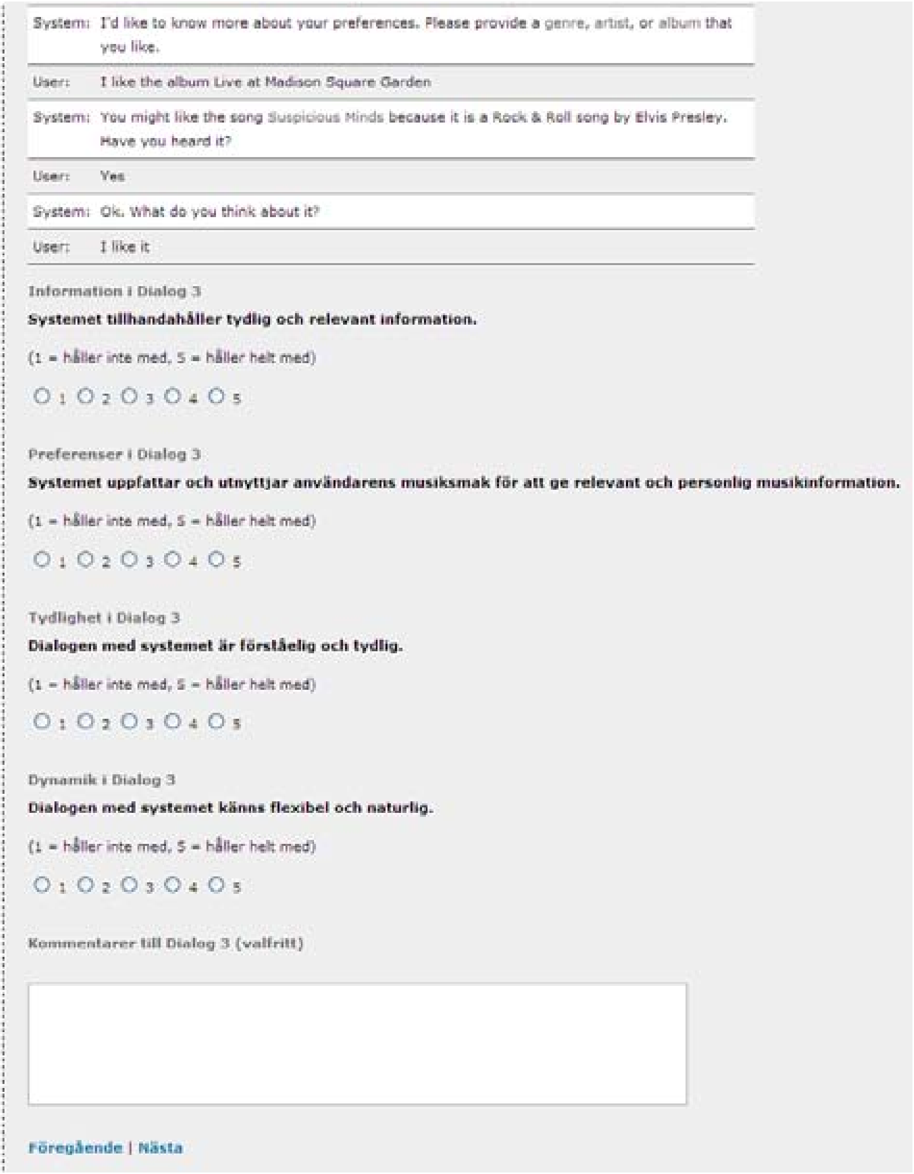
6.6 Summary

The BCORN recommendation dialogue strategy model consists of DBD strata that each encapsulates empirically derived chunks of dialogue behaviors that occur in recommendation dialogue. It uses the PCQL formalism as message-passing format both in and out of the system, and as communication language within a BCORN system. The CoreSong implementation is a functional music conversational recommender system that was evaluated in the overhearer evaluation paradigm. The results (a)

---

[14] $p < 0.001$ n.s. for all differences reported below.

indicates that the behavior-based approach gives rise to coherent and natural recommendation dialogue with high informativeness, and (b) verifies the soundness of the conventional, direct delivery, interview, and indirect delivery DBD designs.



System:	I'd like to know more about your preferences. Please provide a genre, artist, or album that you like.
User:	I like the album Live at Madison Square Garden
System:	You might like the song Suspicious Minds because it is a Rock & Roll song by Elvis Presley. Have you heard it?
User:	Yes
System:	Ok. What do you think about it?
User:	I like it

Information i Dialog 3

Systemet tillhandahåller tydlig och relevant information.

(1 = håller inte med, 5 = håller helt med)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Preferenser i Dialog 3

Systemet uppfattar och utnyttjar användarens musiksmak för att ge relevant och personlig musikinformation.

(1 = håller inte med, 5 = håller helt med)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Tydlighet i Dialog 3

Dialogen med systemet är förståelig och tydlig.

(1 = håller inte med, 5 = håller helt med)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Dynamik i Dialog 3

Dialogen med systemet känns flexibel och naturlig.

(1 = håller inte med, 5 = håller helt med)

○ 1 ○ 2 ○ 3 ○ 4 ○ 5

Kommentarer till Dialog 3 (valfritt)

Figure 6.11: Part of web page for the overhearer evaluation.

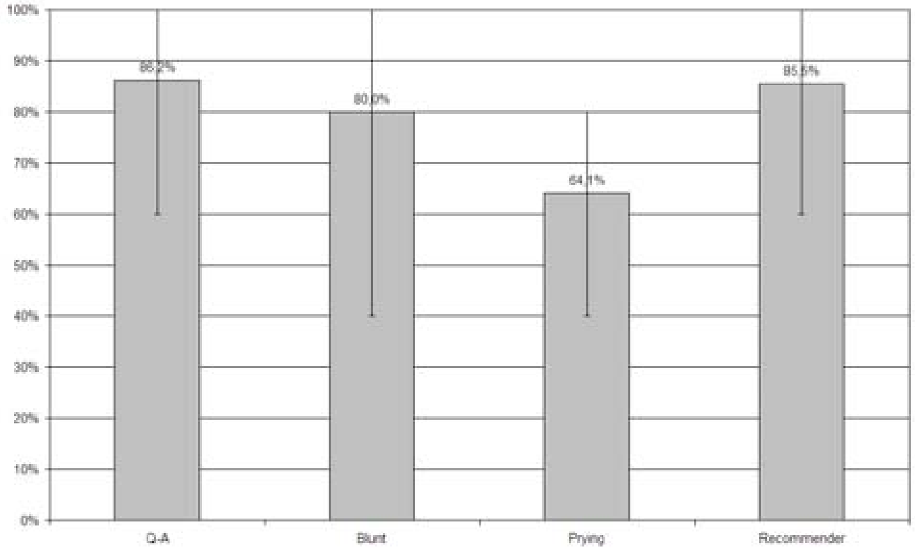


Figure 6.12: Graph of experiment results for the Informativeness aspect. The x axis corresponds to the four behavior configurations presented in Table 6.9.

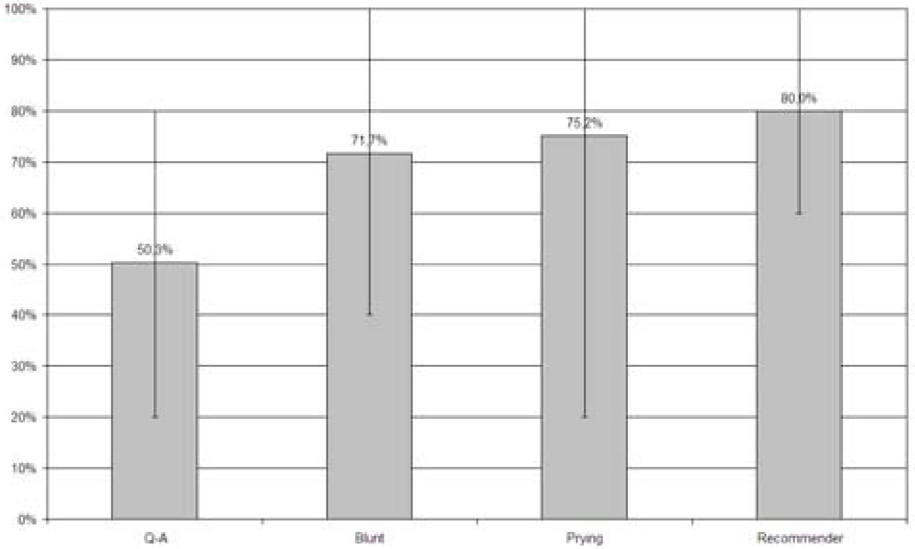


Figure 6.13: Graph of experiment results for the Preference Management aspect. The x axis corresponds to the four behavior configurations presented in Table 6.9.

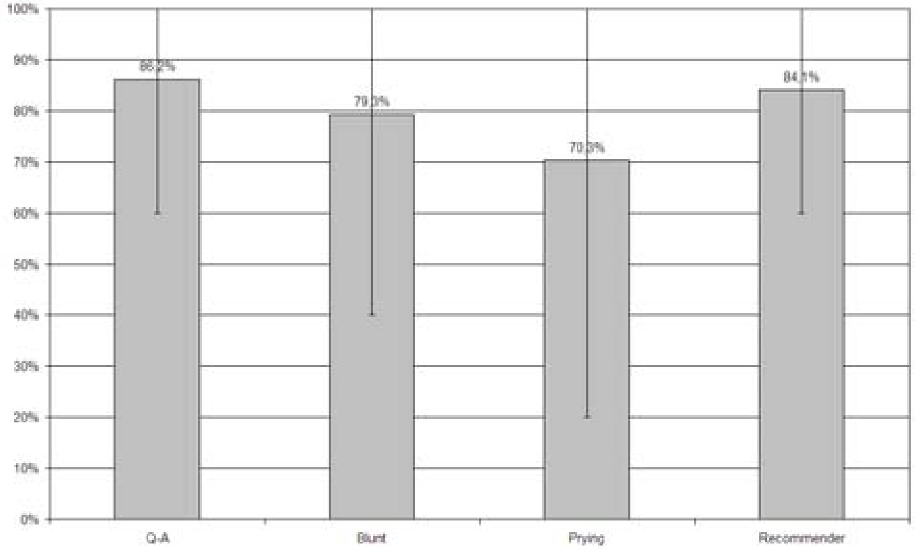


Figure 6.14: Graph of experiment results for the Coherence aspect. The x axis corresponds to the four behavior configurations presented in Table 6.9.

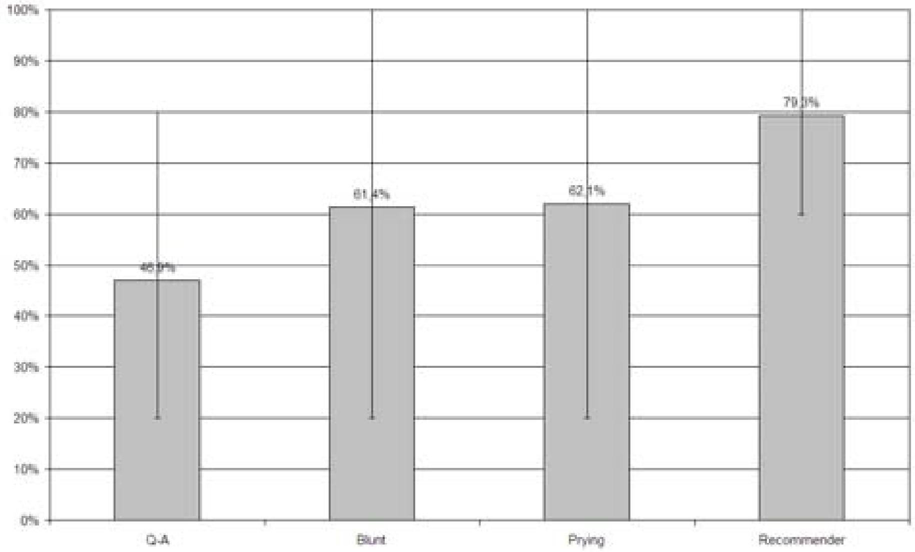


Figure 6.15: Graph of experiment results for Naturalness aspect. The x axis corresponds to the four behavior configurations presented in Table 6.9.

7

Conclusion

We end with a coda, which is both a summary of the previous chapters as well as a discussion around the implications and how this research could continue in the future.

DIALOGUE MANAGEMENT FOR conversational recommender systems has been the focus of the previous chapters. In this final chapter, a concluding summary is first provided in Section 7.1, before some indications of areas for future research are given in Section 7.2.

7.1 Summary

This thesis reports on work on recommendation dialogue for dialogue strategy management in conversational recommender dialogue systems. The research field resides in the intersection of natural language interaction, and personalization for recommender systems, and the work is motivated from an interaction perspective. For example, natural language dialogue can allow users to express their preferences qualitatively, and in contexts where they are motivated to elicit them. Furthermore, detecting a user's preferences and using them for recommending items is a collaborative venture, where coherent conversation with a dialogue partner seems a natural choice of interaction style. Since users benefit from using a recommendation agent that "thinks" like them, either in terms of attribute weights or decision strategies [Aksoy *et al.*, 2006], we want to discover natural and coherent dialogue strategies, and human-like preference modeling, and then through systematic analysis arrive at a computational model suitable for conversational recommender system dialogue interaction.

We started out by surveying theories on dialogue system design and development; user preference modeling and recommender systems; conversational approaches to recommender systems; and software engineering, interaction design, and evaluation of conversational recommender systems (Chapter 2).

Based on the survey, the first step to further examine recommendation dialogue was to study human dialogue participants in a movie recommendation situation (Chapter 3). By applying the dialogue distilling analysis method, we arrived at design implications for dialogue strategy control in the context of conversational recommender systems. As noted by Larsson *et al.* [2000], the dialogue distilling process itself provides important insights on the properties of human-computer dialogue interaction. The analysis in Chapter 3 provides guidelines for (among other issues) focus management, implications depending on the recommender engine utilized in an application, avoiding recommender bias to maintain objectivity, and how to reach efficiency and effectiveness in the human-computer dialogue.

The corpus is also a source for characterization of typical recommendation dialogue. In particular, we described (a) the roles and attached initiatives, which have an impact on how the dialogue progresses; (b) the relations between information requests and preferential statements; (c) a list of re-occurring dialogue act types in the corpus (deliberately coarse-grained) characterizing the dialogue moves found in the corpus; and (d) a classification of two principal dialogue behaviors called *interview* and *delivery*. The symmetry between preferential interview and factual interview (clarification sub-dialogue) and the properties of *direct* vs. *indirect* delivery was found to be useful when describing recommendation dialogue in the BCORN dialogue model.

The guidelines arrived at from the distillation, the dialogue characterization, and the act set were used as the basis of a domain-dependent statechart representation of movie recommendation dialogue presented in Chapter 4. This model was implemented in the ACORN system, and verified in a user study (Chapter 4). The user study indicated that the model gave rise to effective and efficient dialogue with high user satisfaction. The study also gave implications concerning the value of entertainment in order to encourage *exploration* of the domain. High entertainment value resulted in longer dialogues and more efficient preference elicitation. This is termed *conversational impetus* and consists of presenting related information in order to trigger preference statements from the user, consistent with the theory of Carberry *et al.* [1999] and findings of Swearingen and Sinha [2002]. It was also found that it is important to time explanations of recommendations in the dialogue properly.

Based on the empirical studies I and II as well as design and implementation of conversational recommender system prototypes, a behavior-based dialogue model called BCORN was then presented in Chapter 6. BCORN is based on three constructs: First, BCORN utilizes a user preference modeling framework (*preflets*) that supports and utilizes natural language dialogue, and allows for descriptive, comparative, and superlative preference statements, in various situations (Chapter 5).

Second, BCORN uses the message-passing notation PCQL, which is used for describing preferential and factual statements and requests as well as for supporting management of preferences in conversational recommender systems (Chapter 5). PCQL action statements consist of an action tag and a factual and preferential state formula (the FP state), and uses a set of factual and preferential operators that cover descriptive, comparative, and superlative preference statements, as well as factual requests and statements over entities in a domain description. It is obvious that the PCQL notation covers more than what is required by e.g. the CoreSong conversational recommender system. Indeed, there might also be other, more traditional, formalisms in which the needed preferential and factual statements could be expressed. However, a contribution of this work has been to provide the conceptual framework for a generic notation of preferential and factual expressions, closely connected to the theory of Carberry *et al.* [1999], and directly useful in the specific context of conversational recommender system dialogue. PCQL should thus be seen as work toward generics, that will continue beyond this thesis.

Third, we introduced the compact and precise *dialogue behavior diagram* (DBD) notation that is used to describe BCORN's generic recommendation dialogue strategy with conventional, information-providing, and recommendation capabilities. Each DBD describes a natural chunk of a recommender agent's dialogue strategy, and DBD instances are connected to required external resources (Chapter 6). Each DBD in BCORN is based on the classification of recommendation dialogue in terms of interview and delivery.

The DBD notation bears strong resemblance to UML activity diagrams, and therefore to state diagrams (equivalent to Harel statecharts). DBDs can be viewed as a more restricted form of statecharts, with commands that directly support a three-entity interaction model found to work effectively in natural language interface design [Ibrahim and Johansson, 2002b]; and a notation that is tailored to fit recommendation dialogue descriptions for conversational recommender systems.

In the CORESONG implementation of BCORN, the DBDs are run in parallel to give rise to coherent, flexible, and effective dialogue in conversational recommender systems, as indicated by the overhearer evaluation (Study III). The evaluation consisted of generating music recommendation dialogues from four different configurations of CORESONG and let users rate the dialogues on Likert-scales regarding the four parameters *information quality*, *preference management*, *coherence*, and *naturalness* for each configuration. CORESONG was configured by switching DBD instances on or off, and varying the connections between external resources and DBDs. The evaluation provided significant differences between the parameters and the configurations. These results are very promising for the approach for recommendation dialogue management, and indicates that coherent and natural recommendation dialogue of high information quality can be achieved using the behavior-based BCORN model.

We will end this concluding summary with a quotation by Bill Buxton: "notation is a tool of thought" [Buxton, 2007, page 33]. It is important to remember that architecture descriptions typically lean toward informal verbal descriptions and/or flowchart notation, with rather abstract components. They strongly support their own conceptual design philosophies and design methodologies (sometimes without explicitly stating them) [Bryson, 2001]. A formalism bears the promise of a separation between *form* and *content*. That is, a formalism is in some sense free of content and should therefore be used to formalize theory. However, in practice very few "formalisms" are actually completely free from content. On the contrary, many formalisms seem to imply theoretical stand-points. In addition, many formalisms have been proved to be *equivalent*—at least in a mathematical sense. However, this equivalence does not take into account the human designer/developer that carries out the creative work. A formalism encourages designers to think in terms of the formalism (and the associated theoretical chunks that come with it). In order to maintain a creative and exploratory view on how to design and build conversational recommender systems (indeed, any interactive system), we need to maintain a plethora of architectures, models, formalisms, and notations. Choosing the right one to describe and solve the problem at hand is a crucial skill for designers and developers. For example, finite-state machines imply states and transitions; effectively encouraging designers and developers to view problems in terms of such states and transitions, whereas a neural network approach will, on the other hand, support views related to the biological neuron [Pfeiffer and Scheier, 1999]. Nevertheless, they could both express the same functionality—or *behavior*—in a system. Thus, BCORN encourages a view of human-machine recommendation dialogue as a layer-based approach organized in dialogue behavior diagrams centered around preferential and factual statements and requests expressed modeled in PCQL action statements. As shown in this dissertation, the approach has been a successful one, and the involved notations have proved to be a tool for both understanding and expressing the empirically derived qualities of recommendation dialogue for conversational recommender systems.

7.2 Future Work

There are several directions in which this research could continue. Some of them oriented toward Linguistics research (e.g. focusing on dialogue modeling), and some oriented toward Computer Science (e.g. focusing on development of the formal aspects introduced herein, or on tools and platform development). There are also other types of implications, e.g. HCI and interaction design issues, that need to go hand in hand with research and development in the cross-disciplinary approach advocated in this thesis. This section notes some interesting ventures for future research.

Extending the behavior-based approach

One of the most obvious directions, from a dialogue strategy perspective, is to verify the behavior-based model for other types of dialogues. This work has addressed dialogue structure with shorter dialogue-task distance than traditional information retrieval dialogue [Dahlbäck, 1997]. Working toward even shorter distance, such as (general and dynamic) planning dialogue (cf. [Allen *et al.*, 2001]), is an interesting way of exploring the limitations of the proposed approach.

More advanced forms of recommendation dialogue is also an interesting research area. A recommendation dialogue enhancement could be to add dialogue behavior(s) about conflicting preferences. Such features of recommendation dialogue did not occur in the studies reported on in this thesis, and could require long-term studies over several dialogue sessions. Such studies could have implications on extensions of the preflet construct as well as on PCQL. As (negotiative) recommendation dialogue thus becomes more complex, it is likely that argumentation needs to be addressed in more detail (cf. [Larsson, 2002]). Exploring the limitations of the behavior-based approach, as well as studying the amount of work needed to maintain and design dialogue systems in this approach are interesting research issues.

Turn-taking

Whereas this thesis has focused on recommendation dialogue management with strict turn-taking, an important aspect of “natural” (spoken) dialogue is the dynamics of turn-taking and continuous feedback. Since the layered approach has been used to address such factors successfully [Thórisson, 1997; Lemon *et al.*, 2003], it seems a promising direction for the work presented in this thesis as well. Suitable sources for working with turn-taking in similar notations as the DBDs presented in this thesis is the work by Lager and Kronlid [2004]. The empirical material collected in Study I could serve as a base for developing dynamic turn-taking in recommendation dialogue.

Adaptive generation

As shown in this thesis, a promising way to encourage preference eliciting is to employ a dialogue strategy with entertaining dialogues that encourage domain exploration but still allows for efficient handling of user tasks. It is likely that this is a coupling between support for our notion of conversational impetus and “good” natural language generation and prompt design. What constitutes “good” phrasing for a specific system and use context is obviously a challenging problem. Dale *et al.* state in the context of objects in large information repositories: “providing systems with the required communicative sophistication requires the addition of natural language generation technologies” [Dale *et al.*, 1998, page 111]. User-adaptive generation considerations in conversational recommender systems should thus be an important integration aspect to study in detail in the future. BCORN could, from an engineering perspective, be an interesting platform for such work, since the emergent nature of output weaving brings the dialogue strategy very close to the generation problem. Moving from the current template-based approach to a more generic and scalable solution employing state-of-the-art natural language generation techniques [Jurafsky and Martin, 2000], such as adaptive generation based on user models [Walker *et al.*, 2004], would be a natural next step.

Related to this is the generation aspect of explanations of recommendations. In BCORN this is catered for in the MOTIVATE tag and associated templates. User-adaptive explanations of recommendations is a very important issue for the user experience, which has been voiced by other authors (e.g. [Höök, 2000; Herlocker *et al.*, 2000; Buczak *et al.*, 2002]), and highlighted in Study II presented in this thesis (Chapter 4). Explanations are dependent on the user’s preferences, as well as the chosen recommender engine. That is, a collaborative filtering engine reaches its recommendations in a different way than e.g. a content-based engine. This needs to be handled when generating the explanation. Working toward generic solutions for generation of motivations of user-dependent recommendations for recommender engine architectures is thus another important point for future research.

Interpretation

This thesis focuses on dialogue management and the systems reported on herein uses rudimentary interpretation modules. However, it lies in the nature of personalized preferential dialogue to allow users to express e.g. preferences in a free and natural way. This requires robust interpretation mechanisms (e.g. [Jönsson and Strömbäck, 1998]). Investigating the possibilities of constructing robust and generic grammars and lexicons for recommendation dialogue, and take stock on the work effort to customize these for different application domains is an interesting area of research.

Tools and platforms

BCORN and its components have been designed for simplicity to promote rapid development of conversational recommender system applications. However, much more can be done in terms of work toward open development tools and simulation environments for dialogue system development, (cf. [Cunningham, 2000; Degerstedt and Jönsson, 2004; 2006]). An important part of Language Technology research is to support and ease sharing of results and development environments to other researchers and industry. One key to this is to work toward standards in the community to ease "readability" of software engineering artifacts. This includes, for instance, adapting BCORN and the DBD notation to the W3C statechart standard SCXML[1].

---

[1] `http://www.w3.org/TR/scxml/`

Bibliography

[Aksoy *et al.*, 2006] Lerzan Aksoy, Paul N. Bloom, Nicholas H. Lurie, and Bruce Cooil. Should recommendation agents think like people? *Journal of Service Research*, 8(4):297–315, May 2006.

[Allen *et al.*, 2000] James Allen, Donna Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. An Architecture for a Generic Dialogue Shell. *Natural Language Engineering*, 6(3-4):213–228, September 2000.

[Allen *et al.*, 2001] James F. Allen, Donna K. Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. Towards conversational human-computer interaction. *AI Magazine*, 22(4):27–37, 2001.

[Baudisch and Brueckner, 2002] P. Baudisch and L. Brueckner. TV Scout: Guiding users from printed TV program guides to personalized TV recommendation. In *Proceedings of the 2nd Workshop on Personalization in Future TV*, Malaga, Spain, 2002.

[Beck, 2000] Kent Beck. *Extreme Programming Explained.* Addison-Wesley, 2000.

[Berglund and Johansson, 2004] Aseel Berglund and Pontus Johansson. Using Speech and Dialogue for Interactive TV Navigation. *Universal Access in the Information Society*, 3(3–4):224–238, October 2004.

[Billsus and Pazzani, 2000] D Billsus and M J Pazzani. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10:147–180, 2000.

[Bridge, 2002] Derek Bridge. Towards Conversational Recommender Systems: A Dialogue Grammar Approach. In D.W.Aha, editor, *Proceedings of the Workshop in*

Mixed-Initiative Case-Based Reasoning, Workshop Programme at the Sixth European Conference in Case-Based Reasoning, pages 9–22, 2002.

[Brooks, 1991a] Rodney A. Brooks. Intelligence without reason. In *International Joint Conference on Artificial Intelligence'91*, pages 569–595, 1991.

[Brooks, 1991b] Rodney A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Brooks, 1997] Rodney A. Brooks. From earwigs to humans. practice and future of autonomous agents. *Robotics and Autonomous Systems*, 20:291–304, 1997.

[Bryson, 2001] Joanna J. Bryson. *Intelligence by Design: Principles of Modularity and Coordination for Engineering Complex Adaptive Agents.* PhD thesis, MIT, Department of EECS, Cambridge, MA, June 2001.

[Buczak *et al.*, 2002] Anna Buczak, John Zimmerman, and Kaushal Kurapati. Personalization: Improving Ease-of-Use, Trust and Accuracy of a TV Show Recommender. In *Proceedings of the 2nd Workshop on Personalization in Future TV*, Malaga, Spain, 2002.

[Bunt, 1994] Harry Bunt. Context and Dialogue Control. *Think*, 3:19–31, 1994.

[Burke *et al.*, 1997] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[Burke, 2002] Robin D. Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.

[Buxton, 2007] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design.* Morgan Kaufmann, 2007.

[Carberry *et al.*, 1999] Sandra Carberry, Jennifer Chu-Carroll, and Stephanie Elzer. Constructing and Utilizing a Model of User Preferences in Collaborative Consultation Dialogues. *Computational Intelligence*, 15(3):185–217, 1999.

[Carenini *et al.*, 2003] Giuseppe Carenini, Jocelyin Smith, and David Poole. Towards More Conversational and Collaborative Recommender Systems. In *Proceedings of* *the International Conference of Intelligent User Interfaces*, pages 12–18, Miami, Florida, USA, 2003.

[Chin, 2001] David Chin. Empirical evaluation of user models and user-adapted systems. *User Modeling and User-Adapted Interaction*, 11(1–2):181–194, 2001.

[Chu-Carroll, 2000] Jennifer Chu-Carroll. Mimic: An adaptive mixed initiative spoken dialogue system for information queries. In *Proceedings of 6th Applied Natural Language Processing Conference*, pages 97–104, 2000.

[Clark, 1996] Herbert H. Clark. *Using Language.* Cambridge University Press, 1996.

[Cunningham, 2000] Hamish Cunningham. *Software Architecture for Language Engineering.* PhD thesis, University of Sheffield, 2000. `http://gate.ac.uk/sale/thesis/`.

[Dahlbäck *et al.*, 1998] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. Wizard of oz studies – why and how. In Mark Maybury & Wolfgang Wahlster, editor, *Readings in Intelligent User Interfaces.* Morgan Kaufmann, 1998.

[Dahlbäck, 1997] Nils Dahlbäck. Towards a dialogue taxonomy. In Elisabeth Maier, Marion Mast, and Susann LuperFoy, editors, *Dialogue Processing in Spoken Language Systems*, number 1236, pages 29–40. Springer Verlag Series LNAI-Lecture Notes in Artificial Intelligence, 1997.

[Dale and Reiter, 1995] Robert Dale and Ehud Reiter. Computational interpretations of the Gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263, 1995.

[Dale *et al.*, 1998] Robert Dale, Jon Oberlander, Maria Milosavljevic, and Alistair Knott. Integrating natural language generation and hypertext to produce dynamic documents. *Interacting with Computers*, 11:109–135, 1998.

[Degerstedt and Jönsson, 2004] Lars Degerstedt and Arne Jönsson. Open resources for language technology. In *Proceedings of 4th International Conference on Language Resources and Evaluation, LREC 2004, Lisboa, Portugal*, 2004.

[Degerstedt and Jönsson, 2006] Lars Degerstedt and Arne Jönsson. LinTest – a development tool for testing dialogue systems. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech'06)*, 2006.

[Degerstedt and Johansson, 2003] Lars Degerstedt and Pontus Johansson. Evolutionary Development of Phase-Based Dialogue Systems. In *Proceedings of the 8th Scandianvian Conference on Artificial Intelligence*, pages 59–67, Bergen, Norway, November 2003.

[Degerstedt and Jönsson, 2001] Lars Degerstedt and Arne Jönsson. A Method for Systematic Implementation of Dialogue Management. In *Workshop notes from the 2nd IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems, Seattle, WA*, 2001.

[Dybkjær *et al.*, 1997] Laila Dybkjær, Niels Ole Bernsen, and Hans Dybkjær. A Methodology for Diagnostic Evaluation of Spoken Human-Machine Dialogue. *International Journal of Human Computer Studies*, 48:605–625, 1997.

[Dybkjær *et al.*, 1998] L. Dybkjær, N.O. Bernsen, R. Carlson, L. Chase, N. Dahlbäck, K. Failenschmid, U. Heid, P. Heisterkamp, A. Jönsson, H. Kamp, I. Karlsson, J.v. Kuppevelt, L. Lamel, P. Paroubek, and D. Williams. The DISC approach to spoken language systems development and evaluation. In *Proceedings of the First International Conference on Language Resources and Evaluation*, Granada, 1998.

[Fischer, 2001] Gerhard Fischer. User modeling in human-computer interaction. *User Modeling and User-Adapted Interaction*, 11(1/2):65–86, 2001.

[Foster and White, 2005] Mary Ellen Foster and Michael White. Assessing the impact of adaptive generation in the COMIC multimodal dialogue system. In Ingrid Zukerman, Jan Alexandersson, and Arne Jönsson, editors, *Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 24–31, Edinburgh, Scotland U.K., 2005.

[Fowler, 2000] Martin Fowler. *Refactoring: Improving the Design of Existing Code.* Addison-Wesley Object Technology Series, 2000.

[Garlan and Shaw, 1993] David Garlan and Mary Shaw. An Introduction to Software Architecture. *Advances in Software Engineering and Knowledge Engineering, Series on Software Engineering and Knowledge Engineering*, 2:1–39, 1993.

[Göker and Thompson, 2000] Mehmet Göker and Cynthia Thompson. Personalized conversational case-based recommendation. In *Advances in Case-Based Reasoning. Proceedings of the 5th European Workshop on Case-Based Reasoning*, pages 99–111, Trento, Italy, 2000.

[Grice, 1975] Paul H. Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics (vol. 3) Speech Acts*. Academic Press, 1975.

[Gupta and Grover, 2003] Arpita Gupta and P. S. Grover. Proposed evaluation framework for adaptive hypermedia systems. In *Proceedings of the 3rd Workshop on Empirical Evaluation of Adaptive Systems, in conjunction with AH2004*, pages 161–171, Eindhoven, The Netherlands, 2003.

[Gustafson *et al.*, 2000] Joakim Gustafson, Linda Bell, Jonas Beskow, Johan Boye, Rolf Carlson, Jens Edlund, Björn Granström, David House, and Mats Wirén. AdApt – a multimodal conversational dialogue system in an apartment domain. In *Proceedings of 6th International Conference on Spoken Language Processing (ICSLP2000/INTERSPEECH2000), Beijing, China*, 2000.

[Harel, 1987] David Harel. Statecharts: A visual formalism for complex systems. *Sci. Comput. Programming*, 8:231–274, 1987.

[Herlocker *et al.*, 2000] J. Herlocker, J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work*, pages 241–250, December 2000.

[Höök, 2000] Kristina Höök. Steps to take before IUIs become real. *Journal of Interacting with Computers*, 12(4):409–426, February 2000.

[Höök *et al.*, 1996] K. Höök, J. Karlgren, A. Waern, N. Dahlbäck, C. Jansson, K. Karlgren, and B. Lemaire. A Glass Box Approach to Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 6(2–3):157–184, 1996.

[Hulstijn, 1999] Joris Hulstijn. Modelling Usability: development methods for dialogue systems. In J. Alexandersson, editor, *Proceedings of the 1st IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Stockholm, 1999.

[Hulstijn, 2000] Joris Hulstijn. *Dialogue Models for Inquiry and Transaction.* PhD thesis, Universiteit Twente, 2000.

[Ibrahim and Johansson, 2002a] Aseel Ibrahim and Pontus Johansson. Multimodal dialogue systems: a case study for interactive tv. In *Proceedings of the 7th ERCIM Workshop "User Interfaces for All"*, pages 209–218, Paris, France, 2002.

[Ibrahim and Johansson, 2002b] Aseel Ibrahim and Pontus Johansson. Multimodal Dialogue Systems for Interactive TV Applications. In *Proceedings of the 4th IEEE ICMI Conference*, pages 117–222, Pittsburgh (PA), USA, 2002.

[Jacobson *et al.*, 1992] I. Jacobson, M. Christerson, P Jonsson, and G Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach.* ACM Press: Addison-Wesley Publishing, 1992.

[Jönsson and Dahlbäck, 2000] Arne Jönsson and Nils Dahlbäck. Distilling Dialogues - A Method Using Natural Dialogue Corpora for Dialogue Systems Development. In *Proceedings of 6th Applied Natural Language Processing Conference*, pages 44–51, 2000.

[Johansson *et al.*, 2002] Pontus Johansson, Lars Degerstedt, and Arne Jönsson. Iterative Development of an Information-Providing Dialogue System. In *Proceedings of the 7th ERCIM Workshop "User Interfaces for All"*, pages 29–36, Paris, France, 2002.

[Johansson, 2003a] Pontus Johansson. MadFilm - a multimodal approach to handle search and organization in a movie recommendation system. In *Proceedings of the 1st Nordic Symposium on Multimodal Communication*, pages 53–65, Helsingör, Denmark, 2003.

[Johansson, 2003b] Pontus Johansson. Natural Language Interaction in Personalized EPGs. In *Proceedings of the 3rd UM Workshop "Personalization in Future TV"*, pages 27–31, Pittsburgh (PA), USA, 2003.

[Johansson, 2004] Pontus Johansson. Design and Development of Recommender Dialogue Systems. Licentiate Thesis 1079, Linköping Studies in Science and Technology, Linköping University, 2004.

[Jönsson and Dahlbäck, 1988] Arne Jönsson and Nils Dahlbäck. Talking to a Computer is not Like Talking to your Best Friend. In *Proceedings of the First Scandinavian Conference on Artificial Interlligence, Tromsø*, 1988.

[Jönsson and Strömbäck, 1998] Arne Jönsson and Lena Strömbäck. Robust interaction through partial interpretation and dialogue management. In *Proceedings of Coling/ACL'98, Montréal*, 1998.

[Jönsson, 1993] Arne Jönsson. *Dialogue Management for Natural Language Interfaces – An Empirical Approach.* PhD thesis, Linköping University, 1993.

[Jönsson, 1997] Arne Jönsson. A Model for Habitable and Efficient Dialogue Management for Natural Language Interaction. *Natural Language Engineering*, 3(2/3):103–122, 1997.

[Jurafsky and Martin, 2000] Daniel Jurafsky and James H. Martin. *Speech and Language Processing.* Prentice-Hall, 2000.

[Kass and Finin, 1988] Robert Kass and Tim Finin. Modeling the User in Natural Language Systems. *Computational Linguistics*, 14(3):5–22, 1988.

[Kay, 2000] Judy Kay. User Modeling for Adaptation. In C. Stephanidis, editor, *User Interfaces for All, Human Factors Series*, pages 271–294. Lawrence Erlbaum Associates, 2000.

[Kirsh, 1991] David Kirsh. Today the earwig, tomorrow man? *Artificial Intelligence*, 47(1–3):161–184, 1991.

[Kobsa, 1993] Alfred Kobsa. User Modeling: Recent Work, Prospects, and Hazards. In M Schneider-Hufschmidt, T Kuhme, and U Malinowski, editors, *Adaptive User Interfaces: Principles and Practice.* Springer Verlag, Berlin, Amsterdam, North-Holland, 1993.

[Lager and Kronlid, 2004] Torbjörn Lager and Fredrik Kronlid. The CURRENT Platform: Building Conversational Agents in Oz. In P. van Roy, editor, *Proceedings of MOZ 2004*, LNAI 3389, pages 161–174. Springer-Verlag, 2004.

[Langley *et al.*, 1999] Pat Langley, Cynthia Thompson, Renee Elio, and Afsaneh Haddadi. An adaptive conversational interface for destination advice. In *Proceedings of the 3rd International Workshop on Cooperative Information Agents*, Uppsala, Sweden, 1999.

[Larman and Basili, 2003] Craig Larman and Victor Basili. Iterative and incremental development: A brief history. *IEEE Computer*, 36(6):47–56, 2003.

[Larsson and Traum, 2000] Staffan Larsson and David Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering*, 6(3–4):323–340, 2000.

[Larsson *et al.*, 2000] Staffan Larsson, Lena Santamarta, and Arne Jönsson. Using the Process of Distilling Dialogues to Understand Dialogue Systems. In *Proceedings of 6th International Conference on Spoken Language Processing (ICSLP2000/INTERSPEECH2000), Beijing, China*, 2000.

[Larsson, 2002] Staffan Larsson. *Issue-based Dialogue Management.* PhD thesis, Göteborg University, 2002.

[Lemon *et al.*, 2003] Oliver Lemon, Lawrence Cavedon, and Barbara Kelly. Managing dialogue interaction: A multi-layered approach. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 168–177, Sapporo, Japan, July 2003.

[Linden *et al.*, 1997] Greg Linden, Steve Hanks, and Neal Lesh. Interactive Assessment of User Preference Models: The Automated Travel Assistant. In *Proceedings of the 6th Conference on User Modeling*, pages 67–78, 1997.

[Litman and Pan, 1999] Diane J. Litman and Shimei Pan. An empirical evaluation of an adaptable spoken dialogue system. In *Proceedings of the Seventh International Conference on User Modeling*, pages 55–64, 1999.

[Martin, 1999] Robert C. Martin. Iterative and Incremental Development I-III. C++ Report, 1999.

[McGlashan *et al.*, 1992] Scott McGlashan, Norman Fraser, Nigel Gillbert, Eric Bilange, Paul Heisterkamp, and Nick Youd. Dialogue management for telephone information systems. In *Proceedings of the Third Conference on Applied Natural Language Processing, Trento, Italy*, pages 245–246, 1992.

[McTear, 2002] Micheal F. McTear. Spoken Dialogue Technology: Enabling the Conversational User Interface. *ACM Comput. Surv.*, 34(1):90–169, 2002.

[Middleton *et al.*, 2002] Stuart E. Middleton, Harith Alani, Nigel R. Shadbolt, and David C. De Roure. Exploiting synergy between ontologies and recommender systems. In *The Eleventh International World Wide Web Conference (WWW2002)*, Hawaii, USA, 2002. Semantic Web Workshop 2002, WWW2002.

[Montaner *et al.*, 2003] Miquel Montaner, Beatriz López, and Josep Lluís De La Rosa. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review*, 19(4):285–330, June 2003.

[Nielsen, 1993] Jacob Nielsen. *Usability Engineering*. Academic Press, Inc., 1993.

[Norman, 1988] Donald A. Norman. *The Design of Everyday Things*. The MIT Press, 1988.

[O'Neill and McTear, 2000] Ian M. O'Neill and Michael F. McTear. Object-Oriented Modelling of Spoken Language Dialogue Systems. *Natural Language Engineering*, 6, 2000.

[Pakucs, 2003] Botond Pakucs. Towards dynamic multi-domain dialogue processing. In *Proceedings of EuroSpeech'03*, Geneva, Switzerland, September 2003.

[Pang *et al.*, 2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of EMNLP'02*, pages 79–86, 2002.

[Paramythis *et al.*, 2001] A. Paramythis, A. Totter, and C. Stephanidis. A modular approach to the evaluation of adaptive user interfaces. In *Proceedings of Workshop on Empirical Evaluations of Adaptive Systems at UM'01*, pages 9–24, July 2001.

[Pfeiffer and Scheier, 1999] Rolf Pfeiffer and Christian Scheier. *Understanding Intelligence.* MIT Press, 1999.

[Pieraccini and Huerta, 2005] Roberto Pieraccini and Juan Huerta. Where do we go from here? Research and commercial spoken dialog systems. In *Proceedings of the 6th SIGdial Workshop on Discourse and Dialogue*, Lisbon, Portugal, 2005.

[Pohl *et al.*, 1995] Wolfgang Pohl, Alfred Kobsa, and Oliver Kutter. User model acquisition heuristics based on dialogue acts. In *Proceedings of the International Workshop on the Design of Cooperative Systems*, pages 471–486, Antibes-Juan-les-Pins, France, 1995.

[Rashid *et al.*, 2002] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In Yolanda Gil and David B. Leake, editors, *Proceedings of the 2002 International Conference on Intelligent User Interfaces (IUI-02)*, pages 127–134, New York, 2002. ACM Press.

[Reeves and Nass, 1996] B. Reeves and C. Nass. *The Media Equation.* Cambridge: Cambridge University Press, 1996.

[Reiter and Dale, 1997] Ehud Reiter and Robert Dale. Building applied natural-language generation systems. *Natural Language Engineering*, 3(1):57–87, 1997.

[Rich, 1979] Elaine Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979.

[Rich, 1989] Elaine Rich. Stereotypes and user modeling. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, pages 35–51. Springer-Verlag, Berlin, 1989.

[Sparck Jones, 1989] Karen Sparck Jones. Realism about user modeling. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*, pages 341–

363. Springer-Verlag, Symbolic Computation Series, 1989. Also available as Technical Report 111, University of Cambridge Computer Laboratory.

[Steels, 1990] Luc Steels. Towards a theory of emergent functionality. In J-A. Meyer and S. Wilson, editors, *From Animals to Animats (Proceedings of the 1st International Conference on Simulation of Adaptive Behavior)*, pages 451–461. Bradford Books (MIT Press), 1990.

[Steels, 1994] Luc Steels. A case study in the behavior-oriented design of autonomous agents. In Dave Cliff, Philip Husbands, Jean-Arcady Meyer, and Stewart W. Wilson, editors, *From Animals to Animats*, pages 445–452. MIT Press, 1994.

[Swearingen and Sinha, 2002] Kirsten Swearingen and Rashmi Sinha. Interaction Design for Recommender Systems. In *Interactive Systems (DIS2002)*, London, June 2002.

[Thompson *et al.*, 2004] Cynthia Thompson, Mehmet Göker, and Pat Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004.

[Thórisson, 1997] Kristinn R. Thórisson. Layered modular action control for communicative humanoids. In *Computer Animation '97, Geneva, Switzerland*, pages 134–143, 1997.

[Thórisson, 2002] Kristinn R. Thórisson. Natural turn-taking needs no manual: Computational theory and model, from perception to action. In Björn Granström, David House, and Inger Karlsson, editors, *Multimodality in Language and Speech Systems*, pages 173–207. The Netherlands: Kluwer Academic Publishers, Dordrecht, 2002.

[van Deemter *et al.*, 2005] Kees van Deemter, Mariët Theune, and Emiel Krahmer. Real versus template-based natural language generation: A false opposition? *Computational Linguistics*, 31(1):15–24, 2005.

[Viethen and Dale, 2006] Jette Viethen and Robert Dale. Algorithms for generating referring expressions: Do they do what people do? In *Proceedings of the International Conference on Natural Language Generation*, Sydney, Australia, July 2006.

[Walker *et al.*, 1998] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. Paradise: A framework for evaluating spoken dialogue agents. In Mark Maybury & Wolfgang Wahlster, editor, *Readings in Intelligent User Interfaces*. Morgan Kaufmann, 1998.

[Walker *et al.*, 2004] M.A. Walker, S.J. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28:811–840, 2004.

[Weibelzahl and Weber, 2002] Stephan Weibelzahl and Gerhard Weber. Advantages, opportunities, and limits of empirical evaluations: Evaluating adaptive systems. *Künstliche Intelligenz*, 3/02:17–20, 2002.

[Whittaker and Walker, 2004] Steve Whittaker and Marilyn Walker. Evaluating dialogue strategies in multimodal dialogue systems. In W. Minker, D. Bühler, and L. Dybkjaer, editors, *Spoken Multimodal Human-Computer Dialogue in Mobile Environments*, pages 247–268. Kluwer Academic Publishers, 2004.

[Wärnestål *et al.*, 2007a] Pontus Wärnestål, Lars Degerstedt, and Arne Jönsson. Emergent conversational recommendations: A dialogue behavior approach. In *Proceedings of the 8th SIGDIAL Workshop on Discourse and Dialogue*, Antwerp, Belgium, September 2007. SigDial.

[Wärnestål *et al.*, 2007b] Pontus Wärnestål, Lars Degerstedt, and Arne Jönsson. Interview and delivery: Dialogue strategies for conversational recommender systems. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (Nodalida)*, Tartu, Estonia, May 2007.

[Wärnestål *et al.*, 2007c] Pontus Wärnestål, Lars Degerstedt, and Arne Jönsson. PCQL: A formalism for human-like preference dialogues. In *Proceedings of the 5th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, Hyderabad, India, January 2007.

[Wärnestål, 2005] Pontus Wärnestål. User evaluation of a conversational recommender system. In Ingrid Zukerman, Jan Alexandersson, and Arne Jönsson, editors, *Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 32–39, Edinburgh, Scotland U.K., 2005.

[Yankelovich, 1997] Nicole Yankelovich. Using Natural Dialogs as the Basis for Speech Interface Design. In *Automated Spoken Dialog Systems*. MIT Press, 1997.

[Zukerman and Litman, 2001] Ingrid Zukerman and Diane Litman. Natural language processing and user modeling: Synergies and limitations. *User Modeling and User-Adapted Interaction*, 1-2:129–158, 2001.

A

Corpora

Study I and Study II resulted in two corpora, which are presented here.

A.1 Corpus I

Domain: Movie recommendations
Agents: Human-Human
Language: Swedish
Modality: Spoken
Number of dialogues: 24
Total number of utterances: 2684
Total time: 7 hours, 37 minutes
Mean time per dialog: 19 minutes
Mean number of utterances per dialog: 112
Total Preferential statements: 1361 (50.7% of total)
  Descriptive: 878 (64.5% of pref.statements; 32.7% of total)

  Comparative: 242 (17.8% of pref.statements; 9.0% of total)
  Superlative: 242 (17.8% of pref.statements; 9.0% of total)
Total Factual statements: 768 (28.6% of total)

A.2 Corpus II

Domain: Movie recommendations
Agents: Human-Computer
Language: Swedish
Modality: Written
Number of dialogues: 20
Total number of utterances: 452
Mean number of utterances per dialog: 11

B

Experiment Scenarios

Below is a summary of instructions, scenarios, and questionnaire statements and questions for Studies I–III. (Translated from Swedish.)

B.1 Study I

B.1.1 Recommender's Instructions

Introduction In this scenario you play the role of a professional "movie recommender". Your task is to find out your customer's movie preferences in order to create a preference profile. With this profile and the movie information repository Internet Movie Database (www.imdb.com) you should be able to recommend 5 previously unseen movies to your customer that you feel confident that he or she will like. As a professional movie recommender, it is your responsiblity to try to make the interview and recommendation dialogue pleasant and efficient.

Specific tasks and tips

1. Make a list of 5 of the customer's favorite movies, and try to find out what makes them his or her favorites
2. Try to find common characteristics for the favorite movies in terms of e.g. actors, genres, directors, plot elements etc. These characteristics can help you motivate your 5 recommendations.
3. You have a note pad and pencil that you can use to make notes about your customer's preferences and favorite movies.
4. Before the session is complete you should have recommended at least 5 previously unseen movies.
5. When you have completed the session, you end the conversation and both you and your customer will be asked to fill out a questionnaire about your dialogue

B.1.2 Customer's Instructions

In this scenario you play the role of a customer to a professional "movie recommender". Your goal is to get a number of movie recommendations that suit your movie preferences. You and your recommender will engage in a dialogue in order to build a movie preference profile for you. Throughout the dialogue you will receive recommendations. If you have already seen a movie that is recommended to you, you are encouraged to let your recommender know this. When you have completed the session and received at least 5 movie suggestions that you have not seen before, both you and your recommender will be asked to fill out a questionnaire about your dialogue.

B.2 Study II

The scenario presented to the participants consisted of three sub-tasks. They are presented here in an abbreviated form translated from Swedish.

B.2.1 Task 1

Find out if the actor *Brad Pitt* has acted in any comedies. Please mark your answer in the protocol.

B.2.2 Task 2

Find out what actors/actresses are starring in the movie *Entrapment.* Note the name of one of them, and ask ACORN to recommend a drama starring that actor/actress. Please note the actor/actress and recommended title in the protocol.

B.2.3 Task 3

Ask for a comedy starring *Adam Sandler*, and note the recommended title in the protocol. Then find out who has directed the movie, and note his/her name in the protocol.

B.2.4 Questionnaire Statements

1. I could solve the task efficiently
2. It was easy to solve the task
3. It felt natural to solve the task by engaging in a dialogue with ACORN
4. ACORN asked relevant questions that helped me solving the task
5. The recommendations were effective and matched what I had told the system
6. ACORN finds information and recommendations fast
7. The database contains enough movies for me to solve the task
8. The database contains enough information about every movie for me to solve the task
9. ACORN understood my input
10. I knew what I could and could not say during the dialogue

11. ACORN worked as I expected it to
12. ACORN adapted to me
13. It was easy to decide "whose turn" it was to say something
14. ACORN's utterances were easy to understand
15. I rarely needed to ask for help
16. ACORN's instructions were relevant and had enough detail
17. ACORN's graphical user interface is attractive
18. The graphical user interface is important in this kind of system
19. It is entertaining to interact with ACORN
20. It is interesting to use ACORN
21. Overall, I am satisfied with the interaction with ACORN
22. I would consider to use ACORN in the future
23. I think this type of system would be beneficial in other contexts and domains

B.3 Study III

B.3.1 User Instructions

The instructions have been translated from Swedish.

The dialogues you are about to read are transcripts between a human user and a conversational music recommender system. Please read each dialogue and imagine that you are the user. Make an intuitive judgment of your overall impression of the dialouge qualities to the four statements that follow each dialouge on a five-graded scale, where 1 means that you strongly disagree, and 5 means that your strongly agree. You also have the option to provide free text comments for each dialogue if you want to develop your impression.

B.3.2 Questionnaire Statements

The following four statements were posed in the questionnaire. The statements have been translated from Swedish.

Informativeness

The system's utterances are easy to understand and provide relevant information

Preference Management

The system detects and utilizes the user's music preferences in order to give relevant and personalized song recommendations.

Coherence

The dialouge is understandable, unambiguous, and clear.

Naturalness

The dialogue is flexible, dynamic and natural.

Department of Computer and Information Science
Linköpings universitet

Dissertations

Linköping Studies in Science and Technology

No 14 **Anders Haraldsson:** A Program Manipulation System Based on Partial Evaluation, 1977, ISBN 91-7372-144-1.

No 17 **Bengt Magnhagen:** Probability Based Verification of Time Margins in Digital Designs, 1977, ISBN 91-7372-157-3.

No 18 **Mats Cedwall:** Semantisk analys av processbeskrivningar i naturligt språk, 1977, ISBN 91-7372-168-9.

No 22 **Jaak Urmi:** A Machine Independent LISP Compiler and its Implications for Ideal Hardware, 1978, ISBN 91-7372-188-3.

No 33 **Tore Risch:** Compilation of Multiple File Queries in a Meta-Database System 1978, ISBN 91-7372-232-4.

No 51 **Erland Jungert:** Synthesizing Database Structures from a User Oriented Data Model, 1980, ISBN 91-7372-387-8.

No 54 **Sture Hägglund:** Contributions to the Development of Methods and Tools for Interactive Design of Applications Software, 1980, ISBN 91-7372-404-1.

No 55 **Pär Emanuelson:** Performance Enhancement in a Well-Structured Pattern Matcher through Partial Evaluation, 1980, ISBN 91-7372-403-3.

No 58 **Bengt Johnsson, Bertil Andersson:** The Human-Computer Interface in Commercial Systems, 1981, ISBN 91-7372-414-9.

No 69 **H. Jan Komorowski:** A Specification of an Abstract Prolog Machine and its Application to Partial Evaluation, 1981, ISBN 91-7372-479-3.

No 71 **René Reboh:** Knowledge Engineering Techniques and Tools for Expert Systems, 1981, ISBN 91-7372-489-0.

No 77 **Östen Oskarsson:** Mechanisms of Modifiability in large Software Systems, 1982, ISBN 91-7372-527-7.

No 94 **Hans Lunell:** Code Generator Writing Systems, 1983, ISBN 91-7372-652-4.

No 97 **Andrzej Lingas:** Advances in Minimum Weight Triangulation, 1983, ISBN 91-7372-660-5.

No 109 **Peter Fritzson:** Towards a Distributed Programming Environment based on Incremental Compilation,1984, ISBN 91-7372-801-2.

No 111 **Erik Tengvald:** The Design of Expert Planning Systems. An Experimental Operations Planning System for Turning, 1984, ISBN 91-7372-805-5.

No 155 **Christos Levcopoulos:** Heuristics for Minimum Decompositions of Polygons, 1987, ISBN 91-7870-133-3.

No 165 **James W. Goodwin:** A Theory and System for Non-Monotonic Reasoning, 1987, ISBN 91-7870-183-X.

No 170 **Zebo Peng:** A Formal Methodology for Automated Synthesis of VLSI Systems, 1987, ISBN 91-7870-225-9.

No 174 **Johan Fagerström:** A Paradigm and System for Design of Distributed Systems, 1988, ISBN 91-7870-301-8.

No 192 **Dimiter Driankov:** Towards a Many Valued Logic of Quantified Belief, 1988, ISBN 91-7870-374-3.

No 213 **Lin Padgham:** Non-Monotonic Inheritance for an Object Oriented Knowledge Base, 1989, ISBN 91-7870-485-5.

No 214 **Tony Larsson:** A Formal Hardware Description and Verification Method, 1989, ISBN 91-7870-517-7.

No 221 **Michael Reinfrank:** Fundamentals and Logical Foundations of Truth Maintenance, 1989, ISBN 91-7870-546-0.

No 239 **Jonas Löwgren:** Knowledge-Based Design Support and Discourse Management in User Interface Management Systems, 1991, ISBN 91-7870-720-X.

No 244 **Henrik Eriksson:** Meta-Tool Support for Knowledge Acquisition, 1991, ISBN 91-7870-746-3.

No 252 **Peter Eklund:** An Epistemic Approach to Interactive Design in Multiple Inheritance Hierarchies,1991, ISBN 91-7870-784-6.

No 258 **Patrick Doherty:** NML3 - A Non-Monotonic Formalism with Explicit Defaults, 1991, ISBN 91-7870-816-8.

No 260 **Nahid Shahmehri:** Generalized Algorithmic Debugging, 1991, ISBN 91-7870-828-1.

No 264 **Nils Dahlbäck:** Representation of Discourse-Cognitive and Computational Aspects, 1992, ISBN 91-7870-850-8.

No 265 **Ulf Nilsson:** Abstract Interpretations and Abstract Machines: Contributions to a Methodology for the Implementation of Logic Programs, 1992, ISBN 91-7870-858-3.

No 270 **Ralph Rönnquist:** Theory and Practice of Tense-bound Object References, 1992, ISBN 91-7870-873-7.

No 273 **Björn Fjellborg:** Pipeline Extraction for VLSI Data Path Synthesis, 1992, ISBN 91-7870-880-X.

No 276 **Staffan Bonnier:** A Formal Basis for Horn Clause Logic with External Polymorphic Functions, 1992, ISBN 91-7870-896-6.

No 277 **Kristian Sandahl:** Developing Knowledge Management Systems with an Active Expert Methodology, 1992, ISBN 91-7870-897-4.

No 281 **Christer Bäckström:** Computational Complexity of Reasoning about Plans, 1992, ISBN 91-7870-979-2.

No 292 **Mats Wirén:** Studies in Incremental Natural Language Analysis, 1992, ISBN 91-7871-027-8.

No 297 **Mariam Kamkar:** Interprocedural Dynamic Slicing with Applications to Debugging and Testing, 1993, ISBN 91-7871-065-0.

No 302 **Tingting Zhang:** A Study in Diagnosis Using Classification and Defaults, 1993, ISBN 91-7871-078-2.

No 312 **Arne Jönsson:** Dialogue Management for Natural Language Interfaces - An Empirical Approach, 1993, ISBN 91-7871-110-X.

No 338 **Simin Nadjm-Tehrani**: Reactive Systems in Physical Environments: Compositional Modelling and Framework for Verification, 1994, ISBN 91-7871-237-8.

No 371 **Bengt Savén:** Business Models for Decision Support and Learning. A Study of Discrete-Event Manufacturing Simulation at Asea/ABB 1968-1993, 1995, ISBN 91-7871-494-X.

No 375 **Ulf Söderman:** Conceptual Modelling of Mode Switching Physical Systems, 1995, ISBN 91-7871-516-4.

No 383 **Andreas Kågedal:** Exploiting Groundness in Logic Programs, 1995, ISBN 91-7871-538-5.

No 396 **George Fodor:** Ontological Control, Description, Identification and Recovery from Problematic Control Situations, 1995, ISBN 91-7871-603-9.

No 413 **Mikael Pettersson:** Compiling Natural Semantics, 1995, ISBN 91-7871-641-1.

No 414 **Xinli Gu:** RT Level Testability Improvement by Testability Analysis and Transformations, 1996, ISBN 91-7871-654-3.

No 416 **Hua Shu:** Distributed Default Reasoning, 1996, ISBN 91-7871-665-9.

No 429 **Jaime Villegas:** Simulation Supported Industrial Training from an Organisational Learning Perspective - Development and Evaluation of the SSIT Method, 1996, ISBN 91-7871-700-0.

No 431 **Peter Jonsson:** Studies in Action Planning: Algorithms and Complexity, 1996, ISBN 91-7871-704-3.

No 437 **Johan Boye:** Directional Types in Logic Programming, 1996, ISBN 91-7871-725-6.

No 439 **Cecilia Sjöberg:** Activities, Voices and Arenas: Participatory Design in Practice, 1996, ISBN 91-7871-728-0.

No 448 **Patrick Lambrix:** Part-Whole Reasoning in Description Logics, 1996, ISBN 91-7871-820-1.

No 452 **Kjell Orsborn:** On Extensible and Object-Relational Database Technology for Finite Element Analysis Applications, 1996, ISBN 91-7871-827-9.

No 459 **Olof Johansson:** Development Environments for Complex Product Models, 1996, ISBN 91-7871-855-4.

No 461 **Lena Strömbäck:** User-Defined Constructions in Unification-Based Formalisms,1997, ISBN 91-7871-857-0.

No 462 **Lars Degerstedt:** Tabulation-based Logic Programming: A Multi-Level View of Query Answering, 1996, ISBN 91-7871-858-9.

No 475 **Fredrik Nilsson:** Strategi och ekonomisk styrning - En studie av hur ekonomiska styrsystem utformas och används efter företagsförvärv, 1997, ISBN 91-7871-914-3.

No 480 **Mikael Lindvall:** An Empirical Study of Requirements-Driven Impact Analysis in Object-Oriented Software Evolution, 1997, ISBN 91-7871-927-5.

No 485 **Göran Forslund**: Opinion-Based Systems: The Cooperative Perspective on Knowledge-Based Decision Support, 1997, ISBN 91-7871-938-0.

No 494 **Martin Sköld**: Active Database Management Systems for Monitoring and Control, 1997, ISBN 91-7219-002-7.

No 495 **Hans Olsén**: Automatic Verification of Petri Nets in a CLP framework, 1997, ISBN 91-7219-011-6.

No 498 **Thomas Drakengren:** Algorithms and Complexity for Temporal and Spatial Formalisms, 1997, ISBN 91-7219-019-1.

No 502 **Jakob Axelsson:** Analysis and Synthesis of Heterogeneous Real-Time Systems, 1997, ISBN 91-7219-035-3.

No 503 **Johan Ringström:** Compiler Generation for Data-Parallel Programming Langugaes from Two-Level Semantics Specifications, 1997, ISBN 91-7219-045-0.

No 512 **Anna Moberg:** Närhet och distans - Studier av kommunikationsmmönster i satellitkontor och flexibla kontor, 1997, ISBN 91-7219-119-8.

No 520 **Mikael Ronström:** Design and Modelling of a Parallel Data Server for Telecom Applications, 1998, ISBN 91-7219-169-4.

No 522 **Niclas Ohlsson:** Towards Effective Fault Prevention - An Empirical Study in Software Engineering, 1998, ISBN 91-7219-176-7.

No 526 **Joachim Karlsson:** A Systematic Approach for Prioritizing Software Requirements, 1998, ISBN 91-7219-184-8.

No 530 **Henrik Nilsson:** Declarative Debugging for Lazy Functional Languages, 1998, ISBN 91-7219-197-x.

No 555 **Jonas Hallberg:** Timing Issues in High-Level Synthesis,1998, ISBN 91-7219-369-7.

No 561 **Ling Lin:** Management of 1-D Sequence Data - From Discrete to Continuous, 1999, ISBN 91-7219-402-2.

No 563 **Eva L Ragnemalm:** Student Modelling based on Collaborative Dialogue with a Learning Companion, 1999, ISBN 91-7219-412-X.

No 567 **Jörgen Lindström:** Does Distance matter? On geographical dispersion in organisations, 1999, ISBN 91-7219-439-1.

No 582 **Vanja Josifovski:** Design, Implementation and

Evaluation of a Distributed Mediator System for Data Integration, 1999, ISBN 91-7219-482-0.

No 589 **Rita Kovordányi**: Modeling and Simulating Inhibitory Mechanisms in Mental Image Reinterpretation - Towards Cooperative Human-Computer Creativity, 1999, ISBN 91-7219-506-1.

No 592 **Mikael Ericsson:** Supporting the Use of Design Knowledge - An Assessment of Commenting Agents, 1999, ISBN 91-7219-532-0.

No 593 **Lars Karlsson:** Actions, Interactions and Narratives, 1999, ISBN 91-7219-534-7.

No 594 **C. G. Mikael Johansson:** Social and Organizational Aspects of Requirements Engineering Methods - A practice-oriented approach, 1999, ISBN 91-7219-541-X.

No 595 **Jörgen Hansson:** Value-Driven Multi-Class Overload Management in Real-Time Database Systems, 1999, ISBN 91-7219-542-8.

No 596 **Niklas Hallberg:** Incorporating User Values in the Design of Information Systems and Services in the Public Sector: A Methods Approach, 1999, ISBN 91-7219-543-6.

No 597 **Vivian Vimarlund:** An Economic Perspective on the Analysis of Impacts of Information Technology: From Case Studies in Health-Care towards General Models and Theories, 1999, ISBN 91-7219-544-4.

No 598 **Johan Jenvald:** Methods and Tools in Computer-Supported Taskforce Training, 1999, ISBN 91-7219-547-9.

No 607 **Magnus Merkel:** Understanding and enhancing translation by parallel text processing, 1999, ISBN 91-7219-614-9.

No 611 **Silvia Coradeschi:** Anchoring symbols to sensory data, 1999, ISBN 91-7219-623-8.

No 613 **Man Lin:** Analysis and Synthesis of Reactive Systems: A Generic Layered Architecture Perspective, 1999, ISBN 91-7219-630-0.

No 618 **Jimmy Tjäder:** Systemimplementering i praktiken - En studie av logiker i fyra projekt, 1999, ISBN 91-7219-657-2.

No 627 **Vadim Engelson:** Tools for Design, Interactive Simulation, and Visualization of Object-Oriented Models in Scientific Computing, 2000, ISBN 91-7219-709-9.

No 637 **Esa Falkenroth:** Database Technology for Control and Simulation, 2000, ISBN 91-7219-766-8.

No 639 **Per-Arne Persson:** Bringing Power and Knowledge Together: Information Systems Design for Autonomy and Control in Command Work, 2000, ISBN 91-7219-796-X.

No 660 **Erik Larsson:** An Integrated System-Level Design for Testability Methodology, 2000, ISBN 91-7219-890-7.

No 688 **Marcus Bjäreland:** Model-based Execution Monitoring, 2001, ISBN 91-7373-016-5.

No 689 **Joakim Gustafsson:** Extending Temporal Action Logic, 2001, ISBN 91-7373-017-3.

No 720 **Carl-Johan Petri:** Organizational Information Provision - Managing Mandatory and Discretionary Use of Information Technology, 2001, ISBN-91-7373-126-9.

No 724 **Paul Scerri:** Designing Agents for Systems with Adjustable Autonomy, 2001, ISBN 91 7373 207 9.

No 725 **Tim Heyer**: Semantic Inspection of Software Artifacts: From Theory to Practice, 2001, ISBN 91 7373 208 7.

No 726 **Pär Carlshamre:** A Usability Perspective on Requirements Engineering - From Methodology to Product Development, 2001, ISBN 91 7373 212 5.

No 732 **Juha Takkinen:** From Information Management to Task Management in Electronic Mail, 2002, ISBN 91 7373 258 3.

No 745 **Johan Åberg:** Live Help Systems: An Approach to Intelligent Help for Web Information Systems, 2002, ISBN 91-7373-311-3.

No 746 **Rego Granlund:** Monitoring Distributed Teamwork Training, 2002, ISBN 91-7373-312-1.

No 757 **Henrik André-Jönsson:** Indexing Strategies for Time Series Data, 2002, ISBN 917373-346-6.

No 747 **Anneli Hagdahl:** Development of IT-suppor-ted Inter-organisational Collaboration - A Case Study in the Swedish Public Sector, 2002, ISBN 91-7373-314-8.

No 749 **Sofie Pilemalm:** Information Technology for Non-Profit Organisations - Extended Participatory Design of an Information System for Trade Union Shop Stewards, 2002, ISBN 91-7373-318-0.

No 765 **Stefan Holmlid:** Adapting users: Towards a theory of use quality, 2002, ISBN 91-7373-397-0.

No 771 **Magnus Morin:** Multimedia Representations of Distributed Tactical Operations, 2002, ISBN 91-7373-421-7.

No 772 **Pawel Pietrzak:** A Type-Based Framework for Locating Errors in Constraint Logic Programs, 2002, ISBN 91-7373-422-5.

No 758 **Erik Berglund:** Library Communication Among Programmers Worldwide, 2002, ISBN 91-7373-349-0.

No 774 **Choong-ho Yi:** Modelling Object-Oriented Dynamic Systems Using a Logic-Based Framework, 2002, ISBN 91-7373-424-1.

No 779 **Mathias Broxvall:** A Study in the Computational Complexity of Temporal Reasoning, 2002, ISBN 91-7373-440-3.

No 793 **Asmus Pandikow:** A Generic Principle for Enabling Interoperability of Structured and Object-Oriented Analysis and Design Tools, 2002, ISBN 91-7373-479-9.

No 785 **Lars Hult:** Publika Informationstjänster. En studie av den Internetbaserade encyklopedins bruksegenskaper, 2003, ISBN 91-7373-461-6.

No 800 **Lars Taxén:** A Framework for the Coordination of Complex Systems´ Development, 2003, ISBN 91-7373-604-X

No 808 **Klas Gäre:** Tre perspektiv på förväntningar och förändringar i samband med införande av informationsystem, 2003, ISBN 91-7373-618-X.

No 821 **Mikael Kindborg:** Concurrent Comics - programming of social agents by children, 2003, ISBN 91-7373-651-1.

No 823 **Christina Ölvingson:** On Development of Information Systems with GIS Functionality in Public Health Informatics: A Requirements Engineering Approach, 2003, ISBN 91-7373-656-2.

No 828 **Tobias Ritzau:** Memory Efficient Hard Real-Time Garbage Collection, 2003, ISBN 91-7373-666-X.

No 833 **Paul Pop:** Analysis and Synthesis of Communication-Intensive Heterogeneous Real-Time Systems, 2003, ISBN 91-7373-683-X.

No 852 **Johan Moe:** Observing the Dynamic Behaviour of Large Distributed Systems to Improve Development and Testing - An Emperical Study in Software Engineering, 2003, ISBN 91-7373-779-8.

No 867 **Erik Herzog:** An Approach to Systems Engineering Tool Data Representation and Exchange, 2004, ISBN 91-7373-929-4.

No 872 **Aseel Berglund:** Augmenting the Remote Control: Studies in Complex Information Navigation for Digital TV, 2004, ISBN 91-7373-940-5.

No 869 **Jo Skåmedal:** Telecommuting's Implications on Travel and Travel Patterns, 2004, ISBN 91-7373-935-9.

No 870 **Linda Askenäs:** The Roles of IT - Studies of Organising when Implementing and Using Enterprise Systems, 2004, ISBN 91-7373-936-7.

No 874 **Annika Flycht-Eriksson:** Design and Use of Ontologies in Information-Providing Dialogue Systems, 2004, ISBN 91-7373-947-2.

No 873 **Peter Bunus:** Debugging Techniques for Equation-Based Languages, 2004, ISBN 91-7373-941-3.

No 876 **Jonas Mellin:** Resource-Predictable and Efficient Monitoring of Events, 2004, ISBN 91-7373-956-1.

No 883 **Magnus Bång:** Computing at the Speed of Paper: Ubiquitous Computing Environments for Healthcare Professionals, 2004, ISBN 91-7373-971-5

No 882 **Robert Eklund:** Disfluency in Swedish human-human and human-machine travel booking dialogues, 2004. ISBN 91-7373-966-9.

No 887 **Anders Lindström:** English and other Foreign Linquistic Elements in Spoken Swedish. Studies of Productive Processes and their Modelling using Finite-State Tools, 2004, ISBN 91-7373-981-2.

No 889 **Zhiping Wang:** Capacity-Constrained Production-inventory systems - Modellling and Analysis in both a traditional and an e-business context, 2004, ISBN 91-85295-08-6.

No 893 **Pernilla Qvarfordt:** Eyes on Multimodal Interaction, 2004, ISBN 91-85295-30-2.

No 910 **Magnus Kald:** In the Borderland between Strategy and Management Control - Theoretical Framework and Empirical Evidence, 2004, ISBN 91-85295-82-5.

No 918 **Jonas Lundberg:** Shaping Electronic News: Genre Perspectives on Interaction Design, 2004, ISBN 91-85297-14-3.

No 900 **Mattias Arvola:** Shades of use: The dynamics of interaction design for sociable use, 2004, ISBN 91-85295-42-6.

No 920 **Luis Alejandro Cortés:** Verification and Scheduling Techniques for Real-Time Embedded Systems, 2004, ISBN 91-85297-21-6.

No 929 **Diana Szentivanyi:** Performance Studies of Fault-Tolerant Middleware, 2005, ISBN 91-85297-58-5.

No 933 **Mikael Cäker:** Management Accounting as Constructing and Opposing Customer Focus: Three Case Studies on Management Accounting and Customer Relations, 2005, ISBN 91-85297-64-X.

No 937 **Jonas Kvarnström:** TALplanner and Other Extensions to Temporal Action Logic, 2005, ISBN 91-85297-75-5.

No 938 **Bourhane Kadmiry:** Fuzzy Gain-Scheduled Visual Servoing for Unmanned Helicopter, 2005, ISBN 91-85297-76-3.

No 945 **Gert Jervan:** Hybrid Built-In Self-Test and Test Generation Techniques for Digital Systems, 2005, ISBN: 91-85297-97-6.

No 946 **Anders Arpteg:** Intelligent Semi-Structured Information Extraction, 2005, ISBN 91-85297-98-4.

No 947 **Ola Angelsmark:** Constructing Algorithms for Constraint Satisfaction and Related Problems - Methods and Applications, 2005, ISBN 91-85297-99-2.

No 963 **Calin Curescu:** Utility-based Optimisation of Resource Allocation for Wireless Networks, 2005. ISBN 91-85457-07-8.

No 972 **Björn Johansson:** Joint Control in Dynamic Situations, 2005, ISBN 91-85457-31-0.

No 974 **Dan Lawesson:** An Approach to Diagnosability Analysis for Interacting Finite State Systems, 2005, ISBN 91-85457-39-6.

No 979 **Claudiu Duma:** Security and Trust Mechanisms for Groups in Distributed Services, 2005, ISBN 91-85457-54-X.

No 983 **Sorin Manolache:** Analysis and Optimisation of Real-Time Systems with Stochastic Behaviour, 2005, ISBN 91-85457-60-4.

No 986 **Yuxiao Zhao:** Standards-Based Application Integration for Business-to-Business Communications, 2005, ISBN 91-85457-66-3.

No 1004 **Patrik Haslum:** Admissible Heuristics for Automated Planning, 2006, ISBN 91-85497-28-2.

No 1005 **Aleksandra Tešanovic:** Developing Reusable and Reconfigurable Real-Time Software using Aspects and Components, 2006, ISBN 91-85497-29-0.

No 1008 **David Dinka:** Role, Identity and Work: Extending the design and development agenda, 2006, ISBN 91-85497-42-8.

No 1009 **Iakov Nakhimovski:** Contributions to the Modeling and Simulation of Mechanical Systems with Detailed Contact Analysis, 2006, ISBN 91-85497-43-X.

No 1013 **Wilhelm Dahllöf:** Exact Algorithms for Exact Satisfiability Problems, 2006, ISBN 91-85523-97-6.

No 1016 **Levon Saldamli:** PDEModelica - A High-Level Language for Modeling with Partial Differential Equations, 2006, ISBN 91-85523-84-4.

No 1017 **Daniel Karlsson:** Verification of Component-based Embedded System Designs, 2006, ISBN 91-85523-79-8.

No 1018 **Ioan Chisalita:** Communication and Networking Techniques for Traffic Safety Systems, 2006, ISBN 91-85523-77-1.

No 1019 **Tarja Susi:** The Puzzle of Social Activity - The Significance of Tools in Cognition and Cooperation, 2006, ISBN 91-85523-71-2.

No 1021 **Andrzej Bednarski:** Integrated Optimal Code Generation for Digital Signal Processors, 2006, ISBN 91-85523-69-0.

No 1022 **Peter Aronsson:** Automatic Parallelization of Equation-Based Simulation Programs, 2006, ISBN 91-85523-68-2.

No 1023 **Sonia Sangari:** Some Visual Correlates to Focal Accent in Swedish, 2006, ISBN 91-85523-67-4.

No 1030 **Robert Nilsson:** A Mutation-based Framework for Automated Testing of Timeliness, 2006, ISBN 91-85523-35-6.

No 1034 **Jon Edvardsson:** Techniques for Automatic Generation of Tests from Programs and Specifications, 2006, ISBN 91-85523-31-3.

No 1035 **Vaida Jakoniene:** Integration of Biological Data, 2006, ISBN 91-85523-28-3.

No 1045 **Genevieve Gorrell:** Generalized Hebbian Algorithms for Dimensionality Reduction in Natural Language Processing, 2006, ISBN 91-85643-88-2.

No 1051 **Yu-Hsing Huang:** Having a New Pair of Glasses - Applying Systemic Accident Models on Road Safety, 2006, ISBN 91-85643-64-5.

No 1054 **Åsa Hedenskog:** Perceive those things which cannot be seen - A Cognitive Systems Engineering perspective on requirements management, 2006, ISBN 91-85643-57-2.

No 1061 **Cécile Åberg:** An Evaluation Platform for Semantic Web Technology, 2007, ISBN 91-85643-31-9.

No 1073 **Mats Grindal:** Handling Combinatorial Explosion in Software Testing, 2007, ISBN 978-91-85715-74-9.

No 1075 **Almut Herzog:** Usable Security Policies for Runtime Environments, 2007, ISBN 978-91-85715-65-7.

No 1079 **Magnus Wahlström:** Algorithms, measures, and upper bounds for satisfiability and related problems, 2007, ISBN 978-91-85715-55-8.

No 1083 **Jesper Andersson:** Dynamic Software Architectures, 2007, ISBN 978-91-85715-46-6.

No 1086 **Ulf Johansson:** Obtaining Accurate and Comprehensible Data Mining Models - An Evolutionary Approach, 2007, ISBN 978-91-85715-34-3.

No 1089 **Traian Pop:** Analysis and Optimisation of Distributed Embedded Systems with Heterogeneous Scheduling Policies, 2007, ISBN 978-91-85715-27-5.

No 1091 **Gustav Nordh:** Complexity Dichotomies for CSP-related Problems, 2007, ISBN 978-91-85715-20-6.

No 1106 **Per Ola Kristensson:** Discrete and Continuous Shape Writing for Text Entry and Control, 2007, ISBN 978-91-85831-77-7.

No 1110 **He Tan:** Aligning Biomedical Ontologies, 2007, ISBN 978-91-85831-56-2.

No 1112 **Jessica Lindblom:** Minding the body - Interacting socially through embodied action, 2007, ISBN 978-91-85831-48-7.

No 1113 **Pontus Wärnestål:** Dialogue Behavior Management in Conversational Recommender Systems, 2007, ISBN 978-91-85831-47-0.

Linköping Studies in Information Science

No 1 **Karin Axelsson:** Metodisk systemstrukturering- att skapa samstämmighet mellan informa-tionssystemarkitektur och verksamhet, 1998. ISBN-9172-19-296-8.

No 2 **Stefan Cronholm:** Metodverktyg och användbarhet - en studie av datorstödd metodbaserad systemutveckling, 1998. ISBN-9172-19-299-2.

No 3 **Anders Avdic:** Användare och utvecklare - om anveckling med kalkylprogram, 1999. ISBN-91-7219-606-8.

No 4 **Owen Eriksson:** Kommunikationskvalitet hos informationssystem och affärsprocesser, 2000. ISBN 91-7219-811-7.

No 5 **Mikael Lind:** Från system till process - kriterier för processbestämning vid verksamhetsanalys, 2001, ISBN 91-7373-067-X

No 6 **Ulf Melin:** Koordination och informationssystem i företag och nätverk, 2002, ISBN 91-7373-278-8.

No 7 **Pär J. Ågerfalk:** Information Systems Actability - Understanding Information Technology as a Tool for Business Action and Communication, 2003, ISBN 91-7373-628-7.

No 8 **Ulf Seigerroth:** Att förstå och förändra systemutvecklingsverksamheter - en taxonomi för metautveckling, 2003, ISBN91-7373-736-4.

No 9 **Karin Hedström:** Spår av datoriseringens värden - Effekter av IT i äldreomsorg, 2004, ISBN 91-7373-963-4.

No 10 **Ewa Braf:** Knowledge Demanded for Action - Studies on Knowledge Mediation in Organisations, 2004, ISBN 91-85295-47-7.

No 11 **Fredrik Karlsson:** Method Configuration - method and computerized tool support, 2005, ISBN 91-85297-48-8.

No 12 **Malin Nordström:** Styrbar systemförvaltning - Att organisera systemförvaltningsverksamhet med hjälp av effektiva förvaltningsobjekt, 2005, ISBN 91-85297-60-7.

No 13 **Stefan Holgersson:** Yrke: POLIS - Yrkeskunskap, motivation, IT-system och andra förutsättningar för polisarbete, 2005, ISBN 91-85299-43-X.

No 14 **Benneth Christiansson, Marie-Therese Christiansson:** Mötet mellan process och komponent - mot ett ramverk för en verksamhetsnära kravspecifikation vid anskaffning av komponentbaserade informationssystem, 2006, ISBN 91-85643-22-X.