Cognitive Context Elicitation and Modeling

by

Lin Mei

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 2011 by Lin Mei
Abstract

Cognitive Context Elicitation and Modeling

Lin Mei
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2011

As computing becomes ubiquitous and intelligent, it is possible for systems to adapt their behavior based on information sensed from the situational context. However, determining the context space has been taken for granted in most ubiquitous applications, and so that context-adaptive systems often miss the situational factors that are most relevant to users. The mismatch between a system’s computational model and users’ mental model of the context may frustrate and disorient users. This thesis describes the CCM (cognitive context model)-based approach for eliciting individual cognitive views of a context-aware task and selecting an appropriate context space for context-aware computing. It captures the situational and cognitive context for each task, using a structural architecture in which individual participants use a context view to describe their situational perspective of the task. Clustering and optimization techniques are applied to analyze and integrate context views in CCM. Developers can use the optimization output to identify an appropriate context space, specify context-aware adaptation policies and resolve run-time policy conflicts. This approach simplifies the task of context elicitation, emphasizes individual variance in context-aware activity, and helps avoid user requirements misunderstanding.
Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor Prof. Steve Easterbrook, for his inspiration and guidance during the whole work.

I would thank all my advisory committees, Prof. Khai Truong, Prof. Eyal de Lara, and Prof. Eric Yu, for their valuable advice and warm encouragement.

It is an honour to have Professor Alistair Sutcliffe at University of Manchester as the external reviewer. He read this thesis very carefully and provided many useful comments.

I am grateful to all my colleagues and friends in our department, for being a close family during these years I stayed here and for their continued care and attention. People of the SE Lab, Alicia Grubb, Jonathan Lung, Nan Niu, Jon Pipitone, and all others, are especially thanked for their inspiring advices and discussions.

Finally, I would thank my parents and my husband Chenyun Wu. Without their support and understanding, it would have been impossible for me to finish this work.
Contents

1 Introduction .................................................. 1
  1.1 Context Modeling .............................................. 1
  1.2 Cognitive Context Views ....................................... 3
  1.3 Context Space Optimization .................................... 5
  1.4 Contributions of the Thesis .................................... 6
  1.5 Overview of the Thesis ....................................... 8

2 Literature Review .............................................. 11
  2.1 Context .......................................................... 11
    2.1.1 Definitions of Context .................................... 12
    2.1.2 Dimensions of Context .................................... 13
    2.1.3 Summary .................................................... 15
  2.2 Context-aware Human Activity ................................. 17
    2.2.1 Situated Action ............................................. 17
    2.2.2 Activity Theory ............................................. 19
    2.2.3 Distributed Cognition ..................................... 23
    2.2.4 Summary .................................................... 25
  2.3 Context Modeling & Context-aware Computing .................. 27
    2.3.1 Context-aware Computing .................................. 27
    2.3.2 Object-based Context Models ............................... 29
3 Goal-based Analysis of Existing Work on Context-aware Computing

3.1 A Running Example: Smart Meeting Room

3.2 Goal Modeling

3.3 Understanding User Intention with i*

3.3.1 Distributed Intentionality

3.3.2 Individual Intentionality

3.3.3 Scenario Analysis

3.4 Goal-based Evaluation of Context-aware Applications

3.4.1 The Framework of Goal-based Evaluation

3.4.2 Results Analysis

3.5 Summary

4 The Framework of Cognitive Context Modeling

4.1 Methodology Overview

4.2 Cognitive Context Model (CCM)

4.2.1 Task

4.2.2 ObjCt

4.2.3 CogCt
4.3 Summary ................................................................. 83

5 Context Space Specification ........................................... 84

5.1 Context Space Identification .......................................... 84
  5.1.1 The Analytic Hierarchy Process Method ....................... 86
  5.1.2 Integrating Views with AHP ................................... 87

5.2 Context Specification .................................................. 91
  5.2.1 Context-Action Policy .......................................... 91
  5.2.2 Specify Context-Action Policy with CCM ...................... 94
  5.2.3 Handling Rule Conflicts ........................................ 97

5.3 Summary .................................................................. 98

6 Case Studies and Evaluation ............................................. 100

6.1 Case Study 1: Smart Meeting Room ................................. 101
  6.1.1 Data Collection .................................................. 101
  6.1.2 Model Building .................................................. 103
  6.1.3 Model Analysis .................................................. 106

6.2 Case Study 2: Power Management ................................. 115
  6.2.1 Data Collection .................................................. 116
  6.2.2 Model Building .................................................. 117
  6.2.3 Model Analysis and Context Specification ...................... 122
  6.2.4 Conflict Resolution .............................................. 130
  6.2.5 Runtime Context Reconfiguration ............................ 131

6.3 Summary and Analysis ................................................ 133

7 Tool Support ............................................................ 137

7.1 Architecture ............................................................. 137

7.2 User Interface ........................................................... 139

7.3 Data Storage ............................................................. 141
8 Conclusions and Future Work

8.1 Summary ......................................................... 147
  8.1.1 Problem Domain .......................................... 147
  8.1.2 Objectives ................................................. 148
  8.1.3 Solution .................................................... 149
8.2 Analysis and Evaluation ..................................... 150
  8.2.1 Advantages ............................................... 150
  8.2.2 Remaining Problems .................................... 152
8.3 Future Work ..................................................... 153

Bibliography .......................................................... 156
## List of Tables

2.1 The comparison of three theories on context study . . . . . . . . . . . . . . . 25  
2.2 The comparison of different approaches to context modeling . . . . . . . . 35  
3.1 Goal-based analysis on scenarios of smart meeting room . . . . . . . . . . . 59  
3.2 Results of model comparison . . . . . . . . . . . . . . . . . . . . . . . . . . 67  
4.1 Dimensions of context space . . . . . . . . . . . . . . . . . . . . . . . . . . 80  
5.1 Mapping CCM to AHP hierarchy . . . . . . . . . . . . . . . . . . . . . . . . 87  
5.2 Concept Mapping between CCM and context-action policy . . . . . . . . . 94  
6.1 Summary of the smart meeting CCM . . . . . . . . . . . . . . . . . . . . . . 107  
6.2 Different models of smart meeting room . . . . . . . . . . . . . . . . . . . . 109  
6.3 Time contexts of computer power management . . . . . . . . . . . . . . . 126
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Zimmerman’s five fundamental categories of context information</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>The basic activity model [58]</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>Engestrom’s activity model</td>
<td>21</td>
</tr>
<tr>
<td>2.4</td>
<td>Hierarchical levels of an activity [58]</td>
<td>22</td>
</tr>
<tr>
<td>2.5</td>
<td>Modeling context in context-based service composition [80]</td>
<td>30</td>
</tr>
<tr>
<td>2.6</td>
<td>Extended ORM for context modeling [45]</td>
<td>32</td>
</tr>
<tr>
<td>2.7</td>
<td>CONON for modeling context in pervasive Environments [112]</td>
<td>33</td>
</tr>
<tr>
<td>3.1</td>
<td>Layout of a meeting room</td>
<td>48</td>
</tr>
<tr>
<td>3.2</td>
<td>Interaction and state transition during a presentation</td>
<td>53</td>
</tr>
<tr>
<td>3.3</td>
<td>Scheduling a meeting: sequence diagram</td>
<td>54</td>
</tr>
<tr>
<td>3.4</td>
<td>Collaboration in a meeting scenario</td>
<td>55</td>
</tr>
<tr>
<td>3.5</td>
<td>Discussion during a presentation: dependency diagram</td>
<td>56</td>
</tr>
<tr>
<td>3.6</td>
<td>Goal refinement for the “Data Sharing” task in a smart meeting room</td>
<td>57</td>
</tr>
<tr>
<td>3.7</td>
<td>Evaluating context-aware computing with goal models</td>
<td>61</td>
</tr>
<tr>
<td>3.8</td>
<td>Steps for extracting system goal models</td>
<td>63</td>
</tr>
<tr>
<td>3.9</td>
<td>The user goal model for a smart meeting room</td>
<td>65</td>
</tr>
<tr>
<td>3.10</td>
<td>The system goal model for iRoom project</td>
<td>65</td>
</tr>
<tr>
<td>3.11</td>
<td>The system goal model for EasyLiving project</td>
<td>66</td>
</tr>
<tr>
<td>3.12</td>
<td>Evaluation result</td>
<td>68</td>
</tr>
</tbody>
</table>
4.1 The structure of adaptation engine ........................................ 72
4.2 Context-aware task processing .......................................... 73
4.3 The process of context elicitation and modeling ...................... 74
4.4 The conceptual structure of CCM ....................................... 76
4.5 An example of task model ................................................. 77
4.6 An example of context view ............................................... 82

5.1 A simple AHP hierarchy .................................................... 86
5.2 An example of AHP calculation .......................................... 89
5.3 Optimized context view ..................................................... 90

6.1 Ontology-based context model for smart meeting room (reproduced from [48]) 102
6.2 The task model of smart meeting room ................................ 103
6.3 A context view of the “meeting” task .................................. 104
6.4 Cognitive context model of smart meeting room ..................... 106
6.5 An integrated context view of the “meeting” task .................... 110
6.6 The task model of power management .................................. 117
6.7 Power state changes over time .......................................... 118
6.8 K-means clustering on context views (k=3) .......................... 119
6.9 Context views of power management .................................. 121
6.10 The XML representation of CCM for power management ......... 122
6.11 The optimized view of power management .......................... 123
6.12 Optimizing time factors for context-aware power management .... 125

7.1 The system architecture of supporting environment ................. 138
7.2 A screen snapshot from CCM toolkit, showing the interface of adding
    contextual elements to a context view. ............................... 139
7.3 Plug-in tools in the CCM toolkit ....................................... 140
7.4 A screen snapshot from CCM toolkit, showing the interface of adding contextual elements to a context view. 140
7.5 A screen snapshot of XML-formatted data storage 141
7.6 The data model of CCM toolkit 142
7.7 The use case diagram of CCM toolkit 143
Chapter 1

Introduction

Context-awareness is one of the key challenges in ubiquitous computing. Using an appropriate context model that relates contexts with specific activities is one step towards efficient and high-quality context-aware system design. This thesis describes a framework for the elicitation of cognitive views from activity participants, and the subsequent process of identifying patterns in gathered views, balancing trade-offs, and optimizing context space for ubiquitous computing.

1.1 Context Modeling

In a complex world, human action is highly dependent upon its material and social circumstances, i.e., its context [104]. Context-dependent action is essential for effective learning, reasoning and communication, in almost every area of our thinking and action e.g., language understanding, memory, social cognition, and problem solving. During a conversation, a lecture, doctor-patient interaction, exercises or TV watching, people monitor their environment, and take their context into account for planning, executing, controlling and understanding.

As computing becomes ubiquitous and intelligent, users also expect the devices they use to adapt their behaviour based on information sensed from the physical and computa-
Chapter 1. Introduction

tion environment. However, context-awareness exhibited by people is radically different than that of computational systems. People notice and integrate a vast range of cues, both obvious and subtle, and interpret them in light of their previous experience to define their context. In contrast, context-aware ubiquitous systems detect a very small set of cues, typically quantitative variations of the dimensions for which they have sensors, and use hard coded context models or explicit adaptation rules, which provide limited learning ability [34]. There is always a conflict between the infinite, subjective detail of human activity and the finite, objective aspects of system design.

Ubiquitous computing embraces a model in which users, services, and resources discover other users, services and resources and integrate them into a useful experience. This inevitably involves deciding exactly the relevant contextual facts for discovery, and minimizing system’s model of interaction and user’s mental model of the system. The elicitation and formulation of contextual facts as well as their interdependencies to form unified context space of a ubiquitous task is termed as context modeling.

Unified and precise description of relevant context is vital for ubiquitous projects. The relevant context is some subset of the environment, where contextual elements relate in some way to the information content of the system [24, 98]. However, the scope of the context is not fixed; it can be adjusted to include more or less knowledge into the reasoning process [22]. Hence, context modeling involves the development of an appropriate analytical abstraction that discards irrelevant details while isolating and emphasizing those properties of artifacts and situations which are most significant for design.

A fundamental goal of research in context modeling is automation. Contextual-reasoning is not amenable to classical reasoning, i.e. knowing B in context A is not transferable into statements like “if A then B”. The benefit of recognizing context relies on the fact that we can learn a simple model in one circumstance and successfully use it in another circumstance that is approximately similar to the first (i.e. in the same
“context”) [25]. Contextual-reasoning offers a number of advantages for users, such as automating task execution [14, 55, 56], providing relevant services for a user [4, 15, 93], and tagging context information to support later retrieval [12, 116]. It is widely accepted that a unified context specification can facilitate context discovery, contextual-reasoning and interoperability of applications.

As context modeling is concerned with the interpretation of the world in which ubiquitous system operates, it can never be completely formalized. Rather, the major goal of context modeling is to provide better support for the construction of context space which appropriately reflects end-users’ mental state and provides scalable methods of context processing and management. This support should include the guidance for the elicitation and formulation of contextual facts and the subsequent knowledge management and validation process. This thesis concentrates on how these activities might be supported.

1.2 Cognitive Context Views

Humans are the essential source of information and the main target of computation in ubiquitous computing environment. Context models must be consistent with human cognition so that the system interacts smoothly with people. Although a simple piece of contextual information, e.g. time and location, can be used in ubiquitous system to provide relevant information and/or services to the user, theories of sociology and philosophy, especially the ethnomethodology and phenomenology, suggest that user experience, such as subjectively perceived features and past experience of similar context, may influence current activity [25]. Over-simplifying, over-objectifying and over-formalizing a context model may pose risks of user frustration and disorientation.

Human behavior encompasses both apparent human behavior and the hidden mental states behind behavioral performance. The mental models of human activity are very complex and contextual cognition varies widely among individuals due to knowledge,
experience and preference. It is important to understand user specific needs and examine situational facts in the process of system activity [104]. The need to reflect mental diversity and complexity in ubiquitous computing suggests that individual cognitive views need to be identified and described in the process of context modeling. Ideally, all relevant facts and all individual variances should be identified with system specification and used for guiding the system’s responsive action.

The Cognitive Context Model (CCM) developed in this thesis allows individual or community participants of a system activity to specify their situational considerations and preferences that form the individual mental models of a ubiquitous computing activity. The individual mental model is termed a context view. A context view usually corresponds to an individual or a group. CCM embodies the interface between personal knowledge of context-aware activities and the shared beliefs of all participants in ubiquitous computing. The collection of context views allows system designers to analyze the cognitive variance of context-aware behavior, and provides them the knowledge base for deciding system domain and designing context-aware adaptation rules.

Each context view does not necessarily contain all relevant facts. For each context view, the five W’s (who, what, when, where, why/how) are used as the basics of information-gathering [5]. The why and how questions are directly related to recognizing specific cognitive status. Participants need to answer each question to elicit relevant facts which significantly influence human/system activity. The collection of all 5W answers forms a detailed view interpreting human/system activities. Each context view in CCM has a structural representation, containing facts, interrelations between facts, as well as the weighting of each fact. As contextual facts are usually not isolated, depicting interrelationships between facts improves the expressiveness of the model, making it more understandable. Compared to the form of questionnaire, this structure can facilitate information querying, storage and management.

A context view is an abstracted representation of context space developed indepen-
dently by an individual, normally reflecting a subject of interest or focus of attention. The acknowledgement of contextual facts and the measurement of their correlations are subject to individual experience. Each context view normally provides only a biased or incomplete context space. Since explicit models are always susceptible to incorporating too many or too few facts about the situation, the inclusion of diverse individual views into context modeling will help emphasizing the fuzzy and fluid aspect of context. Through the collection of a set of independent views, the knowledge base of contextual information is formed in CCM and it allows analysts to interrogate and manipulate the information to produce a consistent and optimized context space for the system.

1.3 Context Space Optimization

A context model informs both recognition and mapping by providing a structured, unified view of the world in which the system operates [19]. According to John Dewey’s theory [22], context has two components: background and selective interest. Background refers to the objective facts which are both relevant and ubiquitous in all system/human activities. There is, in principle, no limit to relevant facts.

Humans are quite successful at using background information and reacting appropriately. The context views capture the diversity and dynamics in human cognition and provide input from people for user-centric system design. However, computers are not currently well enabled to take full advantage of the context of human-computer interaction. Ubiquitous computing requires a consistent view of the context domain for decision making and task processing. In ubiquitous computing, the identification of an “appropriate” context space, which defines a scope of “interesting” contexts for monitoring, is not only subject to subjective interests but also conditioned by physical/technical constraints.

Existing context modeling techniques do not explicitly address the importance of
balancing user requirements and technical constraints to achieve maximized user satisfaction. Such techniques aim at providing a unified infrastructure to facilitate contextual information management, but depend mainly on system designers to decide the domain of context, which is typically the location, identity and state of people, groups and computational and physical objects [24].

Since human activities are dynamic and complex in nature, limited physical and computation resources and the objective of system interacting smoothly with users are usually conflicting. Trade-offs exist in deciding the contextual domain for a ubiquitous system. The process of choosing the best or the most “appropriate” context space from some set of alternatives can be interpreted as an optimization problem.

A set of context views define a limited number of objective facts. This makes it possible to apply optimization techniques on a variety of context views to produce an optimal context space from a set of context views. Among several optimization techniques, this thesis introduces the application of AHP algorithm to calculate an optimized score for each object context. The output, i.e., the optimized context view, which integrates objective surroundings and cognitive information, can thus be used by a system analyst to determine the context space under the restriction of computational resources.

1.4 Contributions of the Thesis

The following summarizes the contributions of this thesis:

- Through a review of theories in human context-aware activity and the goal-based analysis of context-aware computing, this thesis demonstrates that independent modeling of individual context views and explicit analysis of cognitive variations will result in a more precise specification of context space and context-aware activities. Diverse human cognition can form a knowledge base for system analysts to observe and deal with the trade-offs and conflicts in social and technical con-
straints. Clustering and comparison of various context views will reveal patterns and assumptions hidden in explicit human activities. Furthermore, with a context model in which individual perspectives can be traced, the responsive system actions of a ubiquitous system can be more reliable and user-friendly.

• The thesis presents a systematic framework for the context modeling process, which provides guidance for identifying individual context views of the world in which an activity is situated, analyzing cognitive variances among them, and optimizing the context space of a ubiquitous task in terms of cognitive and technical trade-offs. The context views are represented as *cognitive contexts*, and a collection of contextual facts are represented as *objective contexts*. Cognitive context, which normally arises from previous experience, cognitive ability, and the subjective role in the activity, emphasizes the fuzzy aspect of context information. The novelty of this work lies in the support of explicit analysis of cognitive variance in context-aware activities, and in the integration of cognitive contexts and objective contexts into a unified context model – *Cognitive Context Model (CCM)* – which provides a formatted scheme for context description and analysis.

• In addition to a context model for context elicitation and specification, automated tools are developed and form an environment in which the contextual knowledge collected can be organized, manipulated and interrogated. The support environment comprises a knowledge base which contains all the gathered information, an analysis engine with the operations which might be carried out on the knowledge base, and an inference engine which assists in the refinement of the knowledge and generation of context-aware adaptation rules. Organizing contextual information as a knowledge base allows context views to be continually added to a context model. The environment supports detailed tracing and recording of dependencies throughout the knowledge base and the inference process.
Throughout the thesis, examples from the domain of smart meeting environment, which has been a topic of great interest in context-aware ubiquitous computing, are used to illustrate various aspects of our work, which makes the characterization of the problem and our methodology easy to understand. This thesis also uses two case studies, i.e., computer energy saving and green transportation, to further evaluate the CCM framework and supporting tools.

The thesis identifies five criteria for high-quality context modeling: 1) context space coverage, i.e., the ability to elicit contextual facts; 2) visualization, i.e., the ability to give good insights and good design choices to system designers; 3) scalability, i.e., the ability to handle complex and massive context information; 4) traceability, i.e., the ability to link context specification back to individual views and forward to context-aware rules; 5) flexibility, i.e., the ability to allow flexible adjustments. These criteria have been applied to model evaluation in case studies.

Various aspects of the thesis have been described in papers published elsewhere. The analysis of human intention and how it affects human activities (§ 3.3) was introduced at Ubicomp 2007 Workshop on Attention Management in Ubiquitous Computing Environments [76]. The characterization of the problem which motivates our work (§ 3.4) was outlined at the ICSE’07 Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments [77]. The framework for the cognitive context modeling process (§ 4.1) was presented at Pervasive 2007 Doctoral Colloquium [75]. A case study on eliciting and analyzing contexts for smart power management (§ 6.2) was shown at the 9th International Conference on Cognitive Modeling [78] (short paper).

1.5 Overview of the Thesis

The next chapter reviews existing work in software engineering and ubiquitous computing, with the conclusion that most of existing context models concentrate on the
computational representations of contexts which can be tracked and recorded, but ignore
cognitive properties that are essential to human activities and decisions. This observation
is used as pretext for an analytical review of fields from the social and cognitive science
which cover aspects of cognitive variance and situated actions.

Chapter 3 provides goal-based analysis of scenarios in context-aware environment and
presents a technique for measuring the gap between the user’s expectation and a system’s
context-awareness design. The goal-based analysis leads to the insight that there is the
need to address in context-aware environment the social dependencies and individual
variance. Since context-aware computing will have a very open domain of individual and
collective user goals that must be interpreted dynamically, a context model, which allows
explicit description of individual views and contextual concerns, is required for exploring
decision space, alternatives and trade-offs.

The remaining chapters describe the framework of cognitive context modeling which
elicits and manages diverse context views of individual participants in a task. Chapter
4 explains the ideas underlying the framework and describes the architecture and com-
ponents of CCM, a scheme for structural representation of a task and its objective and
cognitive contexts.

CCM captures a wealth of cognitive information of context-aware activities. Chapter
5 presents the statistical techniques from related areas for manipulating and interrogat-
ing the contextual knowledge collected in CCM. It demonstrates that the abstraction
and structural description of individual views allows the techniques of statistical data
analysis, such as clustering and optimization, being applied to facilitate context space
identification. Chapter 5 also gives a formal definition of context-aware action with policy
specification language and shows that an optimized context view describing the context
space can assist context-awareness design and conflict resolution.

Chapter 6 describes a toolkit that assists the whole process of CCM model build-
ing, visualization, storage and analysis. The toolkit comprises a knowledge base which
contains all the context information elicited from individuals, and provides tools for the analysis of cognitive variance and the refinement of context space.

Chapter 7 demonstrates the CCM framework’s capability in extracting diverse cognitive contexts and assisting subsequent context-aware system design within the scenarios of smart meeting room and computer power management. The criteria of coverage, visualization, scalability, traceability and flexibility are applied to the case studies to facilitate qualitative evaluation and comparison with other context modeling techniques.

Chapter 8 presents the conclusions. The advantages of cognitive context modeling and the remaining problems of the CCM framework are discussed, along with plans for future work.
Chapter 2

Literature Review

This chapter analyzes the literature in related areas. It begins by surveying existing definitions of context and enumeration of context elements (§ 2.1). Then, it reviews three principle theories for context analysis in social and cognitive science(§ 2.2). Current work in context-aware computing, especially approaches to the elicitation and representation of context information, is reviewed and analyzed in the following sections (§ 2.3, § 2.5). The remainder of the chapter surveys work in related fields that measure the performance of context modeling.

2.1 Context

Context-aware systems offer entirely new opportunities for both software developers and users. However, it is a challenging task to even define the word “context”. This section surveys various definitions of context and dimensions of context space in ubiquitous computing.
2.1.1 Definitions of Context

Although the term context has been widely used in computer science since the term context-aware computing was first introduced by Schilit and Theimer in 1994 [98], the definition of context has experienced an evolution in the area of context-aware computing.

Schilit et al. claimed that the important aspects of context are: where you are, who you are with, and what resources are nearby [98]. They defined context as “location, identities of nearby people and objects, and changes to those objects”. Different from Schilit’s definition of context, which focuses on the enumeration of context elements, Hull et al. characterized context using synonyms. Based on the concept of situated computing, they describe context as “the aspects of the current situation” [50].

Addressing the notion and operational definition of context, Dey provided a more accurate description, which is probably the most widely accepted: “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” [24].

The majority of existing definitions in the research area of ubiquitous computing are driven by the ease of implementation, emphasizing objective features that can be tracked and recorded relatively easily and deemphasizing or avoiding aspects of the user experience such as subjectively perceived features and the way past experience of similar context may influence current activity.

With the active involvement of users in context-aware software design process, subjective aspects have been highlighted in recent definitions of context. John Dewey, with the perspective of psychology, gave a comprehensive explanation of context. According to Dewey [22], context has two components: 1) background, which is both spatial and temporal and is ubiquitous in all thinking; 2) selective interest, which conditions the subject matter of thinking. The background is that part of context that “does not come into explicit purview, does not come into question; it is taken for granted".
Viewing context as a type of domain knowledge and distinguishing contextual knowledge to external knowledge, Brezilon et al. [10] defined context as “that which constrains a focus for an actor without intervening in it explicitly”. They specified three main elements justifying this definition: 1) context is relative to the focus; 2) as the focus evolves, its context evolves too, and 3) context is highly domain-dependent.

Based on the theory of embodied action, Dourish [25] argues that the importance of context is not what it is but what it does in interaction - the role that it plays and the ways in which it is sustained and managed. Therefore, instead of assuming some stable separation between the “context” and “content” of activities that people might perform, he refers to context as something being continually renegotiated and defined in the course of action, and through this negotiation, the actions that individuals undertake can become intelligible and meaningful to each other. In a word, context and activity are mutually constitutive.

2.1.2 Dimensions of Context

Context is complex and dynamic. Nearly all context definitions are intended to be adequately general to describe the essence of context-aware activities. However, each of the provided definitions requires a considerable amount of expert knowledge to further constrain its universality. Clustering and structuring context information facilitates the engineering of a context model for context-aware activity and is vital for context-aware system design.

Dey et al. separate context elements with the statement that “Context is typically the location, identity and state of people, groups, and computational and physical objects” [23]. Abowd and Mynatt argue that the five W’s of context are a good minimal set of necessary context [5]:

- **Who**: It is important to incorporate into systems not only the identity of the particular user but also information about other people in the environment.
• What: Interaction with continuously worn, context-driven devices will likely need to incorporate interpretations of human activity to be able to provide useful information.

• Where: In many ways, the \textit{where} component of context has been explored more than the others. Of particular interest is coupling notions of \textit{where} with other contextual information, such as \textit{when}.

• When: With the exception of using time as an index into a captured record or summarizing how long a person has been at a particular location, most context-driven applications are unaware of the passage of time. Of particular interest is understanding relative changes in time as an aid for interpreting human activity.

• Why: Even more challenging than perceiving \textit{what} a person is doing is understanding \textit{why} that person is doing it. Sensing other forms of contextual information that could give an understanding of a person’s affective state, such as body temperature, heart rate, and galvanic skin response, may be a useful place to start.

In the field of modeling and reasoning within background knowledge, Lenat suggests to concretely divide context-space into 12 (classes of) dimensions, among which 4 dimensions refer to spatial-temporal issues and the remaining eight dimensions relate to human intent \cite{61}:

• Absolute Time: a particular time interval in which events occur;
• Type Of Time: a non-absolute type of time period, such as just after eating;
• Absolute Place: a particular location where events occur, such as Paris;
• Type Of Place: a non-absolute type of place, such as in bed;
• Culture: linguistic, religious, ethnic, age-group, wealth, etc. of typical actors;
• Sophistication/Security: who already knows this, who could learn it, etc.;
• Topic/Usage: drilling down into aspects and applications not subsets;
• Granularity: phenomena and details which are (and are not) ignored;
• Modality/Disposition/Epistemology: who wants/believes this content to be true;
• Argument-Preference: local rules for how to resolve pro-con argument disputes;
• Justification: are things in this context generally proven, observed, on faith...;
• Lets: local bindings of variables etc. that hold true in that context.

To bridge the user-developer gap for the creation of user-friendly context-aware software, Zimmermann et al. extend Dey’s definition of context and argue that “any information describing an entity’s context falls into one of five categories for context information: individuality, activity, location, time, and relations” [124]. The structure of context space is shown in Fig. 2.1. The individuality category contains properties and attributes describing the entity itself; the activity category covers all tasks this entity may be involved in; location and time provide the spatial-temporal coordinates of the respective entity; and relation represents information about any possible relation the entity may establish with another entity.

Figure 2.1: Zimmerman’s five fundamental categories of context information

2.1.3 Summary

The majority of context definitions are driven by the ease of implementation. An understanding of context will enable application designers to choose what context to use and
what context-aware behavior to support in their applications[24]. Existing definitions of context fall into two classes: enumeration of context elements and characterization of context using synonyms. Generally, context can be characterized by the following features:

- **Scope and extensibility.** Context refers to the surroundings or setting of something that we are interested. It is a subset of the environment, which connects to the “content”. The scope of the context is not fixed, it can be adjusted to include more or less knowledge into the reasoning process.

- **Relevance and imprecision.** Context describes the surrounding facts that add meaning. Context is not a reified entity that can be reasoned about, i.e. knowing B in context A is not transferable into statements like $A \rightarrow B$. Rather the power of context comes from its combination of “fuzzy”, fluid context identity and crisp, relatively simple context “contents”.

- **Separation and Interrelationship.** Context refers to the relatively constant factors that are separable from the relevant entity. The benefit of recognizing context relies on the fact that we can learn a simple model in one circumstance and successfully use it in another circumstance that is sufficiently similar to the first(i.e. in the same “context”).

The context of an entity can be interpreted as a point in a multi-dimensional context space. The categorization of context elements into reasonable dimensions can facilitate the engineering of a context model for context-aware application. The five W’s provide an intuitive and effective schema for dividing context space. Most categorization approaches conform to the five-W division, though the content of each W varies slightly among them. For example, Abowd’s definition of who refers to people, while in both Dey’s and Zimmermann’s definition, it also includes physical objects. Diversity also exists in the description of why context. Abowd et al. describe the why category of context as a
person’s affective state, which, in most other categorization schemes, is defined as a part of a *who* element. Lenat highlights human intent in categories of context information. Zimmermann et al. use a special category of context – *relation* – to represent the dependance between context elements. Both context relation and human intent can be viewed as implicit *why* facts which condition the context selection.

## 2.2 Context-aware Human Activity

The use of context exists in almost every area of our thinking and action, e.g., language understanding, memory, social cognition, and problem solving and reasoning, and hence the idea of context plays a fundamental role in the forms of social and cognitive analysis. This section gives a brief review of three principle theories for context analysis in social and cognitive science: *Activity theory, Distributed Cognition* and *Situated Action*, and discusses the merits and limitations of each approach to the study of context.

### 2.2.1 Situated Action

In the area of ethnomethodology, an analytic approach put forward by Lucy Suchman is typical. She introduced the term *situated action* as a way to reformulate the problem of purposeful action. “The aim of research, according to this approach, is not to produce formal models of knowledge and action, but to explore the relation of knowledge and action to the particular circumstances in which knowing and acting invariably occur” [104].

Rather than attempting to abstract action away from its circumstances and represent it as a rational plan, situated action posits the notion that people’s behavior is contextualized, i.e. the situation is a very important factor in determining what people will do. Every course of action is highly dependent upon its material and social circumstances focusing on moment-by-moment interactions between actors, and between actors and the
environments of their action [104].

Situated action emphasizes the responsiveness to the environment and the emergent, contingent nature of human activity. People often have plans of action mapped out in their heads, but may need to change that plan depending on what is actually happening in a specific situation. They use their embodied skills or past experiences to get them through the situation. The situation is significant in affecting detailed plans of action before people perform a task. And the plan ends up changing only when the person is actually performing the task. Suchman believes that people construct their plan as they go along in the situation, creating and altering their next move based on what has just happened. People can attempt to make a plan, but their situation will ultimately determine what actual plan of action they make.

Unit of analysis

The basic unit of analysis for situated action is “the activity of persons acting in setting” [60], which forces the analyst to pay attention to the flux of ongoing activity, to focus on the unfolding of real activity in a real setting. A setting is defined as “a relation between acting persons and the arenas in relation with which they act”. An arena is a stable institutional framework. A supermarket is an arena in which activity takes place, while for an individual shopper, the supermarket is experienced as a setting because each shopper shops only for certain items in certain aisles, depending on her needs and habits.

While not denying the patterns of activities across situations, situated action recognizes the opportunistic, flexible way that people engage in real activity. Situated action models emphasize on that which is emergent, contingent, improvisatory. Persistent, durable structures that are routine and predictable and spanning situations are not central.
Description of context

In situated action, every activity is by definition uniquely constituted by the confluence of the particular factors that come together to form one situation. Situated action models have a slightly behavioristic undercurrent in that it is the subject’s reaction to the environment (the situation) that finally determines action.

In situated action, one activity cannot be distinguished from another by reference to an object (motive); in fact Lave [60] argues that “goals are not a condition for action”. In other words, goals are constructed, often in verbal interpretation about why we did something after we have done it; goals are “retrospective and reflexive”. Suchman [104] and Lave [60] regard goals and plans as post hoc rationalizations for actions whose meaning can arise only within the immediacy of a given situation.

Methodology

Suchman and Trigg (1991) reports four ways of conducting situated action analysis [84]:

- a stationary video camera to record behavior and conversation;
- shadowing or following around an individual to study his or her movements;
- tracing of artifacts and instrumenting of computers to audit usage;
- event-based analysis tracking individual tasks at different locations in a given setting.

2.2.2 Activity Theory

Activity Theory is a cultural-historical theory of activity initiated by S.L.Rubinstein and A.N.Leontiev in the early 20th century. It seeks to explain social and cultural work practices by relating them to the cultural and historic context in which the work activity is taking place. The rapid expansion in information technology and subsequently in HCI has lead to the focus on the end users. Designers often have difficulties in identifying
the user’s problems or the complex organizational conditions of the implementation of new technology, where the user’s needs are the main focus. Bodker [8], Engestrom [33] and Nardi [84] brought Activity Theory into Human Computer Interaction (HCI) and Computer Supported Cooperative Work (CSCW) to provide a framework to study the development of activity systems and reinterpret the concept of user needs.

The adoption of Activity Theory in HCI is a set of basic principles that constitute a general conceptual system [54]. The following are some of the most important principles:

- **Hierarchical structure of activity.** The unit of analysis is an activity directed at an object which motivates activity, giving it a specific direction. Activities, actions and operations make up the three levels of activity.

- **Object-orientedness.** The principle of object-orientedness states that human beings live in a reality that is objective in a broad sense: the things that constitute this reality have not only the properties that are considered objective according to natural sciences but socially/culturally defined properties as well.

- **Internalization/externalization.** Activity Theory differentiates between internal and external activities. Internal activities (e.g., mental simulations, imaginings, considering alternative plans, etc.) allow people to try interactions with reality without performing actual manipulation with real objects. Internal activities cannot be understood if they are studied separately from external activities, because they transform into each other.

- **Mediation.** Activity Theory emphasizes that human activity is mediated by tools. Tools are created and transformed during the development of the activity itself and carry with them a particular culture - historical remains from their development.

- **Developing.** Activity theory considers activities not as given or static but as dynamic. Activities are always changing and developing. Development is taking pace at all levels.
Chapter 2. Literature Review

Figure 2.2: The basic activity model [58]

Figure 2.3: Engestrom’s activity model

Unit of analysis

The basic unit of analysis as proposed by Activity theory is an activity. An activity is a form of doing directed to an object. Transforming the object into an outcome motivates the existence of an activity. The relationship between the subject and the object of activity is mediated by a tool (Figure 2.2).

Considering the social and communal relations between an individual and her environment in an activity, a systemic model (Figure 2.3) is proposed by Engestrom [32] to explain collective activities and cooperative work. An activity is undertaken by a subject who is motivated toward the solution of an object and mediated by tools in collaboration with the community, those who share the same object.

Leontiev [62] proposed three levels in an activity: the activity level, the action level and the operation level (Figure 2.4). Activities consist of actions or chains of actions, which in turn consist of operations. Activities are differentiated based on their motives. Actions are goal-directed processes that must be undertaken to fulfill the object. They are
conscious and different actions may be undertaken to meet the same goal. Operations are ways of executing actions. Operations depend on the conditions under which the action is being carried out. Operations, which may start out as conscious actions, may become routinized and unconscious with practice.

Activity theory holds that the constituents of activity are not fixed but can dynamically change as conditions change. All levels can move both up and down. Thus operations over time can become unconscious actions. Also the line between action and activity is difficult to define as goals and motives can often overlap or be used interchangeably.

**Description of context**

Activity theory proposes that the activity itself is the context. Context is constituted through the enactment of an activity involving people and artifacts. Context is not just a set of external resources, it is both internal and external to people, involving artifacts, other people, settings, specific objects and goals. And the external and internal are fused, unified.

Activity theory emphasizes motivation and purposefulness. In activity theory, activity is shaped first and foremost by an object held by the subject. Activity theory is also concerned with the historical development of activity and the mediating role of artifacts.
Methodology

Engestrom [33] notes that activity theory does not offer ready-made techniques and procedure for research; rather, it is a conceptual tool. These conceptual tools must be adapted to the specific nature of the object being studied. Nardi [84] describes four methodological consideration for activity theory:

- Allow for research time frames long enough to understand user’s objects. The changes over time in objects and their relationships must be studied;
- Pay attention to broad patterns of activity rather than narrow episodic ones that fail to reveal the overall direction and import of an activity;
- Use varied sets of data collection techniques including interviews, observations, video, and historical materials, without undue reliance on any one method;
- A commitment to understanding things from the user’s viewpoint - user centring the inquiry process.

2.2.3 Distributed Cognition

Computation is conceived broadly as “the propagation of representational state across representational media” [51]. Traditional cognitive science emphasizes an internalism that ignores the role of external representation and problem solving in cooperative contexts. Motivated by the basic insight that cognition is a socially (also, materially and temporally) distributed phenomenon, the theory of distributed cognition [51] has been used to study the representation, propagation and transformations of knowledge between individuals and artifacts.

Distributed cognition does not do away with the notion of individual cognition, but the focus is on how cognition is distributed across people and artifacts, and on how it depends on both internal and external representations. It argues that cognitive processes generally are best understood as situated in and distributed across concrete socio-
technical contexts. “The ethnography of distributed cognitive systems retains an interest in individual minds, but adds to that a focus on the material and social means of the construction of action and meaning. It situates meaning in negotiated social practices, and attends to the meanings of silence and the absence of action in context as well as to words and actions.” [47]. In a socially distributed system, people interact with artifacts to create and coordinate representations. Thus, representational state can be “propagated” across distinct media and across different representational systems.

**Unit of analysis**

Traditional cognitive theory takes the individual person as the unit of analysis. On this traditional view, cognitive processes are internal processes. Distributed cognition employs as a unit of analysis a cognitive system composed of individuals and the artifacts they use, i.e., the cooperating people and artifacts are the focus of interest, not just individual cognition in the head. Distributed cognition is concerned with understanding the coordination among individuals and artifacts, that is, to understand how individual agents align and share within a distributed process.

The theory of distributed cognition does not prevent the individual from being the unit of analysis. The larger cognitive systems can be decomposed into subsystems that can be similarly decomposed. Individual cognition should be analyzed as a process that is distributed among the important functional components of the brain. Enlarging the unit of analysis in this way has the benefit that representations internal to the system are now ”external” representations with respect to the individual agents that use and make use of them. Thus, cognitive processes are fully observable.

**Description of context**

A distributed cognition analysis begins with the positing of a system goal, which is similar to the activity theory notion of object, except that a system goal is an abstract systemic
Table 2.1: The comparison of three theories on context study

<table>
<thead>
<tr>
<th></th>
<th>Situated Action</th>
<th>Activity Theory</th>
<th>Distributed Cognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of Analysis</td>
<td>Persons &amp; setting</td>
<td>An activity</td>
<td>Persons &amp; artifacts</td>
</tr>
<tr>
<td>Role of Human</td>
<td>Reactive</td>
<td>Consciousness</td>
<td>Computation agent</td>
</tr>
<tr>
<td>Desc. of Context</td>
<td>Situation</td>
<td>Activities</td>
<td>Artifacts</td>
</tr>
<tr>
<td>Methodology</td>
<td>Observation</td>
<td>Observation &amp; Inquiry</td>
<td>Inquiry</td>
</tr>
</tbody>
</table>

concept that does not involve individual consciousness. However, artifacts designed by others, sharing ideas across time and space is a central focus in distributed cognition.

Methodology

For research in the area of socially distributed systems, an especially important method is a new kind of cognitive ethnographic study [47]. “The ethnography of distributed cognitive systems retains an interest in individual minds, but adds to that a focus on the material and social means of the construction of action and meaning. It situates meaning in negotiated social practices, and attends to the meanings of silence and the absence of action in context as well as to words and actions.” Then, inquiry conducted by experts should be the best way to collecting data for a distributed cognition analysis.

2.2.4 Summary

Situated action, activity theory and distributed cognition are three principle cognitive theories which are valuable in underscoring the need to look at real activity in real situations for the study of context-aware human activity. Table 2.1 gives a brief summary of the key ideas of each theory. Although three approaches describe and analyze contexts and activities in different ways, all of them to the study of context have merit.

Situated Action suggests the importance of understanding each user or organization’s
specific needs and examining situational and organizational factors in the process of tasks. Situated action views an activity as being uniquely constituted by a “situation”.

In both activity theory and distributed cognition, an object or goal is the beginning point of analysis, i.e., an object precedes and motivates activity. Activity theory emphasizes on the importance of human motive and consciousness and views artifacts as mediators of human thought and behavior. The conceptual framework of Activity theory has been seen as a way of providing a means of analyzing the actions and interactions with artifacts within a historical and cultural context [90].

Distributed cognition differs from activity theory in that a system goal is an abstract systemic concept that does not involve individual consciousness. In distributed cognition, people and artifacts are agents in a system; persistent structures, especially artifacts, are the focus of study. The theory can accommodate the rich variety of representational media and systems that in fact implement a group or organization’s cognitive processes. It helps to reveal constraints that are implied by the embodied nature of the representational media that are inevitably employed in carrying out a given task.

Overall, the flexibility and contextualization of activity is an important feature that distinguishes situated action, activity theory, and distributed cognition from traditional cognitive theories and approaches. However, none of these theories provides a methodology that one can readily pick off the shelf and apply to a design problem [92]. Situated action models ignore the subjective and do not account very well for observed regularities and durable, stable phenomena that span individual situations [84]. The most important problem for distributed cognition seems to be with ethnographic data collection, a methodology to which this theoretical perspective is admittedly committed [47]. Activity theory does not provide a scheme for system designers to distinguish between “action” and “activity”. In certain complex organizational contexts what is an activity to one person or role may be an action to another. Furthermore, since the levels evolve and move up and down, it would make a designer’s view over time unable to track and understand
these movements.

2.3 Context Modeling & Context-aware Computing

It has become increasingly clear that context-dependent thought is an essential tool for enabling effective learning, reasoning and communication in a complex world. With the emergence and wide usage of mobile and embedded computing devices and with the development in wireless networking, it is possible for systems to sense their use of context and adapt their behavior accordingly. The inclusion of context-awareness in computing provides convenience and efficiency to users for their ubiquitous access.

Structural and formal representation of context usually play a vital role in context-aware applications to support context reasoning and to guide responsive action. A well designed model can facilitate knowledge sharing and reasoning. This section begins by summarizing the concept and categories of context-aware computing. Then it gives an overview of current context modeling approaches.

2.3.1 Context-aware Computing

Context-aware computing was first discussed by Schilit and Theimer in 1994 as software that adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time [98]. Dey et al. suggest that “a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task” [24]. In general, context-aware computing refers to the computational systems that can sense and respond to aspects of the settings in which they are used [25].

A motivation behind context-aware computing is to find ways of compensating for limitations in human cognition, e.g., attention, memory, learning, comprehension, and decision-making, through the use of sensor-based and computational tools [91]. Context-
aware applications fall into three major categories [24]:

- presentation of information and services to a user;
- automatic execution of a service for a user;
- tagging of context to information to support later retrieval.

The *Active Badge Location System* developed by Want et al. [114] is considered to be one of the first context-aware applications. The infrared-technology-based system was able to determine a user’s current location which was used to forward phone calls to a telephone close to the user. Location information is by far the most frequently used attribute of context. Several projects emerged in the middle of the 1990s which provided information according to the user’s current location, such as GPS-based car navigation systems, the location-aware tour guides Cyberguide [4] and GUIDE [15].

The combination of various sources of context elements, e.g., time, roles of human, and device states, have been explored in a couple of recent projects to initiate responsive reactions. Aiming at developing an extensible reactive system that learns from users interactions and evolves with time, the EasyLiving project [13] from Microsoft designed a behavior engine that polls a world model database, examines a set of context variables in order to decide what actions to take, and then issues commands to software agents (devices and services). MIT’s iRoom project [11, 18] also provided several mechanisms for inducing reactivity, such as hard coding the behavior logic in the platform itself, using GUI tools that aid users in dynamically inducing reactivity and identifying appropriate reactions based on observing users’ interaction in a ubiquitous space.

A variety of services and devices are usually available in pervasive environments and there are trade-offs among various tasks or performance dimensions, which implies the existence of several alternatives to accomplish certain tasks. Therefore, user preferences are sometimes required for selecting the appropriate way of execution. Instead of providing content adaptation based on the target device alone, W.Y.Lum et. al [72] adopted the concept of quality of service (QoS), using a score to quantify and measure user satis-
faction along different quality axes, such as color, transmission time, etc.; and designed a decision logic to select the appropriate trade off among the various quality dimensions corresponding to a content version that best matches the user’s preference and is deemed renderable by the device. The Gaia project at UIUC presented a paradigm for adapting to changing contexts and availability of resources based on goal specification and AI planning techniques [94, 89].

Context-aware systems offer entirely new opportunities for application developers and for users by gathering context data and adapting systems behavior accordingly.

2.3.2 Object-based Context Models

Models of objects are the most simple and efficient structure for representing contextual information. Context-aware services usually employ attribute-value tuples to describe situational attributes, and then operate an exact matching algorithm on these attributes. Most early-stage context models are object-based, though they vary in the scheme of data structures which are used to store and exchange contextual information.

Schilit et al. [98] used key-value pairs to model the context by providing the value of a context information to an application as an environment variable. In this work, context information is primarily location. Active Map objects gather context information related to a physical spatial area and make it available to client applications. This approach assumes that a location can be assigned to all context information.

Markup schemes are also widely used for describing context information by providing a hierarchical data structure consisting of markup tags with attributes and content. Typically, context is represented by profiles. CC/PP (Composite Capabilities / Preference Profile) [111] and UAprof (User Agent Profile) [36] are two standards for describing delivery information about device capabilities and user preference for WAP based mobile devices.

Dey’s Context Toolkit [23] is probably the earliest attempt at representing various
context elements with key/value-based objects. They introduce the concept of context widgets which mediate between the environment and the application in the same way graphical widgets mediate between the user and the application. Context widgets are basic building blocks that manage sensing of a particular piece of context (e.g. presence, identity and activity of people and things) [96].

2.3.3 Logic-based Context Models

Logic provides a high degree of formality and the main focus of logic-based approach is less on context modeling than on context reasoning. In a logic based context model, the context is consequently defined as facts, expressions and rules.

Mostéfaoui et al. [80] identified five types of context and a slice of all possible elements which are relevant in a service discovery and composition process, as shown in Figure 2.5. They model the context information as a recursive function:

\[
context(t) = f(u, c, t, ph, h),
\]

Fig. 2.5: Modeling context in context-based service composition [80]
which shows that context is function of users, computing context, the time, the physical
context, and the context history. Recursivity is used in order to capture the context
history, i.e. \( h = context(t - 1) \). And \( u = f(role, pref, soc, loc, pm) \), i.e. the user is in
turn represented by its role (student, staff ... etc), its preferences, social situations, etc.

Instead of giving a definition of what context is, McCarthy et al. [73] formalize
context as first class objects, i.e. abstract mathematical entities with properties useful
in AI reasoning. They identify two types of basic relations which relate the truth in one
context to the truth in another context:

- \( ist(c,p) \) meaning that the proposition \( p \) is true in the context \( c \), and
- \( value(c,e) \) designating the value of the term \( e \) in the context \( c \).

This allows for formulas such as \( \forall t(ist(light1(t),signal = 1) \equiv door\_open(t)) \), which
supposes the meaning of the signal is that the door of the microwave oven is open or
closed according to whether the signal on is 0 or 1.

**2.3.4 Graphical Context Models**

In both key/value-based and logic-based approaches, context elements are regarded as
independent variables. However, in reality, situational facts or entities are usually not
separated. A graph, in which nodes represent variables and pairs of nodes connected by
arcs correspond to variables that are not independent of one another, therefore exhibits
better capability for sophisticated structuring and enabling efficient context reasoning
algorithms.

Object Role Modeling (ORM) is a method for designing and querying database models
at the conceptual level, where the application is described in terms easily understood by
non-technical users [41]. The modeling of a domain using ORM involves identifying
appropriate fact types and the roles that entity types play in these. Henricksen et al. [45]
classify context facts into four types – sensed, static, profiled, derived – according to
Figure 2.6: Extended ORM for context modeling [45]

their persistence and source, and propose a context extension to ORM to allow context facts being differentiated and represented. An example of the context model is shown in Figure 2.6. Fact dependencies are captured by the dependsOn relation to represent a special type of relationship between facts, where a change in one fact leads automatically to a change in another fact.

Ontology, which is mostly used in artificial intelligence, the semantic web, software engineering and information architecture as a form of knowledge representation to structure the shared understanding of some domain of interest, is another instrument to specify concepts and interrelations [108]. An ontology describes four types of elements:

- **Individuals (instances)**: the basic, ground-level objects;
- **Classes (concepts)**: abstract groups, sets, or collections of objects;
- **Attributes**: properties, features, characteristics, or parameters that objects can have and share;
- **Relations**: ways that objects can be related to one another.

Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties,
Chapter 2. Literature Review

(1) CONON Upper Ontology  
(2) A Specific Ontology for Home Domain

Figure 2.7: CONON for modeling context in pervasive Environments [112]

and relations. There are a number of such languages for ontologies, such as OWL(Web Ontology Knowledge), KIF(Knowledge Interchange Format) and CycL.

Believing that Web ontology and other Semantic Web technologies can also be employed in modeling and reasoning about context information in pervasive computing environments, Wang et al. propose an OWL encoded context ontology (CONON) for modeling context and supporting logic-based context reasoning [112]. Figure 2.7 shows an example of context ontology defined by CONON. The context model is divided into upper ontology(Figure 2.7(1)) which captures general features of basic contextual entities, and domain specific ontology(Figure 2.7(2)) which defines the details of general concepts and their features in each subdomain. In this model, location, user, activity and computational entity are identified as the four most fundamental context elements in the executing situation.

Another approach within the ontology category is the Aspect-Scale-Context Information (ASC) model proposed by Strang et al. as [103]. The model is implemented by applying selected ontology languages. The implementation builds up the core of a non monolithic Context Ontology Language (CoOL), which provides an uniform way for...
specifying the model’s core concepts as well as an arbitrary amount of subconcepts and facts, altogether enabling context-awareness and contextual interoperability during service discovery and execution in distributed systems.

2.3.5 Summary

Context-aware systems typically operate in highly dynamic environments and are required to interact smoothly with their changing environments. Such systems are usually implemented with models of the surroundings so that they are aware of changes in their environments and adjust their behavior accordingly. The main focus of these models is to facilitate context representation, context sharing and reasoning.

Current context models in ubiquitous computing can be classified into three categories: attribute-value tuples, logic-based models and graphical models. Logic-based context models exhibit the highest level of formality, which facilitates data exchange and interpretation in context-aware computing. However, the specification of contextual knowledge within a logic model is very error-prone without formal validation and verification. Both logic models and object-based context models are easy to manage in context-aware applications, but lack capabilities for sophisticated structuring for enabling efficient context retrieval algorithms. Graphic models, e.g. ORM and Ontology, exhibit strength in describing the structure of contextual knowledge. The extended ORM identifies static, dynamic, dependency, and quality facts that rely on context information and captures the roles that entities play in these. It can be applied to derive ER models for information management. Ontology-based context models focus on shared understanding in a given subject area and allow automated reasoning using ontology tools.

While all context models are useful for context understanding, sharing and reasoning in context-aware computing, different modeling approaches seem to address different aspects of context information. Table 2.2 compares the context models according to the level of formality, richness of information, the capability to handle imperfect, dynamic
Table 2.2: The comparison of different approaches to context modeling

<table>
<thead>
<tr>
<th></th>
<th>Formality</th>
<th>Richness</th>
<th>Evolution</th>
<th>Imperfection</th>
<th>Dynamics</th>
<th>Cooperation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-value</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Profile</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Widget</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Logic</td>
<td>⊕</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>ORM</td>
<td>○</td>
<td>⊕</td>
<td>○</td>
<td>⊕</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Ontology</td>
<td>⊕</td>
<td>⊕</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

○ low  ○ medium  ⊕ high

and continually evolving contextual data, and the ability to identify social collaboration in context-aware activities.

2.4 Adaptation Modeling and Specification

Context-awareness and self-adaptation are among the many challenges facing the design of ubiquitous computing systems [97, 113]. These two features work together to provide the ability for a ubiquitous system to adapt at runtime to the changes in the environment and the user needs. However, research in context-awareness and self-adaptation is usually carried out separately. Context-awareness is concerned with how to capture, use and manage the environmental information, while self-adaptation is more concerned with how to reconfigure the system behavior in response to requirements changes.

There have been a number of research efforts investigating and handling the high-variability features in dynamic adaptive systems (DAS).

Liaskos et al. [65] argue that goal models can be used to capture variability in the early requirements phase before variation points of the system-to-be are defined. Based on the semantic characterization of OR-decompositions of goals and a set of pre-defined
variability concerns, they propose a variability-intensive process for goal decomposition that is tailored to support requirements identification for adaptive software. While the process aims at attaining completeness in variability acquisition, it also allows reasoning about the alternatives by applying background variability as selection criteria. The background variability is defined as attributes of three basic entity types, i.e., agent, location, and object.

In [66], Liaskos et al. propose an approach to reasoning about the alternatives by using quantitative prioritization of high-level stakeholder’s preferences as selection criteria. They distinguish between mandatory and optional goals and utilize a preference-based planner to identify alternatives of the mandatory decompositions that best satisfy the specified preferences. They found that reasoning about preferences and alternatives allows better understanding of the connection between the stakeholder attitudes and alternative behavioral designs.

Considering a DAS as a collection of steady-state systems which can be adapted through the dynamic transition from executing one steady-state system (source system) to running another steady-state system (target system), Zhang and Cheng [123] propose to adopt a model-driven approach for designing and validating dynamically adaptive software systems. They separate the system’s adaptive behavior from the non-adaptive behavior and use the state-based diagram Petri-Nets [87] to model the system’s adaptive behavior, where they use context change as guidance for the transition between the system states. This exhaustive representation allows validating intensively the system at design time. Using code generation, adaptive programs are derived from the models.

Addressing the need to handle changes to the requirements and corresponding behavior of a DAS in response to varying environmental conditions, Goldsby et al. [40] propose four levels of RE as an approach to modeling the requirements of a DAS using i* goal models [120]. Each level of RE corresponds to the work of a different type of developer (the system developer, the adaptation scenario developer, the adaptation infrastructure
developer, and the DAS research community) to construct goal mode(s) specifying their requirements, e.g., the problem domain, the stakeholder objectives, the (non-adaptive) business logic, a set of (adaptive) steady-state systems, and the adaptation infrastructure. They argue that the i* models produced could be integrated into the model-driven development of a DAS.

The high variability of features in dynamically adaptive systems introduces an exponential growth of the number of possible runtime system configurations and mode transitions. If there exists \( N \) possible configurations, this may lead to \( N(N-1) \) possible transitions; and if \( N \) is large, it rapidly becomes difficult to handle these transitions by hand [123]. Morin et al. [79] propose an aspect-oriented modeling (AOM) approach for coping with the complexity of dynamically adaptive systems. This approach combines aspect-oriented modeling and model-driven techniques to limit the number of artifacts needed to realize dynamic variability. They derive a wide range of modes by weaving aspects into an explicit model reflecting the runtime system, and use these generated modes to automatically adapt the system.

Overall, current adaptation mechanisms can be classified into two categories: 1) rule-based, which requires the designer to specify a set of rules of the form “IF (condition), THEN (actions)” [123, 79]; 2) goal-based, which uses goals to infer the required adaptation actions [65, 66, 40]. Rule-based adaption specification allows the efficient development and validation of adaptive system behaviors, while goal-based variability identification can provide a better understanding of the system domain and alternatives.

However, all the techniques mentioned above concentrate on the behavior of the adaptive systems. The environmental information are considered as a part of the adaptation mechanism (e.g. conditions, factors), and the environment values are monitored and passed directly to the adaptation engine to trigger responsive actions. None of them maintain an explicit context model to represent the complex, extensible and evolving environment. As a result, these techniques are unlikely to scale up for processing large-scale
environments where the contextual facts are complex and interrelated. Furthermore, the hard-coded adaptation strategies are not flexible to changes, making adaptive systems unable to handle unexpected context changes at runtime.

2.5 Critical Analysis

We have surveyed a number of definitions of the note “context” and current approaches for context modeling. Work on context-aware human activity is closely related to context-aware computing. An understanding of how context is used in human thought will enable application designers to choose what context-aware behavior to support in their applications.

2.5.1 Cognitive Bias

Although the term “context” has been widely used in computer science, the ability to recognize the context and determine the appropriate action requires considerable intelligence. Context-aware activity encompasses both apparent human behavior and the hidden mental states behind behavioral performance. The mental models of human activity are very complex and the contextual cognitive models vary widely among individuals due to knowledge, experience and preference [104].

Cognitive bias is common in human thought. People tend to evaluate ambiguous information in a way beneficial to their interests [37]. All the approaches to context modeling so far share a shortcoming: They all concentrate on finding their own definition for what context actually includes and developing a single description of the context. There is no support available for developing and maintaining alternative descriptions from various suggestions. Such techniques aim at providing a unified infrastructure to facilitate contextual information management, depending mainly on system designers to decide the domain of context, which is typically the location, identity and state of people,
groups and computational and physical objects [24]. The acknowledgement of contextual facts and the measurement of their correlations are subject to individual experience.

Several approaches have been proposed to address cognitive bias in the research area of decision making. Schwartz [99] describes the process of scenario planning to reflect on individual biases and assumptions and think critically about the most important factors or driving forces. Through the use of narrative strategies and story-telling techniques, scenarios allow stakeholders to walk through a structured process of uncertainty evaluation and options consideration. This approach has demonstrated repeated effectiveness at opening up a range of options up for discussion, creating a common intellectual framework (or shared mental model) and thereby producing more creative, perceptive and adaptive policies and plans.

Web- and game-based participation approaches offer a solution to encourage larger participation in the decision-making process and synthesize diverse decision perspectives. Massively multiplayer online environments are effective at motivating distributed users to socialize, interact, and execute complex challenges in dynamic and uncertain problem contexts [63]. For instance, storyboarding is effective at illustrating an envisioned scenario of how an application feature works and allows various participants depicting higher level concepts surrounding the motivation and emotion during system use [107]. Wiki-featured web tools have also demonstrated benefits on supporting collaboration in early stages of distributed requirements engineering and overcoming the limitations of spatially and temporally distributed stakeholders [71].

In the field of requirements engineering, the capture of multiple perspectives and the resolution of conflicts between them are emphasized to avoid the single viewpoint bias and to form a consistent model of the specification activity [27]. Coakes et al. [17] developed a method of modeling that appropriately reflect the conflicting and competing data and multiple perspectives of participants and stakeholders to improve the interactivity and conflict management in software system. Sutcliffe et al. [106] argue that “it is impor-
tant to analyze requirements from an individual viewpoint, especially in domains where customization is important”. They introduce the concept of individual users and contexts as a focus for requirements engineering, and propose a three-layered framework for general requirements, individual characteristics, personal goals, as well as their changes over time and location.

Humans are the essential source of information and the main target of computation in context-aware environment. Relying on a single perspective to describe complex context-aware activities may result in a biased or incomplete context space and pose risks on user frustration and disorientation. The need to reflect mental diversity and complexity in context-aware computing suggests that multiple participants of context-aware activity should be involved and all subjective views need to be explicitly identified in the process of context modeling. Every participant contributes to context elicitation, together with a number of biases. A set of individual views form a knowledge base which provides a richer picture than a single view.

2.5.2 Trade-offs Between Cognition and Computation

Context-awareness exhibited by people is radically different than that of computational systems. People notice and integrate a vast range of cues, both obvious and subtle, and interpret them in light of their previous experience to define their context. In contrast, context-aware systems detect a very small set of cues, typically quantitative variations of the dimensions for which they have sensors, and use hard coded context models or explicit adaptation rules, which provide limited learning ability [34]. There is always a conflict between the infinite, subjective detail of human activity and the finite, objective aspects of system design. Therefore, one of the key issues for developing context-aware application has been how systems can represent work and its context without over-formalizing, over-simplifying and over-objectifying it.

The importance of exploring the design space, alternatives and trade-offs has con-
cerned the field of early-phase requirements engineering [121]. The notion of goals has been adopted to provide the rationale for defining which agents should best perform which actions to fit prescribed constraints (according to their capabilities, reliability, cost, load, motivation, and so forth) [21].

Trade-off also exists in the decision of contextual domain for a context-aware system. As have been reviewed in § 2.2 and § 2.3, CSCW (Computer Supported Cooperative Work) and context-aware computing show differences in the interpretation of context and the study of context-awareness. CSCW focuses on subjective and social aspects of context, while context-aware computing often concentrates on objective features that can be tracked, recorded and represented relatively easily. Theories of sociology and philosophy, especially the ethnomethodology and phenomenology, suggest that user experience such as subjectively perceived features and the way past experience of similar context may influence current activity [25].

Computers are not currently well enabled to take full advantage of the context of human-computer interaction. Context-aware systems usually require a consistent view of the context domain for decision making and task processing. The majority of current context models are driven by the ease of system design and implementation, while lack of analysis on the trade-offs and alternatives in deciding context domain of context-aware applications. Typically, none of current context models explicitly address the importance of balancing user requirements and technical constraints to achieve maximized user satisfaction.

Applying a simplified and consistent context space to context-aware computing inevitably involves marginalizing and ignoring individual differences and special cases, leading to occasional fallibility of context-aware applications. Users of context-aware applications may find this occasional fallibility frustrating, if there is lack of transparency on what the system knows, how the system adapts, and why the system knows and adapts. Studies by Lim et al. [67] have shown that users care about why the application behaved
as it did in specific situations and how it works in general and how to produce certain actions or decisions. “Providing reasoning trace explanations for context-aware applications to novice users, and in particular Why explanations, can improve user’s understanding and trust in the system” [68].

The key question for context modeling is therefore to confront the blooming, complicated context information and still produce some generalizable results. A context model should support the selection of an “appropriate” context space, which balances subjective interests and technical constraints. With context modeling, system designers should be able to answer “why” questions regarding their choice of context spaces for monitoring and reasoning on.

2.6 Evaluation Criteria

Developing a structure that can better explore and represent the world is one concern in context modeling. Another concern is regarding the metrics: how can the performances of a context model be evaluated?

Krummenacher and Strang distinguish the main criteria specific to ontology-based context models, as described in [57]:

- **Applicability**: Dealing with very heterogeneous kinds of data, devices, etc., the adaptability of the model must be rated.

- **Comparability**: How will the model deal with diverse and non-countable information? If the model deals with, for instance, temperature values and geographic locations, it must find a way to represent them.

- **Traceability**: The origin of information can change everything, since it can bring a whole different meaning to data in different contexts, where different rules apply.

- **History**: Data should be cached in some way, since context is evolving not only in space but also in time. Decisions can depend on past events.
• **Quality**: Rating reliability, integrity, precision, resolution, etc.

• **Satisfiability**: To rate conformance of derived information to the defined model, we should be able to evaluate the range of context data.

• **Incompleteness**: The model should be able to deal with ambiguity.

Fact-based context modelling approaches, including the ORM approach described in § 2.3.4 and Context Modelling Language (CML) described by Henricksen et al. [44], originated from attempts to create sufficiently formal models of context to support query processing and reasoning, as well as to provide modelling constructs suitable for use in software engineering tasks such as analysis and design. Such approaches have their early roots in database modelling techniques. A set of criteria can be used in the evaluation of alternative data models: [74]:

• **Simplicity**: A model should have the smallest possible number of structure types, composition rules, and attributes.

• **Elegance**: A model should be as simple as possible for a given direct modeling capability.

• **Picturability**: Model structures should be displayable in pictorial form.

• **Modeling directness**: A model should have as many direct counterparts to real-world concepts as possible.

• **Modeling uniqueness**: A model should not provide equivalent direct modeling techniques.

• **Provision of schemas**: A model should include structure schemas to permit data definition.

• **Implementation independence**: A model should be free of implementation matters.

• **Overlap with co-resident models**: A model should mesh smoothly with other co-resident models.

• **Partitionability**: A model should have structures which facilitate the administrative
partitioning of data.

- **Nonconflicting terminology**: A model should use terminology which does not conflict with established terminology.

- **Proximity to implementation base**: A model should not be too far from its implementation base.

- **Applicability of safe implementation techniques**: A model should permit the use of proved well-understood implementation techniques.

Filho et al. [35] argue that the Quality of Context information (QoC) plays an important role for improving context-based adaptation processes and for ensuring the correct behavior of context-aware applications and services. A set of QoC indicators - privacy, security, precision, completeness, and resolution - are defined in their approach for measuring quality of raw, inferred, and derived context information and providing QoC-enriched context information of users to context-aware applications and services.

Strang et al. attempt to identify a set of high demands on context modeling approach from the perspective of ubiquitous computing environment and apply these demands as metrics for model evaluation [102]:

- **distributed composition**: Any ubiquitous computing system is a derivative of a distributed computing system which lacks a central entity that is responsible for the creation, deployment and maintenance of data and services.

- **partial validation**: It is highly desirable to be able to partially validate contextual knowledge on structure as well as on instance level against a context model in use.

- **richness and quality of information**: A context model should inherently support quality and richness indication.

- **incompleteness and ambiguity**: The set of contextual information available at any point in time characterizing relevant entities is usually incomplete and/or ambiguous.
• **level of formality**: It is highly desirable, that each participating party in an ubiquitous computing interaction shares the same interpretation of the data exchanged and the meaning behind it (so called shared understanding).

• **applicability to existing environments**: A context model must be applicable within existing infrastructure of ubiquitous computing environments.

Overall, current measures of context modeling emphasize the quality of contextual data and the support of context reasoning, but lack metrics for evaluating alternative approaches for context elicitation and context space decision.

### 2.7 Summary

This chapter has surveyed work from a number of relevant fields. Section 2.1 discussed various definitions of context. There are two categories of definition: enumeration of context elements and characterization of context using synonyms. These definitions play vital role in context selection and context-aware computing. The enumerative definitions identify the dimensions of context space, while the characterizing of context helps differentiating “context” and “content”.

Section 2.2 discussed work in social and cognitive science, and in particular techniques for modeling human context-aware activities. Cognitive theories, e.g. situated action, activity theory and distributed cognition, suggests the importance of examining situational and organizational factors in the process of tasks and provide conceptual frameworks to describe and analyze human actions and interactions with artifacts in historical and social context. However, none of these theories provides a methodology that one can readily pick off the shelf and apply to a design problem.

Section 2.3 reviewed the concept of context-aware computing and current context modeling approaches in this field. There have been various types of approaches for context modeling, including the early-stage object-based, logic-based models and recent
graphic models which have capabilities for capturing the richness of context information and handling the imperfection and dynamics of contextual data. However, all these methods are driven by the ease of implementation, emphasizing abstraction, automation and generalization, while marginalizing subjective diversity and evolution.

Context modeling inevitably involves deciding exactly the relevant contextual facts for discovery. The acknowledgement of contextual facts and the measurement of their correlations are subject to individual experience. All context models so far share a shortcoming: They do not provide support for developing and maintaining alternative descriptions from various suggestions. Section 2.5 described recent work which recognizes the need to handle cognitive biases in decision making and to deal with trade-offs between cognition and computation.

The last section of this chapter surveyed evaluation criteria for context models, in particular the metrics for measuring the performances of a context model. There are briefly three aspects addressed by current evaluation frameworks: the quality of contextual data, the support for automatic reasoning, and the ease of implementation. The ability to explore contextual elements and identify contextual alternatives is usually ignored.
Chapter 3

Goal-based Analysis of Existing Work on Context-aware Computing

The previous chapter has reviewed context definitions and current approaches to context modeling in related fields. This chapter focuses on the analysis of context-aware activities in context-aware environments and the evaluation of existing context-awareness design. Section 3.1 gives the description and relevant scenarios of a context-aware environment. Section 3.2 introduces the goal modeling technique. Section 3.3 analyzes the scenarios of context-aware activities using i* framework. Section 3.4 presents a technique for measuring the gap between context-aware system design and user’s expectation, and applies it to the evaluation of existing smart meeting projects. The last section summaries the analysis results and discusses the rationale behind cognitive context modeling.

3.1 A Running Example: Smart Meeting Room

In the literature of ubiquitous computing, a closed environment equipped with many devices has been a topic of great interest. Projects, such as Gaia [94], the neural house [81], the intelligent classroom [118], and responsive offices [31], were developed to monitor the environment and adapt intelligently to changing contexts and user intents. Throughout
the thesis, we’ll use scenarios of a smart meeting room, which accommodates the meetings with furniture, devices and services, to illustrate various aspects of our work. Figure 3.1 shows the layout of a small meeting room allowing 10 to 20 people in the meetings.

In a smart meeting room, people interact smoothly with devices and services to perform various tasks. The following scenarios describe three context-aware activities possibly happening in a meeting:

**S1: Scheduling a meeting**  Alex is going to present her paper at a conference in a few weeks. She’d like to give a practice talk soon and get some comments from her colleagues. Since her research group has a regular meeting 3-4pm on Wednesday each week, she prefers to having the practice talk during the meeting. So she sends out an email asking if a 20-minute practice talk after the group meeting (i.e., 4-4:20pm) works for everyone in the group. Since scheduling the talk is one of the most urgent task for her recently, she marks the email as urgent and identifies Bob as the
group meeting coordinator and a list of contacts in her address book as important attendees that their response may affect her decision for the scheduling. After doing that in her email client, she continues working on preparing and polishing the presentation slices. Two days later, when Alex turns on her computer and starts to work, a notification message appears on the screen showing that she has finally gotten feedback from all the important colleagues and a summary of the feedback is generated for her. The summary shows that the time is fine for most people except that one of her coauthors has another meeting at 4pm. At the same time, the group meeting coordinator Bob also receives the summary and a notification of the conflicts. While Alex is thinking of scheduling another time for the talk, she received a notification of important email from Bob. She checks the email immediately and is glad to know that Bob has suggested to postpone the group meeting to 3:30pm, so that all the colleagues can attend her talk at 3pm.

S2: Service collaboration in a meeting  Alex is working on a computer in her office. Her e-calendar reminds her that she should give a presentation in 20 minutes at a meeting room and it will take her 10 minutes to walk there and get ready for the presentation. Alex stops her current work immediately and indicates the files that will be used in the presentation. The files are transferred from the desktop computer to her laptop while she walks out with her laptop. At the same time, the meeting assistant system in the meeting room is informed about the presentation. The projector is automatically opened and warmed up. Lights in the meeting room are automatically adjusted according to the state of attendees, i.e., whether they are watching and listening to the presenter or they are discussing with paper or white board. There is a recording system in the meeting room which can do both audio and video recording during the meeting. Before the meeting, the system has sent an agreement to all the attendees, checking whether or not each one wants to be recorded. When everyone has taken a seat, the system recognizes each attendee’s
position and turns off the camera for him if he has not sent back the agreement. A notification is sent to each attendee’s carry-on device (e.g., laptop, PDA, cellphone) if he is being recorded, along with an instruction on the control of the camera so that he can choose to turn off the camera during the meeting. As the presentation proceeds, Alex is about to demonstrate a new tool. People show great interest and have many questions. By calculating the remaining time for the presentation, a warning message is displayed to Jane’s laptop, indicating the time limit and suggesting some of the remaining slides, which deal with the most popular questions or the contents that haven’t been covered. When the presentation is finished, the projector and lights are shut down. The record of the presentation is transferred to Alex’s desktop computer. A list of the attendees who have shown great interest and been very active during her talk is also generated with a summary of their comments.

**S3: Discussion and knowledge sharing**  
Mike and Lisa are sitting next to each other in a meeting room, while Alex is giving a talk. Alex is talking about an issue which Mike thinks very relevant to his work. He points it out and talks a little about his viewpoint of the problem. While he is talking, the link to his website of the work is sent to Alex’s laptop with a picture showing the key idea of his work. As both Alex and other attendees show great interest in his work, Alex invites him to present a few of those materials by turning over the control of projector to him. A message is sent to Mike’s laptop showing that his screen can be mapped to the projector now. He accepts the invitation. A notification warns him that he has some private information on the screen. He checks the system and chooses the public setting with all the private information becoming invisible to the public. The projector starts working with Mike’s laptop. After they finish the discussion, Alex gets back the control of projector automatically. Lisa also has great interest in both Mike and Alex’s work, since she has done a project which addressed some of the
problems they mentioned. Lisa identifies some of the relevant papers and files in her laptop and makes them accessible by Mike and Alex. Right after the talk, both Mike and Alex receive the files that Lisa has shared with them.

3.2 Goal Modeling

Human behavior and the hidden mental states are complicated, affected by situational, social and historical factors. Social and psychological studies show that human activity is usually goal-directed, as has been discussed in 2.2. The notion of goals has been adopted in requirements engineering to provide the rationale for requirements. Goals provide the basis for defining which agents should best perform which actions to fit prescribed constraints (according to their capabilities, reliability, cost, load, motivation, and so forth) [21]. Goals can be refined into sub-goals by asking how these goals should be achieved; while super-goals are found by asking why a certain goal is sought [110]. The consideration of goals raises the possibility of success and failure, not just truth versus falsity, which leads to the exploration and consideration of alternatives, decision spaces and trade-offs [121]: alternatives can be identified by asking what different ways of satisfying the super-goal exist, and conflicting goals can be figured out by asking what obstacles can be found for a goal and what goals may conflict with one another.

The main sources for identifying goals are scenarios, use cases, interview transcripts, corporate mission statements, policy statements, corporate goals etc. [7, 59]. Goals can be refined into sub-goals by asking how these goals should be achieved; while super-goals are found by asking why a certain goal is sought [110]. Goal-driven RE methods also attempt to define alternative and conflicting goals: alternatives can be identified by asking what different ways of satisfying the super-goal exist, and conflicting goals can be figured out by asking what obstacles can be found for a goal and what goals may conflict with one another.
Goal modeling [110] has been considered effective for representing stakeholder requirements of a new system. There have been several goal modeling and specification techniques, e.g. KAOS [21], Goal-Based Requirements Analysis Method (GBRAM) [7], i* [120], and NFR [83], developed to support goal-based reasoning in RE subprocesses, such as requirements elaboration, consistency and completeness checking, alternative selection, evolution management, and so forth.

The i* framework [119] offers a modeling language for understanding the problem domain. i* goal models not only provide vertical traceability from high-level strategic concerns to low-level technical details, but also allow modeling and reasoning about organizational environments and their information systems composed of heterogeneous actors with different, often competing, goals that depend on each other to undertake their tasks and achieve these goals. The i* framework consists of two main modeling components: the Strategic Dependency (SD) model and the Strategic Rationale (SR) model.

- **Strategic Dependency (SD) model.** A SD model describes a network of dependency relationships among various actors in an organizational context. The model consists of a set of nodes and links connecting the actors. Nodes represent actors and each link represents a dependency between two actors. The depending actor is called Depender and the actor who is depended upon is called the Dependee.

- **Strategic Rationale (SR) model.** A SR models allows modeling of the intentions associated with each actor and their dependencies, and provides information about how actors achieve their goals.

By explicitly modeling and analyzing strategic relationships among multiple actors, the approach incorporates rudimentary social analysis into a systems analysis and design framework. It covers both actor-oriented and goal-oriented modeling and answers the questions of WHO and WHY.
3.3 Understanding User Intention with \( i^* \)

Context-aware ubiquitous applications usually manage device-rich and highly dynamic environments, receiving only high-level guidance from users [89]. Ideally, the systems should be invisible to users, i.e., they should not be disruptive, authoritarian, or intrusive but inherently amiable and therefore invisible [115]. Therefore, understanding, capturing and processing knowledge about users plays an important role in such environments.

The support of smooth interaction between users and devices tend to be tackled by user modeling in context-aware computing. For instance, probabilistic models have been applied for making inferences about the goals of car drivers in navigating in traffic [88], and for predicting actions in a multi-user computer game [20]. For example, Bayesian networks were employed to model the interpretation of patterns of evidence by flight engineers at the NASA Mission Control Center [29], and the concurrent inference with user and expert models was used to select the most valuable information to display [28]. User models not only act as the component of context-aware applications for guiding adaptive system activity, but also serve as a cheaper alternative to user testing. By modeling skills and knowledge of human users, cognitive models, e.g., GOMS [101] and ACT-R [53] can predict human error and learning time of a specific task or tool.
Both probabilistic models and cognitive models for user modeling tend to be focused on a single process or cognitive phenomenon. User intentions or social contexts are not addressed in analysis. However, in a context-aware system, the process of a task is usually dynamically affected by context changes. Figure 3.2 illustrates the actions and state transitions of a presentation activity in a smart meeting environment, as was described in scenario S2 (§ 3.1). The context-aware process is very complex, involving not just activity routines but also context-sensitive interruptions and notifications. A comprehensive understanding of problem domain and users is important for the success of a system.

This section firstly applies the $i^*$ framework to model individual and distributed intentions involved in meeting activities, then it further analyzes the scenarios (§ 3.1) within the problem domain.
3.3.1 Distributed Intentionality

The key idea underlining the i* framework is the notion of distributed intentionality, that is, actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. The scenario S1 described in § 3.1 provides an instance of social collaboration during the activity of proposing and deciding a meeting schedule. Figure 3.3 illustrates the detailed collaboration process with a sequence diagram.

People involved in this activity play different roles, i.e., speaker, coordinator, and attendees. It requires the collaboration between them to schedule a meeting. At any point of the process, one might depend on another to proceed to the next action. Figure 3.4 shows the i* SD model for the task of organizing a meeting. Dependencies among different actors can be applied for assessing user’s strategic positioning in a multi-agent social context, and which in turn implies when and who should be notified to do what.

3.3.2 Individual Intentionality

Scenario S3 (§ 3.1) describes the case of discussion in a meeting, in which audiences choose different methods to share their idea with the speaker. As shown in Figure 3.5,
the difference is driven by personal preferences and situational consideration.

As has been discussed in § 2.2, activity theory argues that there are three levels in an activity - activity, action, and operation. Activities and actions are conscious and goal-directed processes. Different actions may be undertaken to meet the same goal, while the unconscious operations depend on the conditions under which the action is being carried out [84]. The idea of activity being directed by goals is reflected in the goal refinement process when building the i* SR model. Figure 3.6 shows an example of i* model for representing individual goals and the refinement into lower-level actions and operations of the “Data sharing” task in smart meeting room.

The model provides a detailed level of modeling by looking inside actors to model internal, intentional relationships, which provide rationale for lower-level functionalities. Operations are constrained by computing resources, while different actions may relate to different qualitative concerns. The intentional elements specified in the models include goals (e.g., “data sharing”), softgoals (e.g., “privacy”), tasks (e.g., “multi-user control”) and resources (e.g., “device description”). The notion of softgoal is used to deal systematically with quality attributes, or non-functional requirements. Intentional elements appear in the SR model not only as external dependencies, but also as internal...
elements linked by means-ends relationships and task-decompositions. The means-end links provide information about why an actor would engage in some tasks, pursue a goal, need a resource, or want a soft goal; the task-decomposition links provide a hierarchical description of intentional elements that make up a routine.

### 3.3.3 Scenario Analysis

Systems that react appropriately to users need to have a deep understanding of the context of reaction, e.g., services, physical resources, user intentions and expectations. Goals can be useful when modelling contexts. The i* framework offers a modeling language for understanding the problem domain, in particular the internal intentionality and social dependencies, of a human task. The contextual elements represented in the goal model include:

- **Physical artifacts.** Services and devices in the environment are represented as resources;
- **Tasks & functionalities.** Tasks and lower-level system behavior correspond to hard-goals in the i* framework;
• **User preferences.** Humans usually have high-level qualitative concerns that may guide their action and strategic decision. Such concerns are represented in $i^*$ as soft-goals;

• **Roles and dependencies within a community.** A network of dependency relationships among various actors in an organizational context is described in the goal model to show who an actor is and who depend on the work of an actor;

• **Strategic Rationale.** The goal graph provides vertical traceability from high-level strategic concerns to lower-level tasks and system behavior.

This section described the goal-based analysis of scenarios in a smart meeting environment. The $i^*$ framework was applied for identifying and representing physical, social and subjective concerns in context-aware activities. Table 3.1 summarizes our observations. S1 emphasizes the socialization in human activities: People depend on each other to achieve a task. It is required that a context-aware system be aware of the social dependencies and be able to notify users the state changes of others. S2 emphasizes the collaboration and state transition between devices and services. A context-aware system need to monitor state changes and react appropriately to those changes. S3 exhibits the variability existing in human cognition and preferences. User intents and preferences need to be identified for context-aware engines to select from alternatives to better serve users.

Overall, cognitive and social factors, especially social dependencies and individual intentionality, to a great extent affect human behaviour. The diversity of human behaviour is partly caused by cognitive variance. Therefore, the support of intelligent notification and interaction in context-aware environment requires a full knowledge of users in social and subjective settings. Instead of merely processing behavioural patterns with user modeling, we need to emphasize traceability between lower-level operations and high-level strategic concerns in the process of completing a task.
### Table 3.1: Goal-based analysis on scenarios of smart meeting room

<table>
<thead>
<tr>
<th>Features</th>
<th>S1- Scheduling</th>
<th>S2 - Presentation</th>
<th>S3 - Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>social dependency</td>
<td>service collaboration</td>
<td>cognitive variability</td>
<td></td>
</tr>
<tr>
<td>collaboration</td>
<td>state transition</td>
<td>interaction</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Factors</th>
<th>S1- Scheduling</th>
<th>S2 - Presentation</th>
<th>S3 - Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>organizational goals</td>
<td>organizational goals</td>
<td>organizational goals</td>
<td></td>
</tr>
<tr>
<td>individual goals</td>
<td>individual goals</td>
<td>individual preference</td>
<td></td>
</tr>
<tr>
<td>collaborator’s action</td>
<td>devices/service status</td>
<td>devices/service status</td>
<td></td>
</tr>
<tr>
<td></td>
<td>notification</td>
<td>interruption/notification</td>
<td>interaction</td>
</tr>
</tbody>
</table>

### 3.4 Goal-based Evaluation of Context-aware Applications

The focus of current research on context-aware computing is on the design of system infrastructures and applications. The problem of verifying context-awareness design has attracted little attention. However, the development of pervasive systems is very difficult due to limited computational resource and dynamically changing and fault-prone devices and services, which implies that the developed system is susceptible to errors and may pose significant challenges for system evolution. Therefore, evaluation is a fundamental stage in their development. It is necessary to evaluate whether the context-awareness really improves the system performance and whether users really prefer the context-aware interaction.

Formative and summative evaluation of context-aware systems is difficult due to the complexity of situations. There are numerous external users and contextual factors that confound the evaluation process. Most context-aware applications presented only preliminary evaluations. The evaluation methodologies borrowed from HCI, for instance, questionnaires [85] and user tests, are used for measuring user’s satisfaction. However,
user tests usually only provide some preliminary results, and it is hard to assess reli-
ability and validity of user justifications. Some metrics derived from the evaluation of
information retrieval systems have also been exploited to evaluate context-awareness,
e.g., accuracy of recommendations, accuracy of system predictions, similarity of expert
rating and system predictions, etc. [38]. The data to be analyzed by these metrics are
usually gathered using log files, which provide a reliable measurement for certain aspects
of performance. However, it is limited, as it requires a running system and a dichotomous
classification of user-relevant content.

Goal modeling [110] has been considered as an effective way of representing stake-
holder requirements for a new system. Goal models can not only capture user goals but
also provide rationale to lower-level functionalities. In this section, we present a frame-
work for evaluating context-aware applications by comparing the design of a context-
aware system to user expectations. The main idea is to reveal and clarify users’ view-
point of context-awareness with goal modeling techniques, and then verify the design
against those requirements. In this framework, goal models are used to represent user
expectations and system design respectively, and metrics and principles are developed
for comparing system goal models to user goal models. A case study on the smart meet-
ing room application (as described in Section 3.1) was conducted to illustrate how this
framework works.

3.4.1 The Framework of Goal-based Evaluation

The framework we used for context-aware system evaluation is shown in Figure 3.7. Goal
modeling has demonstrated benefits for identifying resources, goals, and alternatives. We
use the i* goal models to identify user expectation (user goal model) and system design
(system goal model) on context variables and adaptation strategies, and evaluate user
satisfaction by measuring the degree of matching between them. Three types of context
variables “resource”, “hard-goal” (or functional goals) and “soft-goal” (or quality goals)
are identified in the goal models. The positive or negative contribution links constitute the representation of strategies for context-aware reaction. If two goals have positive contribution to a same higher-level goal, these two lower-level goals form two alternatives for achieving the higher-level goal.

**Collecting user expectations**

The main sources for identifying the user’s viewpoint of a system are scenarios, use cases, interview transcripts, etc. The subjects who are interviewed should be representative potential users. Prior to each interview, the interviewer provides some materials, introducing the goal of the interview, the system environment, availability and limitation of the techniques. Interviewees are expected to express their functionality or quality concerns with a full understanding of the problem domain and without being biased by the introduction materials. To ensure the validity of information collected from interviews, some other materials or methods, e.g. user observation, might also be used to validate the interview results. The validated interview transcripts are then employed as the main source for identifying the contextual elements and for building a draft user goal model. The goal model should be built by experts.
A user goal model contains the information collected from different sources and different individuals. The model builder needs to identify conflicts, inconsistency and alternatives in the model or by annotations. The quality of goal models can be assessed by mapping back to the interview transcripts.

**Recapturing system goal models**

Reverse engineering [16] has been accepted as a way of abstracting information from lower-level software implementations, for instance, extracting the structure of a legacy system from code and data with the goal of transferring this information into the minds of the software engineers trying to reengineer or reuse it [82]. As most software changes result from changing requirements, several approaches have been proposed for recapturing and understanding software requirements from implementation, such as reverse engineering state-based requirements specification for process-control software [46], and recapturing goal models from legacy code [122], etc. The process we applied for extracting goal models from context-aware applications is shown in Figure 3.8. To ensure the completeness of system goal models, all available software artifacts, e.g., source code, demos, and technical papers, should be used for extraction. And the extracted draft goal models are verified with iterative and bidirectional mapping to original software artifacts.

**Comparing system goal models to user goal models**

Goal models are high-level, loosely structured, so it is hard to compare two goal models automatically. Therefore, the comparison should be carefully conducted by an expert, who is familiar with both goal modeling and the problem domain. We defined two metrics, *coverage* and *demand*, to evaluate the adaptation design against user expectations. *Coverage* measures the percentage of goals in user goal model that are achieved by an existing system. *Demand* measures the percentage of goals in the system goal model that are demanded by the users.
Figure 3.8: Steps for extracting system goal models

The calculation of coverage and demand is based on careful examination of both the descriptive and structural information inside the models. The following principles are defined for the model comparison process to ensure the validity of results:

- The comparison of goal models should follow a bottom-up search strategy for checking the overlap between models;
- Two resources are considered an overlap only if they belong to the same category or have the same device specification;
- Two hard-goals are considered an overlap only if they contain the same information or support similar functionality;
- Two soft-goals are considered an overlap only if they contain the same information and there exists at least one lower-level goal in each goal model that has positive contribution to the soft-goal.
3.4.2 Results Analysis

Considering the generality and reasonable complexity of problem, we use a smart meeting room with 10 to 20 seats, as described in Section 3.1 for evaluation. We choose two projects for analysis: the Interactive Workspaces (iRoom) project [52, 3] at Stanford University, and the EasyLiving project [13, 1] at Microsoft.

User expectations for a smart meeting room were identified and collected from individual interviews with potential users. The subjects were randomly selected from graduate students majoring in computer science and employees in different companies. The only constraint was that the subjects have some prior experience with workshop meetings (either leading or auditing in the meetings). The procedure for collecting user requirements and building the user goal model was carefully planned, conducted and validated. The user goal model built from the interview transcripts is shown in Figure 3.9. Soft-goals, hard-goals and the dependency relations were mainly elicited from asking “how” and “why” questions during the interviews. For example, if the soft-goal “easy to use” was mentioned by the interviewee, asking “how” questions may help him to identify lower-level functional concerns, e.g., “remote control” and “automatic control”.

Figure 3.10 and Figure 3.11 shows the goal models of the iRoom project and the EasyLiving project respectively. The models are constructed by following the reverse engineering process (as illustrated in Figure 3.8). A draft system goal model was extracted from source code, capturing all lower-level functions and goals. This draft model was then refined by mapping to the technical papers and documents, in which the higher-level quality concerns was explained for the system design. To ensure the quality of the system goal model and minimize the subjective bias of the model builder, we validated the model by inviting another goal modeling expert to map it back to all the documents.

We compared the system goal models to the user goal model according to the principles described in Sec. 3.4.1. Table 3.2 summaries the results. The coverage and demand values illustrate the degree of overlapping between design and user expectation. The
Chapter 3. Goal-based Analysis of Existing Work on Context-aware Computing

Figure 3.9: The user goal model for a smart meeting room

Figure 3.10: The system goal model for iRoom project
Figure 3.11: The system goal model for EasyLiving project
Chapter 3. Goal-based Analysis of Existing Work on Context-aware Computing

Table 3.2: Results of model comparison

<table>
<thead>
<tr>
<th></th>
<th>User GM</th>
<th>EasyLiving</th>
<th>iRoom</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#(goals)</td>
<td>coverage</td>
<td>demand</td>
</tr>
<tr>
<td>resources</td>
<td>4</td>
<td>75%</td>
<td>75%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100%</td>
<td>51%</td>
</tr>
<tr>
<td>hard-goals</td>
<td>27</td>
<td>41%</td>
<td>79%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>48%</td>
<td>75%</td>
</tr>
<tr>
<td>soft-goals</td>
<td>10</td>
<td>50%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>70%</td>
<td>100%</td>
</tr>
<tr>
<td>total</td>
<td>41</td>
<td>46%</td>
<td>84%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>59%</td>
<td>74%</td>
</tr>
</tbody>
</table>

results show that neither of the two projects completely cover all the user goals, nor do users expect all of the goals being implemented in these projects. Differences exist between user and system goals. Another interesting observation is that although neither of the projects completely cover the soft-goals in user model, the demand rates of soft-goals are both 100%, which implies that the soft-goals proposed by both projects are accurate but incomplete in comparison with the user goal model. In other words, users are helpful for identifying non-functional requirements.

The evaluation results of two projects are shown in Figure 3.12 with X-axis referring to coverage rate and Y-axis referring to demand rate. The high rate of coverage and demand means that the system design is consistent to user expectation. The low rate of both coverage and demand means that the system design deviates greatly from user expectation. The system, which exhibits high demand but low coverage, supports limited functionalities which are expected by user. If the coverage is high but demand is low, it means that there are a number of features provided by the system but not expected by the users. Such systems require a lot of learning effort of users. Overall, the iRoom project exhibits higher coverage than EasyLiving. This evaluation result is consistent to the fact that EasyLiving was an experimental system, while iRoom is already used as the main project meeting room and the research has been motivated by the requirements in
In addition to quantitative comparison, a difference also exists on the identified resources, goals and strategies between user goal model and two system goal models. Regarding the soft-goals, the EasyLiving project focuses on facilitating interaction with devices and computers, iRoom was designed to improve the interaction among meeting participants, while users emphasize both goals as well as the goals of system security and user privacy.

Since the \( i^* \) model integrates viewpoints from all users, the user goal model captures a variety of quality dimensions and ways to achieve these soft-goals. For example, in Figure 3.9, three alternatives were identified for “sharing data among participants”, and each contributes to different levels of efficiency, security and privacy. However, fewer alternatives were provided by the EasyLiving and iRoom project. EasyLiving was implemented as a fully reactive system, in which user activities are tracked and the environment automatically reacts to activity changes. iRoom let users adjust the environment as they proceed with their task, and system is only responsible for providing a fluid means of executing the actions.

Significant difference exists in lower-level goals and resources. Most of the goals in user goal model are not covered by existing systems because the lack of alternatives and
high-level goals supported by systems. Some lower-level goals are not demanded in the
user goal model because of users’ limited domain knowledge. For example, “fingerprint
database” in EasyLiving and “barcode scanner” in iRoom were identified as the lower-
level resources for the goal of “identify a person”, but none of them was mentioned by
users. In this sense, the users do not show advantages in identifying functional require-
ments or lower-level computation resources.

3.5 Summary

As a modeling technique for eliciting social dependencies as well as physical and qualita-
tive concerns in early-phase requirements engineering, the $i^*$ framework can be applied
to capture knowledge of users in pervasive environment. This chapter has used $i^*$ goal
models and a running example to analyze context-aware activities in context-aware en-
vironment.

Section 3.3 analyzed three scenarios of context-aware activities with UML models
and built $i^*$ models for understanding individual and distributed intentionality which
affects human behavior. It shows that social dependencies, cognitive variance and state
changes of devices or services are three key elements conditioning responsive actions in
context-aware environment.

Humans are the centre of computation in ubiquitous environment. A system model,
which is inconsistent to user’s mental model, is more likely to risk user frustration and
dissatisfaction. Section 3.4 provided the results of user evaluation for two smart meeting
projects. It shows that a difference exists, from the higher level qualitative concerns to
lower-level functionalities and resources, between user expectations and system design. The project which has been motivated by requirements in real use exhibits better coverage
of user expectations.

Both the scenario analysis and the goal-based evaluation have emphasized the need
to address social and individual contexts in context-aware computing. The $i^*$ framework has demonstrated the effectiveness of modeling user goals for the exploration and consideration of alternatives, decision spaces. Context-aware computing will have a very open domain of individual and collective user goals that must be interpreted dynamically: inferring individual user task goals for service adaptation. There is a need of developing context models that capture the expected or perceived contexts from individuals at a higher, less implementation-specific level.

Independent and individual context views and the explicit analysis of cognitive variations will result in a more precise specification of context space and context-aware activities. Diverse context views can form a knowledge base for system analysts to observe and deal with the trade-offs and conflicts in social and technical constraint. Clustering and comparison of various context views will reveal patterns and assumptions hidden in explicit human activities. Furthermore, with a context model in which individual perspectives can be traced, the responsive system actions of a ubiquitous system can be more reliable and user-friendly.
Chapter 4

The Framework of Cognitive Context Modeling

The review of context concepts and modeling techniques (Chapter 2), as well as the results of goal-based analysis on scenarios and applications of context-aware computing (Chapter 3), has suggested the importance of addressing cognitive bias in context modeling and in the specification of context-based system actions. The remainder of this thesis describes a cognitive context modeling framework that seeks to address the problems in context modeling and meet the objectives set out in § 3.5.

This chapter presents the key concepts of the framework. Section 4.1 analyzes the pattern of context-aware task processing and identifies the key steps in context modeling with the concern of cognitive bias. Section 4.2 presents the structure and elements for representing task and its contexts. Section 4.3 provides a brief summary of this chapter.

4.1 Methodology Overview

Ubiquitous computing envisions a world in which users interact smoothly with device-rich environments to perform various kinds of tasks [89]. In a ubiquitous environment, the system needs to monitor the state of services, devices and users, infer user goals
from their activities, and adjust its behavior accordingly. An essential feature of ubiquitous computing is the ability to initiate responsive reactions to contextual changes. The adaptive behavior results from effective judgement of the situation and dynamic reconfiguration of the operations.

As shown in Figure 4.1, a context-aware adaptation engine for run-time service composition and reconfiguration usually consists of three components: context monitor, adaptation policy and reconfiguration actions. The context monitor detects the state of a specific source of information that is “interesting” for the operation. The adaptation policy describes how the system action should be performed according to the contexts being monitored. The reconfiguration action refers to the invoking of specific adaptation actions as directed by the policy.

![Figure 4.1: The structure of adaptation engine](image)

Since the purpose of context-aware computing is to provide applications that serve a real or perceived human need, it is important to design the system from the end-user’s perspective, considering how any system or infrastructure to be built will be used and why people use it. Traditional techniques of requirements elicitation, e.g., interviews, scenarios and field observations, have recently been applied to the design process of ubiquitous systems for identifying the operational requirements that process an object into an outcome [49, 64]. The goal of our methodology is to complement existing requirements elicitation techniques and provide a systematic framework specifically for the elicitation and modeling of environmental settings of a ubiquitous task.

Within this framework, the context-aware task processing is divided into two branches,
as shown in Figure 4.2: the task operation branch that turns a task into an outcome, and the context-awareness branch that manages the context information and guides the task execution process. Accordingly, in order to elicit the contexts of a ubiquitous application, the framework addresses three aspects of modeling: i) system operations for executing the task; ii) context space that the system should be interested in; and iii), adaptive policies for guiding responsive system activity.

There is, in principle, no limit to relevant facts for a given task. According to John Dewey’s theory of context and human thought [22], context has two components: 1) background, which is both spatial and temporal and is ubiquitous in all thinking; 2) selective interest, which conditions the subject matter of thinking. Explicit modeling techniques are always susceptible to incorporating too many or too few facts about the situation [30]. Since computing resources are usually limited, bringing in too many redundant or irrelevant factors into the monitoring and reasoning process may affect the efficiency of context-awareness. On the contrary, an over-simplified context model that marginalizes the complexity and diversity in human-computer interaction will result
in inaccurate interpretation of the situation and will cause context-aware behavior that deviates from users’ expectation.

The key is therefore to capture the mental models of participants in a context-dependant activity. The mental models should cover a variety of environmental facts and provide a knowledge base for the selection of an “appropriate” context space that balances subjective complexity and technical constraint, containing only the most relevant facts. The context elicitation and modeling process should allow system designers to answer “why” questions regarding their choice of the context domain for monitoring and reasoning on.

The success of a software system usually involves an iterative process of planning, prototyping, testing and deployment. Similar to the spiral lifecycle of software development [9], the framework of cognitive context modeling envisions the task of context elicitation as a continuously developing and evolving process, as shown in Figure 4.3.

![Figure 4.3: The process of context elicitation and modeling](image)

The cognitive context modeling process contains four phases:

1. *Data collection*: selecting representative participants of the task and collecting raw mental data of each participant with interview, think aloud, etc.;
2. **Model building**: constructing a cognitive context model that captures episodic representations of personal knowledge as cognitive context views;

3. **Context space identification**: identifying the context space based on the classification, integration and optimization of context views;

4. **Context-awareness specification**: specifying rules for initiating context-aware activity and providing conflict resolution strategies for the context space.

Phase 1 and 2 focus on early-phase user-centric knowledge elicitation and representation, while phase 3 and 4 are more system-oriented, i.e., they are conducted by analysts to identify the contexts for monitoring and to specify the context-adaptive system behavior. The cognitive context model is used as a communication medium between end-users and system developers in order to ensure that the finally released context profile balances cognitive complexity and physical constraint. User satisfaction is assured with the systematic development of a cognitive context model, which captures a variety of context elements and cognitive views. The growth along the radial dimension and the spiral cycles reflects incremental and iterative growth of context definition. The aggregation of partial and continually evolving personal knowledge during the model building phase (division 2 in Figure 4.3) allows traceability and risk management when building context-aware system prototypes.

We’ll use the following example to demonstrate various aspects of the cognitive context modeling framework:

*To reduce the environmental impact of transportation, people may use greener ways to get around by choice, e.g., walking, biking, or public transit; however on occasion, they may choose to drive a car or take airplanes for efficiency. The decision over which transportation methods to use for a particular trip are context dependent, and are affected by the time and situation.*
4.2 Cognitive Context Model (CCM)

As has been reviewed in § 2.1, contexts are rich in features; no index can exhaust contexts [86]. To determine which environmental elements are relevant requires the involvement of human cognition. Basically, the contexts can be classified into two categories: objective context and cognitive context. Objective context refers to observable factors within which a course of action occurs; cognitive context refers to a set of beliefs belonging to an individual or a community, indicating which objective factors might be relevant in any situation. The objective context can be sensed with a certain level of accuracy, while the cognitive context is subjective and fluid.

![Figure 4.4: The conceptual structure of CCM](image)

Figure 4.4 shows the conceptual structure of CCM. There are three layers in CCM: task, ObjCt and CogCt. The top (Task) layer of the structure contains a high-level object (task) and flows of operations for executing the task. The bottom (ObjCt) layer represents the objective settings in the environment during the process of task. The CogCt layer lies between Task and ObjCt, capturing
the beliefs (cognitive views) of the participants in the task.

A CCM is constructed through the specification of a task, a collection of cognitive views (CogCt) and a depiction of the world (ObjCt):

1. Task specification: decomposing the task execution process into a flow of operations;
2. View collection: eliciting individual cognitions of the world relevant to the task;
3. World depiction: modeling the world in which a task executes.

The following subsections describe in detail the structure of each layer and the procedures for building the model.

4.2.1 Task

The task represents an object or goal of a subject or a community. A task can be interpreted as a flow of operations for transforming the object into an outcome. The process of the transformation, e.g., when and where to execute which operation, is affected by the state of a set of contexts.

Figure 4.5: An example of task model

A precise identification of the main operations involved in a task and an abstract description of series of actions that transform an object into an outcome are necessary
for limiting the application domain and providing both users and developers a better understanding of the problem domain. The precise specification of a task can be obtained through two phases: analysis and modeling.

The analysis phases focuses on exploring and identifying operations and alternative strategies that form a task process. Different techniques can be used in this phase, such as interviews, questionnaires, storyboards, scenarios, etc. [39, 69]. The result of task analysis is an informal and detailed list of operations and attributes, which can be used to develop an abstraction of the problem domain.

The purpose of task modeling is to precisely describe the operations and their relationships according to the results of task analysis. Figure 4.5 shows an example of the task model for “green transportation”, as described in § 4.1. According to this model, the task of green transportation can be interpreted as reducing actual transport usage by walking, or switching to a cleaner mode, e.g., biking or taking public transit in turn. The actual transport decision, i.e., choosing the “Yes” or “No” branch associated to each operation box, is subject to cognitive and situational contexts.

A task can be described precisely using different models. As has been illustrated in § 2.4, several techniques, such as GOMS [101], i* [120], and Petri-nets [87], can be used to model the actions and alternatives of the task process. Generally, a task model should provide coverage for the following elements:

- a brief description of the task/object;
- operations/actions that constitute a task process;
- conditions/prerequisites for executing an operation;
- the outcome of a flow of operations.

In the CCM framework, the task model, which is both abstract and readable, is the driving force of context modeling. An explicit and precise specification of the task will ensure the validity of context elicitation and identification.
4.2.2 ObjCt

The *ObjCt* identifies the detectable surroundings, i.e., *objective contexts*, during the task process. For the “green transportation” task, potential objective contexts include “rush hour” (time), “nearby bus station” (location), and “warm and sunny weather” (event), etc. Assuming that $P$ is part of the task model while $F$ is part of the objective surroundings of the task, there should be two general assumptions for determining the validity of an *objective context* for a given task:

- **R1**: a distinction can be made between the operations ($P$) in task and the factors ($F$) relevant to the operations, i.e. $F \notin P$;

- **R2**: the possible cause or effect of the factors ($F$) to the operation ($P$) of a task remains relatively constant, i.e., $F \rightsquigarrow P$.

**R1** ensures that the identified context factor $F$ is not an operation or the condition of an operation; while **R2** ensures the generalizability of the identified context factor, such that a model learned in one circumstance can be successfully applied in the same context (“the same context” means that the circumstance is sufficiently similar).

*ObjCt* represents the physical world in which an activity situates. There have been many approaches to the identification of world dimensions and representation of different types of elements in the world, as described in § 2.1. We adopt the “5W+” (*who, what, when, where, why, how*) methodology [5, 26] and extend Zimmermann’s division of context space [124] to classify context sources into eight dimensions, i.e., human, device, time, location, activity, and event. The detailed description of each subcategory is shown in Table 4.1. Different from traditional context modeling framework that isolates the system from human activities, we assume that computing devices act as a mediator in human activities. The device is a special type of participant (*who*) in a task and the system event can be viewed as a type of action (*what*) performed by devices.
Table 4.1: Dimensions of context space

<table>
<thead>
<tr>
<th>W5</th>
<th>Dimensions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>who</td>
<td>human</td>
<td>participant’s information: in terms of role, state and preference</td>
</tr>
<tr>
<td></td>
<td>device</td>
<td>device specification: in terms of model, components and state</td>
</tr>
<tr>
<td>when</td>
<td>time</td>
<td>date (YY/MM/DD) and time (HH:MM:SS)</td>
</tr>
<tr>
<td>where</td>
<td>location</td>
<td>location information: in terms of address, or geographic coordinates (longitude, latitude)</td>
</tr>
<tr>
<td>what</td>
<td>activity</td>
<td>participant’s action or interaction with device</td>
</tr>
<tr>
<td></td>
<td>event</td>
<td>behavior that is triggered by a device or physical environment</td>
</tr>
<tr>
<td>how</td>
<td>relations</td>
<td>user’s view of relations between context elements</td>
</tr>
<tr>
<td>why</td>
<td>relevance</td>
<td>cognitive view of context in terms of a score of relevance</td>
</tr>
</tbody>
</table>

The ObjCt captures not only the physical elements in the world, but also the inter-relationship between them. The cognitive context model identifies three basic types of the relationship:

- **ownership (human – device)**: indicating the ownership between human and device;
- **operation (device – event; human – activity)**: indicating the involvement of devices in an event or the participation of people in an activity;
- **parameter** ({event, activity} – {time, location}): indicating the time / location tag of an event or an activity.

### 4.2.3 CogCt

Different participants of a task play different roles, and they have different concerns during the process of a task. Therefore, we structure the CogCt component in CCM as a composition of cognitive views on the task and its context. Each view is drawn from
an individual or a community participating in the task, such that various participants involved in the task can express their own observations and considerations.

The representation of a context view can be viewed as a subset of ObjCt, including a set of objective contexts and a set of relations between these context elements. What distinguishes the context view from an ObjCt is the inclusion of subjective information. A context view represents an episodic portion of the world, which is biased due to the limit of human cognition. The recognition of context is reflected by weighting objective contexts for their relevance, which is indicated with links between context elements and the task as well as the score values on them.

Figure 4.6 shows an example context view for the “green transportation task”. A context view contains three types of elements: the task(object), objective context, and the score of objective context. Objective contexts are grouped by source; and in each group, each objective context is associated with a score, indicating the relative importance of the context in its group (a higher score means that its associated context element is more important than those with lower scores). For instance, the device context “GPS” is scored “2” while the score of “cell phone” is “1”, which implies that although both GPS and cell phone might be relevant to the green transportation task, the viewpoint holds that GPS plays a more important role.

There are two types of links between elements in a context view: the arrow line which indicates the relevance of an objective context to the task, and the arc which represents the relationship between two objective contexts of different sources. For example, the arcs between “Rush Hour”, “Biking”, and “Alex” imply that “Alex prefers biking during rush hour”; while the arcs between “Alex”, “Cell phone” and “Raining” imply that “Alex’s cell phone provides weather information”.

The elicitation of a context view from participants provides a detailed and empirical account of some aspects of the notion of awareness, i.e., being conscious of an event, action, object or person and their properties. For example, the context view in Figure 4.6
Figure 4.6: An example of context view

The arrow line denotes potential effects on the task, the prefix number ranks the relevance. Three colours of arcs denote three types of relationship between contextual elements (blue-ownership, red-operation, green-parameter).

represents the view of one particular participant (“Alex”) on the social and technical context for the “green transportation” task. This view includes Alex’s three activities, i.e., taking public transit, biking, and walking, that contribute to greener transportation, and the situation, e.g. factors such as the weather and road condition that affect Alex’s selection of transportation method.

The context views do not imply that all levels and all details in the process of a task are always conscious. Context is fuzzy and extensive in scope. The CogCt layer in CCM addresses this issue by aggregating views from various perspectives to form a systematic cognitive context. The views are weighted according to the importance of the role, for which the view owner played in the task. For example, since Alex’s context view, as shown in Figure 4.6, has identified “Lisa” as a potential participant of the task, it suggests that Lisa’s context view should also be identified and included in CogCt.
4.3 Summary

This chapter presents a new methodology for eliciting cognitive context, identifying context dimensions and the quantitative variations of the dimensions, for a ubiquitous task. It uses a three-layered structure - Cognitive Context Model - to represent task, contexts of the task, and the mental models of the participants in the task. This structural model allows individuals or community participants of a task to specify their situational considerations and preferences, which form the individual mental models of a ubiquitous task. The inclusion of individual situational perspectives into context model addresses the fuzzy and fluid aspect of context, emphasizing the modeling of diversity and dynamics in human cognition and context-aware activities.

Using clustering and optimization techniques, the diverse contextual factors captured in CCM can be analyzed and integrated to generate an optimized context view. The optimization output can then be used as a basis for context specification, i.e., specifying the operations for transforming the task into an outcome, identifying “appropriate” context space for monitoring, and designing the context-action policies for guiding responsive system behavior. The components of CCM, the use of optimization techniques for mental model integration, and the use of CCM optimized view for context-aware behavior specification are illustrated by a running example.
Chapter 5

Context Space Specification

The previous chapter presented the framework of cognitive context modeling and described the structure of cognitive context model (CCM) for representing contexts of a task and managing multiple cognitive views. Diverse context concerns from task participants constitute a knowledge base for analyzing cognitive variability and dealing with cognitive biases in context space decision. This chapter presents the techniques for integrating context views into an optimized context space (§ 5.1) and identifying adaptation rules of responsive actions in context-aware computing (§ 5.2).

5.1 Context Space Identification

Humans are quite successful at using background information and reacting appropriately, and human cognition varies widely among task participants due to personal knowledge, experience and preference. The CCM model represents a task and its objective and cognitive contexts in a structural and flexible form. In CCM, the ObjCt captures the objective and interconnected contexts surrounding the task, and the CogCt component is used specifically for capturing diverse end-user cognitive views of the contexts surrounding a task. The CogCt represents the diversity in human cognition with context views, which provide input from potential end-users for user-centric system design.
However, computers are not currently well enabled to take full advantage of the context of human-computer interaction. Context-aware computing requires a consistent view of the context domain for decision making and task processing. In ubiquitous computing, the identification of an “appropriate” context space, which defines a scope of “interesting” contexts for monitoring, is not only subject to subjective interests but also conditioned by physical/technical constraints. An “appropriate” context space should balance end-user requirements and technical constraints to achieve maximized user satisfaction. Therefore, context specification for a context-aware ubiquitous system can be divided into two sub-tasks: i) identifying an “appropriate” context space, which defines a scope of “interesting” contexts for monitoring; ii) specifying context-aware behavior that maps context state to system activity.

Limited physical and computation resource and the objective of system interacting smoothly with users are usually conflicting. Trade-offs exist in the decision of contextual domain for a ubiquitous system. The process of choosing the best or the most “appropriate” context space from some set of alternatives can be interpreted as an optimization problem. In CCM, there are a number of context views in $CogCt$ and a limited number of objective contexts in $ObjCt$, which makes it possible to apply optimization techniques (e.g., AHP calculation or Genetic Algorithms) on CCM to integrate various views and generate optimized score for each objective context. The optimization output, i.e., the optimized context view, which contains both objective surroundings and cognitive information of a task, can thus be used to facilitate context space identification.

The following subsections describe in detail the AHP-based approach for integrating and optimizing context views to facilitate context space identification. The process of using the optimization output for context definition and context-aware behaviour specification is presented in § 5.2.
5.1.1 The *Analytic Hierarchy Process* Method

The Analytic Hierarchy Process (AHP) [95] is a mathematical technique developed by Thomas Saaty for complex multi-criteria decision making. Based on the assumption that when faced with a complex decision the natural human reaction is to cluster the decision elements according to their common characteristics, it provides a framework for decomposing a decision problem into a hierarchy of sub-problems and evaluating alternative solutions.

![Figure 5.1: A simple AHP hierarchy](image)

Fig. 5.1 shows a simple AHP hierarchy. It consists of an overall goal or problem, a group of options for reaching the goal, and a set of criteria that relate the options to the goal. With the identified criteria and objective or subjective judgement about the elements in the hierarchy, the options can be systematically evaluated by comparing them to one another in pairs. AHP then converts the evaluations to numerical weights or priorities for each element of the hierarchy, allowing diverse and incommensurable elements to be compared to one another. The numerical weights derived for the bottom-level options represent their relative ability to achieve the decision goal.
5.1.2 Integrating Views with AHP

The CogCt component in CCM captures different views on context information from different actors or participants in a task. However, a consistent view of the problem domain is required for decision making or task processing. AHP provides the ability to vary the weighting of factors towards the objectives and is useful for dealing with various cognitive views as presented in Figure 4.4 (see § 4.2). The AHP method involves calculations using simple formulas and hence is expected to provide better computational performance than the alternative optimization techniques such as Genetic Algorithms.

Table 5.1: Mapping CCM to AHP hierarchy

<table>
<thead>
<tr>
<th>Element</th>
<th>AHP</th>
<th>CCM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Goal</td>
<td>task</td>
</tr>
<tr>
<td></td>
<td>Criteria</td>
<td>views</td>
</tr>
<tr>
<td></td>
<td>Options</td>
<td>objective contexts</td>
</tr>
<tr>
<td>Weight</td>
<td>weight of criteria</td>
<td>weight of views</td>
</tr>
<tr>
<td></td>
<td>score of options</td>
<td>score of obj. contexts</td>
</tr>
</tbody>
</table>

Table 5.1 gives the mapping between CCM elements and AHP hierarchy, which transfers the problem of optimizing cognitive context views into an AHP optimization process. To process the diverse cognitive views in CogCt into an optimized view, we first aggregate the context elements and interrelationship between contexts in all views into a systematic ObjCt, then convert the CCM architecture into AHP hierarchies, each hierarchy representing a set of views on a category of objective contexts with regard to their relative importance(score) for the task. An AHP calculation can thus be applied to optimize the score of each objective context.

The AHP-based view optimization and integration consists of the following three
steps:

**Step 1: Decide the relative weight of each view.** Each context view should be assigned a weight value so that the sum of all weights is 1. The weights can be either equal or different. For example, if there are 3 views in a CCM and each view is weighted equally to the others, then the weight \( W(V_i) \) of each view \( V_i \) should be \( W(V_i) = 0.33, 1 \leq i \leq 3 \).

In CCM, context views are drawn from the subjects (individual or community) of a task. Initially, at the design phase, the weights can be assigned by an analyst, who holds a systematic view of the task, according to the relative importance of the subjects in the task. However, at run time, the weights could also be tuned by the user to make the system’s context domain adapt to various situations and preferences. The application of runtime weight adjustment is discussed later in § 6.2.5.

**Step 2: Compute the relative importance of each context element.** As described in § 4.2, the CogCt is composed of views which identify a number of objective elements along different dimensions of the environment. In each context view, the objective contexts of the same category can be compared, according to their relevance to the task. Thus, the goal of this step is to convert the subjective ranking/comparison into relative weights for each dimension of context elements. This computation can be divided into three sub-steps:

1. Assign an integer score (in the range of 1-9) to every context element, where a high score implies the high relevance to the process of the task. For example, if there are 4 system events \( (E_j, 1 \leq j \leq 4) \) identified as contexts for the task \( T \), they may get score \((S_{E_1}, S_{E_2}, S_{E_3}, S_{E_4}) = (9, 5, 3, 1)\) respectively according to their rankings in the view model \( V_i \). The ranking of context elements reflects the users’ priorities in the context space. The scores can be obtained from the
view master’s explicit input or with the assistance of software/hardware that monitors and measures the statistical coincidence of the context elements and the process of task.

2. Based on the score(S) of a group of context elements, calculate the pair-wise comparison matrix for each view $V_i$, such that $P_{ij} = \text{round}(S_i/S_j)$, where $1 \leq i, j \leq n$ and $i \leq j$, and for $i > j$, $P_{ij} = 1/P_{ji}$. E.g., continuing the example above, $P_{E_1E_2} = \text{round}(9/5) = 2$.

3. For each group of context elements in each view, normalize each $P_{ij}$ such that $P_{ij} = P_{ij}/\sum_i P_{ij}$, and then average across rows to obtain the relative weights of the context elements ($W_{V_iCt}$).

The pair-wise comparison matrix, weight matrix and the resulting relative weights for the above example is shown in Fig. 5.2.

**Step 3: Calculate the optimized weight of each context.** The weight of each context in each dimension of context source for the integrated view is the sum of the ObjCt’s relative weights ($W_{V_iCt}$) in each view multiplied by the view weight ($W(V_i)$), i.e., $W_{Ct_j} = \sum_i (W_{V_iCt_j} \times W(V_i))$.

![Figure 5.2: An example of AHP calculation](image)

The output of AHP optimization forms an optimized context view, which integrates all context views and associates each objective context with an optimized score. As an example, Figure 5.3 shows an optimized context view for the “green transportation” task,
which integrates the views from “Alex” and “Lisa”, as shown in Figure 4.6. Alex and Lisa are both the elements of “Human” contexts and the subjects of context views. Since they play a similar role in the task, their views are equally weighted.

![Figure 5.3: Optimized context view](image)

The optimized view synthesizes the context elements and their interconnections from both views. The optimized score associated with each context element scales the relevance of each fact to the task. The rankings of contextual elements provide the reference for deciding the context domain of a context-aware system. An element that has a higher optimized score is more likely to be included into the context domain. The rankings are within each dimension. For example, in Figure 5.3, the highest score in the “Location” category (0.76) indicates that according to the views of Alex and Lisa, the location of “destination” is more relevant than other “location” elements, that is, the “destination” should generally be taken into account for balancing efficiency and carbon emission.

A context space is composed of interconnected temporal, physical, and social fac-
tors, which surround the task and affect the task process. When deciding the context domain for a context-aware system, the relationship between elements of different categories should also be considered. For example, as shown in Figure 5.3, “Biking” has the highest score within the “Activity” dimension. And if it is within the context domain, its connected elements – “Weekday” and “Night” – should also be included, although they have the lowest score within the “Time” dimension.

5.2 Context Specification

In this section, we firstly present the formula of adaptation policy rules; then we describe the approach for extracting adaptation rules from CCM; and finally, we discuss the benefit of utilizing optimized context view for resolving the conflicts in policy rules.

5.2.1 Context-Action Policy

The optimized context view generated from an AHP calculation provides a systematic end-users’ view of a task and its contexts, which can thus be applied as a reference for context-aware system design, i.e., specifying a set of operations that transform the task into an outcome, determining the context space that is “interesting” for monitoring or state detection, and designing the context-action policies for guiding responsive system behavior and invoking service composition and reconfiguration.

A context-action policy specifies when the context-aware adaptation should be performed, how an operation should be adapted and what to adapt, i.e., the adaptation targets. The policy can be described by a set of rules, using a policy specification language [70].

Considering the characteristics of context-aware adaptation, we formulate the policy rules as context-condition-action sets. Specifically, each policy rule is comprised of a set of context definitions, a condition body and an action body. In each policy rule, each
context definition expresses a specific context factor that the rule is interested in. The condition part consists of a logical expression involving the occurrence of factors specified in the context model. The rules are triggered by the context factors and are operated by performing the reconfiguration actions defined in the action part if the condition part is true. Formally speaking, a rule is an expression of the form:

\[
\text{context context.\textit{def}_1, \ldots, context.\textit{def}_i} \\
\text{condition \{ condition \}} \\
\text{action \{} \\
\quad \text{action}_1 \\
\quad \ldots \\
\quad \text{action}_j \\
\text{\}}
\]

For example, the following policy rule specifies that the room should be illuminated when the user is indoors:

\[
\text{context userEnterRoom} \\
\text{condition \{ light = \text{OFF} \}} \\
\text{action \{} \\
\quad \text{setLightON(light)} \\
\text{\}}
\]

In this example, the context definition \textit{userEnterRoom} denotes the event that triggers the adaptation. The condition part is evaluated to be true at the time the context event happens and the action part is executed.

A context definition \textit{context.def}_i can be either a basic context, a group of a basic contexts or an expression formed by a number of basic contexts. A basic context\((ct)\) is an expression of the form

\[
ct : ct_1 \& \ldots \& ct_n
\]

or

\[
ct : ct_1 | \ldots | ct_n.
\]
c_1 \& \ldots \& c_n \text{ represents the occurrence of all of the instances of } c t_1 \text{ through } c t_n, \text{ while } c t_1 | \ldots | c t_n \text{ represents the occurrence of at least one instance of the } c t_i s. \text{ Each element } c t_i \text{ represents a context event from the } what, who, when, where \text{ source. The following shows an example of basic context:}

\text{\textbf{context} } userEnterRoom : drOpen \& \ Lisa \& \ R205

In this example, the \textit{userEnterRoom} is the name of the context factor, and the expression defines this context factor is composed of an system event \textit{drOpen}, the user \textit{Lisa} and the location \textit{R205}.

The \textit{condition} body specifies the logical expression that should be evaluated in order for the action part to be executed. A \textit{condition} expression is of the form \( p_1 \circ p_2 \ldots \circ p_i \):

\text{\textbf{condition} } \{ p_1 \circ p_2 \ldots \circ p_i \},

where each \( p_i \) is a predicate of the form \( t_1 \circ t_2 \). \( \circ \) is a relation operator from the set \( \{=, \neq, <, >, \leq, \geq \} \), and \( \circ \) is a boolean operator from the set \( \{\&, |\} \).

The last part of a policy rule is the \textit{action} body with a list of actions. Each \textit{action} is an expression of the form \( a(t_1, t_2, \ldots, t_n) \), where \( a \) is an action symbol of \( n \) arguments:

\text{\texttt{action}\{
      \textit{action}_1(x_1, x_2, \ldots, x_i),
      \ldots,
      \textit{action}_j(x_1, x_2, \ldots, x_j)
    \}}

The actions represent a sequence of adaptation methods that should be invoked for the system to adapt when the condition of the rule evaluates to true. Specifically, the adaptation methods are available system actions which allow to tune parameters that change the behavior of a service and add, remove, or replace an ordered set of services by another one.
5.2.2 Specify Context-Action Policy with CCM

The CCM optimized view contains flows of operations that transform the task into an outcome, as well as a context space representing a subset of the environment that might affect the task execution process. Table 5.2 shows the general concept mappings from a CCM optimized view, which abstracts a task and its context space with structural representation, to the policy of context-aware adaptive activity, which specifies the judgement of the adaptation targets, the time for adaptive action and the reconfiguration of the resources associated with the action.

**Table 5.2:** Concept Mapping between CCM and context-action policy

<table>
<thead>
<tr>
<th>Optimized View</th>
<th>Policy Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Model</td>
<td></td>
</tr>
<tr>
<td>operation</td>
<td>action</td>
</tr>
<tr>
<td>arrow(opᵢ, opⱼ)</td>
<td>condition</td>
</tr>
<tr>
<td>condition</td>
<td></td>
</tr>
<tr>
<td>Context Space</td>
<td></td>
</tr>
<tr>
<td>event</td>
<td>context</td>
</tr>
<tr>
<td>activity</td>
<td></td>
</tr>
<tr>
<td>arc(ctᵢ, ctⱼ)</td>
<td>context definition</td>
</tr>
</tbody>
</table>

The mapping between the elements of a CCM task model and the actions/conditions of a policy rule is intuitive. The action part of a policy rule is derived from an operation element of the CCM task model; and the condition part of a policy rule corresponds to the prerequisites of the operation in the task model. Initially, each (condition, operation) pair in the task model corresponds to one policy rule; and the analyst specifies the context variable(s) that will trigger the execution of this rule. For example, the task model of green transportation (Figure 5.3) contains four operations, i.e., walking, biking, public transit and driving, its correspondent context specification should contain the following
four policy rules:

R1:  \textbf{context} CloseToStn  
\hspace{1cm} \textbf{condition} \{ PublicTransit = Available \} 
\hspace{1cm} \textbf{action} \{  
\hspace{1.5cm} \text{setTransportMethod(PublicTransit)}  
\hspace{1.5cm} \text{setCarbonEmission(1)}  
\} 

R2:  \textbf{context} RushHour  
\hspace{1cm} \textbf{condition} \{ bicycle = available \} 
\hspace{1cm} \textbf{action} \{  
\hspace{1.5cm} \text{setTransportMethod(Biking)}  
\hspace{1.5cm} \text{setCarbonEmission(0)}  
\} 

R3:  \textbf{context} Sunny, CloseToDst  
\hspace{1cm} \textbf{condition} \{ \}  
\hspace{1cm} \textbf{action} \{  
\hspace{1.5cm} \text{setTransportMethod(Walking)}  
\hspace{1.5cm} \text{setCarbonEmission(0)}  
\} 

R4:  \textbf{context} Night  
\hspace{1cm} \textbf{condition} \{ car = available \} 
\hspace{1cm} \textbf{action} \{  
\hspace{1.5cm} \text{setTransportMethod(Driving)}  
\hspace{1.5cm} \text{setCarbonEmission(10)}  
\} 

The condition and operation information in the CCM task model provides sources for specifying the condition and action part in policy rules, e.g., the “has bike” diamond and the “Biking” box in Figure 5.3 corresponds to the condition \textit{bicycle} = \textit{available} and the action \textit{setTransportMethod(Biking)} in \textit{R2}.  

The condition in the policy rules refers to either the condition (diamond) or the direct link between operations in CCM task model. A direct link arrow \( (op_i, op_j) \) represents a special condition that \( op_j \) cannot execute until the completion of \( op_i \), e.g., the action \( setCarbonEmission(0) \) is not executed until the action \( setTransportMethod(Bike) \) is completed.

The context part in the policy rules denotes the context that triggers the rule. It can be either a simple context dimension, e.g., time and location, or a complex event/activity in the optimized context view. Expertise of the policy language and the CCM framework is required for specifying the context part of the policy rules. The analyst should decide the scope of context domain and provide an explicit definition of each context variable, based on the contextual knowledge captured in a CCM optimized view.

For example, the weather conditions and the destination’s location are included in the optimized context view of green transportation (Figure 5.3). According to this model, the analyst may decide that the contexts of “walking” is that “the weather is sunny and the user’s current location is close to the destination”, as specified by R3. To derive the context part of R3, the analyst should associate two context variables (“Sunny” and “CloseToDst”) to the “Walking” action, and apply the optimized context view to derive detailed definitions of context variables. For example, according to the arc between “cell phone” and “sunny” in Figure 5.3, the following context definition can be derived for the context variable “Sunny”:

\[
\text{context Sunny : Cellphone } \rightarrow \text{WeatherReport}() = \text{SUNNY},
\]

which means that the “Sunny” situation is detected by the weather report in cellphone.

In the complex situation where the context domain is adjustable to include more or less contextual information, multiple policy rules may be specified for the same (condition, operation) pair. That is, the rule set can be extended to include more context variables and context definitions. Examples of this case are presented later in § 6.2.3; and its support for context reconfiguration is discussed in § 6.2.5.
5.2.3 Handling Rule Conflicts

The Context-Action policy provides a flexible mechanism for coordinating the behavior of multiple applications/services to achieve better performance and user satisfaction. However, the management of large sets of objects/services across diverse boundaries often leads to conflicting requirements, which subsequently materialize into policy rule conflicts.

Policy conflict occurs when the objectives of two or more policy rules can not be simultaneously satisfied. For example, among the five policy rules specified above for the green transportation case, R1 is in conflict with R3 when the contexts Sunny, CloseToDst and CloseToStn occur simultaneously, i.e. these two rules establish the reconfiguration actions setTransportMethod(PublicTransit) and setTransportMethod(Walking), which are mutually exclusive.

R1: **context** CloseToStn
   **condition** { PublicTransit = Available }
   **action**
   
   setTransportMethod(PublicTransit)
   setCarbonEmission(1)

R3: **context** Sunny, CloseToDst
   **condition** { }
   **action**
   
   setTransportMethod(Walking)
   setCarbonEmission(0)

Conflicts only occur if the policies are about the same action, on the same target but the objectives of the action are different. There are two approaches for resolving rule conflicts: 1) the system designer predetermines the priority of each policy; 2) the user’s
awareness and involvement are required for conflict resolution. Since the optimized view captures the optimized end-users’ view of contexts and their relative importance to the process of a task, the scores in the optimized view can be used to determine the priority value of each policy rule. The policy rules can be associated with a priority value to resolve conflicts between rules.

For example, the policy rules R1 and R3 given above derive from the optimized view in Figure 5.3. In the optimized view, the two contexts - CloseToStn and CloseToDst - in the policy rules map to the location factor “Transit Stn” and “Destination” in optimized view. As “Destination” has weight 0.76 while “Transit Stn” has weight 0.09, R1 is assigned a priority value greater than that of R2, so that the transport method is set to be “Walking” for the green transportation task, no matter the availability of public transit.

Compared to approaches that rely on the user to oversee the interdependencies of all adaptive actions and to specify the priorities of conflicting policies directly, this approach is more scalable in complex adaptive systems with a large number of rules specified by different end-users. In addition, the priority values on policies can be tagged with the corresponding task, which represents a functional characteristic to which the policies apply.

5.3 Summary

The process of cognitive context modeling can be divided into four steps (see § 4.1): data collection, model building, context space identification and context-behavior mapping. CCM provides a structural description of the process of a task and the complex objective / cognitive context surrounding the task. This chapter has presented the techniques from related area for manipulating the contextual knowledge collected in CCM.

Section 5.1 analyzed the structure of CCM and presented the algorithm that converts
the problem of context space identification into an optimization process, which integrates all context views and associates each objective elements with a score of relevance. The output of the optimization process forms a context space with interconnected temporal, physical and cognitive facts. The systematic context view makes it more easier and reliable for the task of selecting “interesting” facts in context-aware computing.

Section 5.2 demonstrated the use of CCM and optimized context space for specifying context-aware behavior. The section firstly presented the formula of context-aware rules, which are described by policy specification language and can be applied for the specification of context-aware actions. A context-aware rule is composed of three parts: context definitions, condition and action. We used the optimized view of “green transportation” as an exampled to illustrate the benefit of CCM for understanding context-aware activities and for resolving the conflicts in action.

In summary, CCM captures a wealth of cognitive information for context-aware activities. The abstraction and structural description of individual views allows the techniques of statistical data analysis, such as clustering and optimization, being applied to facilitate context space identification. With formal definitions of context-aware action, it is also possible to reduce human effort in converting the CCM elements to a set of rules for guiding responsive action in context-aware computing. The next chapter will describe in detail the automated tools we developed for assisting the whole process of cognitive context modeling.
We now have a well defined framework serving as a context elicitation model which fits into the gap between the infinite, subjective context perception and the finite, objective context-awareness design. Chapter 4 presented the conceptual structure and main components of the cognitive context modeling framework for cognitive context elicitation. Chapter 5 described the process of context space identification and context specification for context-aware computing. The toolkit which supports various tasks within this framework was described in Chapter 7. Throughout these chapters, examples from the domain of green transportation are used to illustrate various aspects of our work, which makes the characterization of problem and our methodology easy to understand.

The next step is now to populate the model with data from real world situations to evaluate the applicability of this model. This chapter chooses two topics of great interest in context-aware computing as case studies to further evaluate the performance of our methodology for context modeling. § 6.1 describes the case study of smart meeting room. § 6.2 describes the case study of context-aware power management. § 6.3 presents a set of evaluation criteria for context modeling and summarizes the results of case studies.
6.1 Case Study 1: Smart Meeting Room

The setting for this case study is supporting context-awareness in a small meeting room. A description of the room and several scenarios have been presented in § 3.1. People involved in a meeting need to deal with various tasks, e.g., scheduling, device operation and data sharing, which require a lot of coordination effort. The main goal of a smart-meeting-room system is to reduce the human effort and make meetings more productive. Detailed goal analysis of the smart meeting room has been described in Chapter 3.

The problem of the smart meeting room is well-known in the area of context-aware computing and has been used regularly in the field of context modeling as a representative ubiquitous environment with many devices and services equipped to facilitate human-human and human-device interactions. Figure 6.1 shows an ontology-based context model of a prototype smart meeting room. Similar context models of smart environment (Figure 2.7) have also been discussed in § 2.3.

6.1.1 Data Collection

During the process of context-aware system development, context modeling focuses on the unified and precise description of environmental factors associated with a human task. To this end, all sources of knowledge about the situation should be collected and the knowledge should cover all dimensions of the world. In this case, we collected knowledge about the meeting task from three types of data sources:

- interview transcripts ([77]),
- scenarios (see § 3.1 and [76]),
- technical reports of smart meeting systems (EasyLiving [13, 1], iRoom [52, 3]),

so that the knowledge-base of context information is enriched to contain knowledge of different roles in a meeting activity, e.g., participants, experts and system developers.
The same materials were used in our goal-based analysis for extracting user and system goals (§3). Similar to the iterative process that we applied for extracting functional and non-functional goals (see Figure 3.8), the environmental factors of a context-aware task were extracted from raw materials, and were verified through iterative bidirectional mappings. The verification process was conducted by a person who did not participate in the extraction process.
6.1.2 Model Building

As described in § 4.2, the cognitive context modeling process is task-oriented. A task model, which specifies the goal and a flow of operations for transforming the goal into an outcome, is the centre and driving force of context modeling. If the task is not defined explicitly, it is impossible to guarantee the validity of a context domain. Figure 6.2 shows a generic task model for a smart meeting room, which is sufficient for capturing the main operations involved in a meeting process.

The main goal of a meeting room is to run meetings. A meeting refers to an act or process of coming together for information sharing and delivery [Merriam-Webster]. According to the different phases of a meeting procedure, the task of smart meeting can be divided into several operations: 1) meeting room initialization, 2) progress monitoring, 3) room reconfiguration, and 4) follow-up actions. The identification of these four major operations was based on the different scenarios (S1 - S3) of the meeting process (see § 3.1). For example, according to S1 and S2, “meeting room initialization” refers to the actions of scheduling a meeting and starting up a meeting room according to the schedule. And according to the results of goal analysis (Figure 3.9), the expected outcome of the flow of operations is “efficient information delivery”, i.e., the meeting should be efficient and productive.

![Figure 6.2: The task model of smart meeting room](image_url)

Context is ubiquitous during the whole process of a meeting. Each identified contextual factor should be able to establish a connection to this task. The CCM framework emphasizes that task participants should be the main source for identifying contextual factors.
factors, and that views of task participants should be collected and abstracted independently.

In this case study, we built all the context view models based on the extracted information from raw data, i.e., the scenarios, interview transcripts and technical reports. Each view model corresponds to one source of data input and was verified through iterative and bidirectional mapping to ensure that it is consistent to the original materials. Each context view is independent, reflecting the perspective of one group, since the scenarios, interviews and prototype systems belong to different groups of people. The various context views constitute the CogCt component of CCM. The CCM toolkit described in Chapter 7 was used for creating and editing the context views.

**Figure 6.3:** A context view of the “meeting” task

Figure 6.3 shows the context view we built based on the information contained in the meeting scenarios (§ 3.1). Compared to the scenario description of contexts involved in a meeting, the context view model is more abstract. For instance, specific individuals, e.g., Alex and Bob, were not considered as context elements. Instead, their roles in the task were identified as possible instances of human contexts. The abstract representation improves generality and reusability of context model.
The context view model extracted not just context factors of different types, but also factor interactions and relationships to reflect the rich details narrated in scenarios. The context view model uses undirected lines to denote the relationships. Such relationships form a special dimension of context space. The connections in the context view model can be divided into three types:

- subject of behavior,
- object of behavior,
- user of device.

For example, in Figure 6.3, the “scheduling” activity connects to three time elements and two human elements. These connections describe the key information of the scheduling scenario that “the presenter and coordinator schedule a meeting according to expected meeting time, duration and time constraint of potential attendees”. Similar to human activities, the events are regarded as device behaviors. Devices are the subject of events. For example, in Figure 6.3, the “email notification” event is connected to “server” and “scheduled time”. It specifies that “the server receives notification about scheduled meeting time”. Together with the connection between “presenter” and “server”, the context view model covers the key contexts of meeting scheduling.

The “scenarios” view model, as a cognitive element, was included in the CogCt layer of CCM. Following the same procedure, we built the other context view models, i.e., interviews, EasyLiving and iRoom. All the view models in CogCt are independent, emphasizing the variance of individual’s contextual cognition for the meeting task.

Based on the context views, the CCM toolkit aggregated all objective elements into the ObjCt layer and formed a subset of the physical world which potentially affects the process of smart meetings. The resulting model is shown in Figure 6.4. Since the view models were extracted from various sources, it usually needs to be refined by the analyst to remove duplicate elements and resolve conflicting use of terminologies. In order to avoid the elicitation of duplicate or conflicting elements, the CCM toolkit keeps track of
all the identified elements and performs similarity checking when a new element is added. The methodology proposed by Shaw and Gaines [100] can also be applied for classifying and handling the conceptual differences.

![Cognitive context model of smart meeting room](image)

**Figure 6.4:** Cognitive context model of smart meeting room

### 6.1.3 Model Analysis

In our everyday life, people notice and integrate a vast range of cues and then react appropriately. Similarly, the success of a context-aware system depends on the obser-
The smart meeting CCM contains three components: task, CogCt and ObjCt. At the centre of the smart meeting CCM is a task model, which provides the basis for the identification of relevant contexts. In this case study, CogCt integrates four subjective context views of the task, and it can be extended to include more context views. ObjCt provides a collection of objective elements mentioned in CogCt. Overall, the CCM-based context modeling identified a total of 42 objective factors and established 54 connections between them.

In the CogCt component, each context view identifies objective factors along six dimensions of the physical world: time, location, human, device, activity and event. Table 6.1 summaries and compares the context views according to the size of each dimension.

**Table 6.1: Summary of the smart meeting CCM**

<table>
<thead>
<tr>
<th>CCM</th>
<th>Size of context view (# of factors)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CogCt</td>
<td></td>
</tr>
<tr>
<td>Scenarios</td>
<td>[76]</td>
</tr>
<tr>
<td>Interviews</td>
<td>user</td>
</tr>
<tr>
<td>EasyLiving</td>
<td>[13, 1]</td>
</tr>
<tr>
<td>iRoom</td>
<td>[52, 3]</td>
</tr>
<tr>
<td>ObjCt</td>
<td></td>
</tr>
</tbody>
</table>

The result in Table 6.1 shows that no single context view covered all objective contexts.
Chapter 6. Case Studies and Evaluation

of the “meeting” task. There was a variation in the size of context views. For example, the “scenarios” context view covered 64% of the elements in ObjCt, while the “EasyLiving” context view only covered 33%. A context view reflects a person or a group’s selective interest, preference, experience, cognition, etc. Through context view modeling, we found that different groups had their own focus of interest, leading to different projections of context space of the same task. For instance, the EasyLiving project focused on human/device interaction, thus the contexts in EasyLiving were mostly meeting devices. Yet the iRoom project showed more interest in participant interactions in a smart meeting room, and identified more contexts of the “event” and “activity” dimension. Overall, by modeling and integrating context views from different task participants, CCM expanded the context space of smart meeting task, and provided a systematic depiction of the surroundings.

As has been analyzed in § 2.3, traditional object- or logic-based context models lack the capability to deal with dynamic and complex context data, for instance, the collaboration between subjects or objects. Although existing graphical models, such as ontology-based context models [112, 103, 48] and ORM-based context model [45], can capture the richness of context information, they are inefficient for handling context changes and context evolution. Compared to these context models, CCM allows the incremental evolution of context space through the aggregation of independent context views. And CCM is the only context model that takes multiple views or preferences into account.

Table 6.2 shows the structural difference between CCM and other models, i.e., the ontology-based context models (Figure 2.7, Figure 6.1) and the i* models of smart meeting room (§ 3.4).

The focus of ontology-based context modeling is the definition of domain concepts, properties, and consistency and validity rules. An ontology model consists of two layers: upper ontology and domain ontology. Elements in a domain ontology are instances of a
Table 6.2: Different models of smart meeting room

<table>
<thead>
<tr>
<th></th>
<th>ontology [48]</th>
<th>i* [77]</th>
<th>CCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>basis</td>
<td>project</td>
<td>goal</td>
<td>task</td>
</tr>
<tr>
<td>components</td>
<td>upper ontology</td>
<td>SD model</td>
<td>task model</td>
</tr>
<tr>
<td></td>
<td>domain ontology</td>
<td>SR model</td>
<td>context view model</td>
</tr>
<tr>
<td>elements</td>
<td>person, device, resource, actor, softgoal, hardgoal</td>
<td>person, device, time, location, activity, event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>location, activity</td>
<td>task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>file, building</td>
<td></td>
<td></td>
</tr>
<tr>
<td>connections</td>
<td>subClassOf</td>
<td>contribution, dependency</td>
<td>relation, relevance</td>
</tr>
</tbody>
</table>

Concept in upper ontology. The basis of ontology modeling is project settings. System designers use the upper ontology to define object categories of the project, then use the domain ontology to identify the specific factors existing in the project domain. In ontology model, there are only top-down connections, elements of the same layer are independent to each other.

The basis of i* modeling is goal refinement. It requires the participation of potential users to explore the subgoals, tasks and resources required for goal achievement. The emphasis of i* modeling is to explore the design space of a software system. It uses SD model and SR model to elicit two types of context information, i.e., actor dependency and user goals. In i* models, an operation belongs to an actor and has positive or negative contributions to some of the actor’s goals.

The CCM framework supports task-centred construction of context space. The theoretical basis of the framework is Activity Theory’s analysis on the hierarchical structure of activity. Similar to i* and ontology models, CCM explores the problem domain using abstracted structures and labels to facilitate knowledge representation and processing. In CCM, a task (system activity) is described as a chain of operations (system actions).
directed to an outcome. CCM uses context view models to capture task participants’ individual cognition of task processing. The predefined six-dimensional structure of context space facilitates the information extraction process.

Overall, compared to existing context modeling approaches, CCM does not support the identification of all resources of a closed environment, but emphasizes the complex interdependencies between environmental factors, and through the combination of task modeling and context view modeling, it helps answering the how and why questions about context selection and context-aware activity.

**Context-aware action planning**

![Optimized View](image)

**Figure 6.5:** An integrated context view of the “meeting” task

The CCM structure uses independent context views to capture the individual cognitive variance of context-awareness. The cognitive space is flexible and extensible. However, context-aware applications usually require a unified context space for monitoring
and computing. The CCM toolkit, described in Chapter 7, provides an “AHP” calculation module for the integration of context views. The calculation needs two parameters, i.e., the weighting of each view, and the ranking of context elements in each view. In this case, we assume that all views have the same weight. Since the context views were constructed by extracting information from given materials (e.g., scenario, project report, interview transcript), where the relative importance of each element was not explicitly specified, we assume that all views consider their identified elements as highly relevant contexts that have the same influence power to the task ($score = 9$), while the elements unmentioned should have very low influence power ($score = 1$). Figure 6.5 shows the context view resulting from AHP calculation. It integrates the information contained in four different context views and forms a unified context space for the smart meeting task. The context elements of each dimension are arranged according to their effect on the task process.

The optimized context model is composed of a task model and a unified context space of the task. The task model depicts the chain of operations that supports efficient information delivery in smart meeting room, while the context space identifies the environmental facts which affect the manner and timing of these operations. According to the definition of context-action policy (§ 5.2) and the concept mappings between CCM elements and policy rule definitions (Table 5.2), we extracted information from the context model and formed a set of context-action rules. The following shows an example of the rule set:

\begin{verbatim}
R1: context StartMeeting
    condition { MeetingRmStatus = IDLE }
    action {
        initializeMeetingRm()
        setMeetingRmStatus(READY) 
    }
\end{verbatim}
The transformation from context model to context-action specification was intuitive. The specification contains four policy rules (R1 ~ R4), corresponding to four operations identified in the task model. Each rule specifies a series of actions, a prerequisite for the execution of actions, as well as the contexts for the invoking of actions. The prerequisites were derived from the condition elements in task model. Rule R1 defines that when a meeting is about to start, an idle meeting room should be activated and preparing for the meeting. R2 and R3 illustrate how meeting room behavior should be adjusted according to different needs of users. R4 defines that when the meeting ends, the meeting room should complete followup actions and then shut down the equipments. For each policy rule, if the inference engine detects the occurrence of contextual events, it will evaluate the condition and wait the answer to be true, then it will continue with the subsequent actions. Compared to current expert systems or planning approaches [6, 18]
which tend to apply a stationary rule set and solve given problems in a recursive fashion, the rule-based inference engine is more flexible, allowing corrective actions at runtime. The context-action rule set is extensible and is able to deal with uncertainty as well as individual needs of different users.

The most tricky part of generating rules from context model was to specify the context of each operation. In the policy rules, the statements, such as “StartMeeting”, “StartPresentation”, “InterruptPresentation”, “EndOfMeeting”, were used to describe the situation that triggers rule execution. Explicit definitions of the situation can be derived based on the CCM integrated context view (Figure 6.5) which identifies the scope of context space as well as context dependencies. For example, the following expression defines that a meeting is about to be held in meeting room, if the activity sequence “scheduling, “preparing slides” and “email notification” occurs.

\[ c_1: \text{context StartMeeting : scheduling \& slides\_preparation \& notification} \]

In the context model, these activity and event elements are connected to other elements, e.g. person, device, etc.. Based on these dependency connections, we can further refine the definition of scheduling, slides\_preparation and notification.

\[ c_{1.1}: \text{context scheduling :\{presenter\&attendees\}→scheduling(expected\_time,duration,constraint)} \]

\[ c_{1.2}: \text{context slides\_preparation :presenter→preparing\_slides()} \]

\[ c_{1.3}: \text{context notification :server→notification(scheduled\_time)} \]

These expressions illustrate that the granularity of context definition can be refined to any level that is appropriate. The expressions shown above use the form object→method(parameter) to represent the engagement of different factors in human activity or system event. Object refers to persons or devices, method refers to an activity or event, and the mark “→” indicates that “person carries out an activity” or “device invokes a service event”. The parameter of an expression is optional, it can be elements of time and location. \( c_{1.1} \) defines that presenter and attendees should cooperate in scheduling a meeting according
to the expected meeting time and the time constraint of the meeting participants. $c_{1.2}$ indicates who should prepare slides for the presentation. $c_{1.2}$ defines that server should invoke the meeting notification service.

\[c_{2}: \text{context StartPresentation} : \text{scheduled_time} \& \text{attendees_seated} \& \text{presenter_ready}\]

\[c_{2.1}: \text{context attendees_seated} : \text{attendee}.\text{position} = \text{seat}.\text{position}\]

\[c_{2.2}: \text{context presenter_ready} : (\text{presenter}.\text{position} = \text{meeting_rm} \& \text{display} \rightarrow \text{UIgeneration}() \]
\[\& \text{presenter} \rightarrow \text{voice}.\text{gesture}()) | (\text{presenter}.\text{mobile_device} \rightarrow \text{UIgeneration}() \& \]
\[\text{presenter} \rightarrow \text{remote}.\text{control}((\text{presenter}.\text{mobile_device}))\]

$c_2$ defines that presentation is started if it is the scheduled meeting time, attendees have been seated and presenter is ready. $c_{2.1}$ and $c_{2.2}$ specify the elements of $c_2$ in more detail so that it can be processed by an inference engine. $c_{2.1}$ illustrates how to determine the attendees are seated. In the context definition, the form $\text{object}_A.\text{object}_B$ is used to represent that $\text{object}_B$ belongs to $\text{object}_A$. $c_{2.2}$ defines that a presenter is ready, if he has interacted with meeting devices either through voice/gesture command or through remote control on his mobile device. The alternatives specified in $c_{2.2}$ are derived from the information identified in smart meeting CCM. It makes the system adapt to any conceivable environment variables and allows room behavior to be adjusted according to the different needs of different users.

\[c_{3}: \text{context InterruptPresentation} : \text{remaining_time} \& (\text{interaction} | \text{discussion})\]

\[c_{3.1}: \text{context interaction} : \{\text{presenter}, \text{attendee}\} \rightarrow \text{interaction}() \& \text{attendee} \rightarrow \text{remote}.\text{control}() \& \]
\[\{\text{attendee}.\text{mobile_device}, \text{presenter}.\text{mobile_device}\} \rightarrow \text{multi}.\text{control}()\]

\[c_{3.2}: \text{context discussion} : \text{attendee} \rightarrow \text{discussion}() | \text{attendee} \rightarrow \text{sketching}(\text{whiteboard}) | \]
\[\text{attendee}.\text{mobile_device} \rightarrow \text{data}.\text{migration}(\text{server})\]

$c_3$ illustrates how an interruption to the presentation can be detected and applied to trigger the rule R3. It defines that the presentation can be interrupted if the time
allows and there is a request for interaction or discussion. $c_{3.1}$ and $c_{3.2}$ provide detailed specifications for the *interaction* and *discussion* activities. Based the context definition and context-action rule, the meeting room is able to provide different configurations according to different situation of the user. Configuring rules in this manner makes the system attentive and adaptive, thus the user’s attention can remain focused on the work being done rather than on the mechanics of interaction.

\[c_4: \text{context } EndOfMeeting : \text{remaining\_time} = 0 \& \text{empty\_room}\]

\[c_{4.1}: \text{context } \text{empty\_room} : \text{presenter\_position} \neq \text{meeting\_rm} \& \text{attendee\_position} \neq \text{meeting\_rm}\]

c4 and c4.1 define the context that triggers follow-up actions after meeting, such as storing and distributing meeting record, shutting down devices, etc.. Executing this rule will reduce the user’s workload and make the meeting more productive.

In summary, the context definitions specify the situational variables which can be monitored by the inference engine to trigger meeting room reconfiguration and adaptive system behavior. The adaptive and attentive system behavior will reduce user’s work on coordination and manipulation and will lead to effective information delivery. Since the context-action rules are derived according to the content of context model, all the context specifications can be traced back to user’s conceivable environment variables and to user’s goal of context-aware action.

### 6.2 Case Study 2: Power Management

With more and more computing devices being deployed in buildings and leading to the steady rise in electricity consumption, there is a pressing need to reduce overall building energy consumption. Shutting down idle PCs contributes to energy saving and it motivates the research of using user context to enable effective power management and to reduce energy consumption.
Harris et al. [43] proposed a CAPM (context-aware power management) framework that employs Bayesian Networks to support prediction of user behaviour patterns from multi-modal sensor data for effective power management. They used coarse location in the form of proximity detection using Bluetooth, microphones as an interesting context for predicting user behavior. They found that “determining user context is the most challenging part of CAPM”. “User location alone is insufficient for context-aware power management and further context is needed to determine fine-grained user behaviour for effective power management” [43].

Harle et al. [42] studied how a set of 40 people used the building they worked in, with an emphasis on whether building energy consumption can be reduced through dynamic optimization based on input from a building-wide location system. They found that 75% of the average user’s working day was spent in the vicinity of their computer, and estimate savings of around 140Wh per PC per day compared to a typical scheme that keeps machines on for the working day (and off otherwise). They emphasized that CAPM systems should be designed to reduce energy consumption under the constraint of having an invisible effect on users.

In this case study, we will apply the CCM framework to elicit the user contexts of energy saving activities and analyze the solutions for CAPM.

6.2.1 Data Collection

The setting of the case study is the usage of computers in a university-based research lab. The lab has 104 computers and each computer is regularly used by one person. The computers were all Linux application terminals, and the users had similar academic background and workload. The lab allows 7-day 24-hour on site and remote access. The subjects of our study were graduate students who have flexible work schedule. We monitored and logged the energy usage of each computer over the course of seven weeks. We also collected data about their energy saving activities through individual interview
6.2.2 Model Building

The centre of the power management CCM is a task model that explicitly specifies an operation chain and the outcome of the operations. The task model reflects the system designer’s view of the problem domain. During the phase of task analysis, the designers and users may work together through an iterative process to resolve any disagreements on the task process. However, the task model needs to be finalized before deciding the context domain, so that the context elicitation/modeling process is focused on identifying the factors triggering state transitions. As shown in Figure 6.6, power management can be divided briefly into three operations: power status checking, startup and shutdown. Shutting off the computer at appropriate times will contribute to “energy saving”. The “appropriate time” should be decided based on the analysis of user context, e.g., the user is using the device, the user is about to use the device, etc.. Based on a systematic user context model, a CAPM system can thus reduce energy consumption while avoiding interrupting and disturbing users.

![Figure 6.6: The task model of power management](image)

Current CAPM systems usually depend on user’s location to control the status of computer. It is making use of the statistical coincidence of computer status and user’s location. A dependable power management should build upon the deep understanding of why the user uses or does not use the computer and how the factors affect user’s activity. During a three-month in situ observation, we recorded the power state changes of each computer in the lab. Figure 6.7 shows the state changes of some of the computers. Each
line plot shows the power-off ratios of one computer at the respective time intervals. The x-axis represents the time factors ranging from Monday morning to Sunday midnight. The power-off ratio represents the observed power-off frequency divided by the total number of observation periods. Since each computer is used by one person, the power state changes of a computer can be regarded as the person’s computer usage pattern. We observed that different users have various computer power management behavior. Some users only occasionally use the computer, e.g., view11 and view49. Some line plots reflect regularities of computer usage. For example, view88 always shuts down the computer for weekends, and view19 usually shuts down the computer after work. However, most users, such as view10 and view98, almost never shut down the computer to save energy.

**Figure 6.7:** Power state changes over time

The line plots in Figure 6.7 can be regarded as special context view models, in which the time factors are potentially the contexts of the CAPM task, while the power-off
ratios measure the level of relevance of each factor to the CAPM task. For a large amount of observation data, we can firstly conduct mathematical analysis to identify the key patterns of people’s computer power management behavior and to explore the main context variables of the energy saving task. In this case study, we conducted simple K-means clustering on the observation results of 104 computers. Figure 6.8 shows the result of 3-means clustering (the ‘+’ lines). The dashed lines in the figure indicates the standard deviation values of respective clusters. A low standard deviation value implies the convergence of context views in the cluster. The clustering calculation classifies the power management activities into three categories: 1) the computer is always power-on \((\text{score} \approx 0 \text{ for most time intervals})\), 2) the computer is usually power-off \((\text{score} \approx 1 \text{ for most time intervals})\), and 3) computer’s power state varies with time.

![K-Means Clustering of Power Save Views (CS Lab)](image)

**Figure 6.8:** K-means clustering on context views \((k=3)\)

The result of in situ observation shows that about 68.3% of the computers were
kept running at all time (category 1), 18.3% of them were kept running during working
hour (category 3), while only 13.4% of the users always put idle computers power-off
(category 2). As described in § 6.2.1, all the computers had similar workload, which
implies that if appropriate power management were used to control the computers of
category 1, the energy consumption can be greatly reduced. Designing interventions,
e.g., regulation, policy, program, measure, etc., to change or influence the user’s power
management behavior is one approach to improving energy efficiency [117]. Another
approach is to develop a CAPM system that manages the computer’s power state to
maximize the energy saving and meanwhile minimize user’s intervention, i.e., the user
contexts are captured and used for the computer’s power management so that the energy
consumption is reduced while user’s behavior remains unchanged.

The success of a CAPM system depends on the systematic analysis and effective use of
physical and subjective contexts of the power management task. Our in situ observation
reveals the existence of diverse perspectives on computer power management. Obviously,
“location alone is insufficient for context-aware power management” [43]. The category
3 in Figure 6.8 exhibits the pattern of computer state changes along the time dimension
of context space, which suggests that the time factors, such as working hour, weekends
and holidays, may affect the power state of computers. Meanwhile, state changes of
the computers in category 1 and category 2 do not show any regularity, implying that
other personal and social elements should be explored to make the CAPM system more
effective.

In order to retrieve personal cognitive knowledge of computer power management,
three students, each from one category (Grad_1 from C3, Grad_3 from C1, Grad_4 from
C2), were interviewed individually. The author of this thesis acted as the context analyst
and conducted the interviews. At the beginning of each interview, the analyst introduced
the task model of power management, so that the interview focused on exploring the
factors that affect a computer’s power state transition. Then the interviewee used the
CCM toolkit (see Chapter 7) to create a context view model. When an interviewee was editing his/her context view, the analyst sat behind and provided assistance whenever requested by the interviewee. The analyst encouraged the interviewee to answer how/why questions and to provide detailed description when a new context element or connection was added to the model.

The interview result is shown in Figure 6.9. Each view model briefly describes an interviewee’s conceivable environmental variables that have major impact on their computer usage. The undirected links represent the connections between contextual variables. The detailed description of connections and variables is saved in the CCM file (as shown
in Figure 6.10) but omitted in the graphic model (Figure 6.9). During the phase of model analysis and context specification, analysts and system designers are allowed to view the detailed descriptions.

![Image of XML representation of sSM for power management]

**Figure 6.10:** The XML representation of CCM for power management

The variability in context views demonstrates that people might have different opinions on how a computer’s power state should be managed, even if they have similar background and situation (all interviewees were graduate students and computer users of the research lab). For instance, the context view model *Grad_1* emphasizes that the user’s working hours as well as the staff’s maintenance schedule are the main factors that determine the computer’s power state. Yet in *Grad_3*, the power state of computer is not managed on a regular basis, instead, computer-related user activities, e.g., programming, editing, and remote access, decide whether the computer will be on or off. The individual variance implies that simple and unitary contexts in CAPM can not meet the diverse needs of users. Therefore, a context model that cover a variety of major environmental variables and a context-aware strategy that address user’s individual requirements are required to support effective energy saving.

### 6.2.3 Model Analysis and Context Specification

Contexts affect behavior and decision making, however, it is usually hard for system designers to determine the scope of contextual factors and their actual influence on task
execution. As has been illustrated in Figure 6.9, the CCM context view models allow individual task participants to abstract contextual facts from various sources and rank the contexts according to their relative importance to task execution. With limited numbers of contextual factors identified and each factors associated with a score, it is possible for analysts to apply optimization techniques to the user data and generate an integrated context view.

Figure 6.11: The optimized view of power management

Figure 6.11 shows an optimized context space generated by AHP-based optimization algorithm (see § 5.1). The model integrates the information from context views of all aspects to form a comprehensive context space that contains contexts of various sources as well as subjectively conceived context dependencies. According to the optimized view, the computer’s power state is potentially affected by many different environmental factors, such as the computer’s running services, the user’s activity, location and schedule, etc.

In each context view, the context elements along each dimension are ranked according to their relative importance in affecting the task process. As a result of the AHP calculation, each context element in the optimized view has an optimized weight, which provides reference for the designers to decide the context domain of the system. For
example, in the “time” dimension, the factor “schedule” has the highest weight, which implies that user’s working schedule should be considered as a context for the CAPM system.

The fact that a computer’s power state is relevant to the user’s working schedule has also been demonstrated in the cluster (3) of Figure 6.8. Based on the task specification shown in Figure 6.6, we can specify the context-action rules as the following:

R1:  \textbf{context} StartWork  \\
\textbf{condition} \{ \text{Running == NO} \}  \\
\textbf{action} \{  \\
\quad \text{turnon()}  \\
\quad \text{setEnergySaving(0)}\}

R2:  \textbf{context} FinishWork  \\
\textbf{condition} \{ \text{Running == YES} \}  \\
\textbf{action} \{  \\
\quad \text{shutdown()}  \\
\quad \text{setEnergySaving(P \cdot t)}\}

Each rule’s condition part corresponds to the condition element in the task model, and the actions correspond to the operations in the task model. The “turnon” and “shutdown” actions are not executed unless the condition part is evaluated to be true. The context part of each rule defines the situation that triggers rule execution.

Compared to “condition”, the contexts are implicit and fuzzy. The definition of “StartWork” and “FinishWork” depends upon both the aspect of user activity and the limit of computing resource. For example, the working schedule might vary daily according to the user’s workload and condition. However, as a starting point, the working schedule can be simply defined as weekly patterns.

The context views illustrated in Figure 6.7 use “power-off ratio” as the score of each time slot, indicating how the factors affect computer power management. AHP calculation can be applied to the view models for optimizing the relative weight of each time
slot. Those views, which have exhibited the time-based power management patterns (see category 3 in Figure 6.8), demonstrate the time-awareness for power management. Therefore, such views are assigned higher view weights in AHP optimization in order to reduce the interference from other contextual factors. The result of AHP calculation is shown in Figure 6.12. The x-axis represents the time factors ranging from Monday morning to Sunday midnight. The line plot shows the optimized weight of respective time factor for the energy saving task.

![AHP Optimized View (CS Lab)](image)

**Figure 6.12:** Optimizing time factors for context-aware power management

The optimized view can be used for identifying the “interesting” time contexts of computer power management activity. As shown in Figure 6.12, each context element’s weight measures the relevance of the element for the energy saving goal. The time element whose optimization weight is greater or close to 0.3 might be a potential factor contributing to the computer’s power-off state, and vice versa. At some point there are explicit value changes, which facilitates the identification of time-sensitive power management activity. For example, compared to “Wed 08”, the optimization weight of
time factor “Wed_09” dropped by 17%, which implies that the users are more likely start using computer at 9AM on Wednesday. Based on this analysis, a set of coarse-grained time context can be identified, as shown in Table 6.3. These time factors are relevant to time-sensitive activities, i.e., “start work” and “finish work”, according to Figure 6.12.

Table 6.3: Time contexts of computer power management

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thr</th>
<th>Fri</th>
<th>Sat</th>
<th>Sun</th>
</tr>
</thead>
<tbody>
<tr>
<td>start work</td>
<td>8am</td>
<td>8am</td>
<td>9am</td>
<td>9am</td>
<td>8am</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>finish work</td>
<td>7pm</td>
<td>9pm</td>
<td>9pm</td>
<td>8pm</td>
<td>7pm</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

According to the observation and optimization results in Table 6.3, we can define the context elements in R1 and R2 as the following:

\[ c_1: \text{context } \text{StartWork} : \text{Mon}_8am|\text{Tue}_8am|\text{Wed}_9am|\text{Thr}_9am|\text{Fri}_8am \]

\[ c_2: \text{context } \text{FinishWork} : \text{Mon}_7pm|\text{Tue}_9pm|\text{Wed}_9pm|\text{Thr}_8pm|\text{Fri}_7pm \]

The policy rules and corresponding context definitions illustrate how computer’s power state should be adjusted according to the working hours of users. The policy rule R2 specifies that a running computer should be shutdown when the user finishes work. The context “FinishWork” is defined as the user’s getting off work schedule (see \( c_2 \)). The specification reflects the user needs of category 3 (see Figure 6.8). Applying the rules in power management will improve the efficient of energy saving and avoid interfering users. However, for computers of category 1 or category 2, the rules might bring about the risk of disturbing users. As has been described in the context views, the users of category 1 prefer to keep computer always running to facilitate access. Shutting off computers will possibly affect their work. And for those who do not use the computer regularly, introducing R1 for power management will result in increased energy consumption.
Due to the existence of different needs and preference for CAPM, the optimal rules should be concerned about user experience and individual cognition. According to the optimized context view shown in Figure 6.11, in addition to the working schedule, user and staff activities (e.g., system maintenance, remote access) are also relevant to computer’s power management activity. The following rules demonstrate the application of CAPM for energy saving considering user’s different needs at different situations.

R3: **context** MaintenanceSchedule, SystemMaintenance
   **condition** { Running == NO }
   **action** {
      turnon()
      setEnergySaving(0)
   }

R4: **context** SystemMaintenance, MaintenanceCompleted
   **condition** { Running == YES }
   **action** {
      shutdown()
      setEnergySaving(P \cdot t)
   }

R5: **context** RemoteAccess
   **condition** { Running == NO }
   **action** {
      turnon()
      setEnergySaving(0)
   }

R6: **context** RemoteAccess, DisconnectAccess
   **condition** { Running == YES }
   **action** {
      shutdown()
      setEnergySaving(P \cdot t)
   }

Computers are assumed to perform different tasks even though they are used by the same users. R3 and R4 illustrate how system maintenance conducted by lab staff affects
the computer’s power management activity. R5 and R6 define the power management activity that satisfies the users who prefer to access the computers remotely. The reference to specific computing task is necessary, if the power management should adapt individually to specific access. Compared to R1 and R2 which adapt power management activity only to the working hours, the context-action rule that relates to specific user activity improves the granularity of context specification and makes CAPM more accurate and effective. However, different from the intuitive context variable time, the contexts, e.g., SystemMaintenance and RemoteAccess, identified in the above rules are more sophisticated. According to the optimized context model which integrates individual knowledge and cognitive context information, we can define the context variables as the following:

\[ c_3: \text{context } MaintenanceSchedule : staff . computer \rightarrow calendar ( maintenanceStart ) = CURRENT \]
\[ c_4: \text{context } SystemMaintenance : staff \rightarrow maintenance() \]
\[ c_5: \text{context } MaintenanceCompleted : staff . computer \rightarrow calendar ( maintenanceEnd ) = CURRENT \]
\[ c_6: \text{context } RemoteAccess : user . laptop \rightarrow remoteaccess() = CONNECT \]
\[ c_7: \text{context } DisconnectAccess : user . laptop \rightarrow remoteaccess() = DISCONNECT \]

The context definitions specify in detail how the independent environmental elements interact with each other and constitute the complex contextual situation. All the definitions can be traced back to user context views. \( c_3, c_4 \) and \( c_5 \) indicate that calendar events and lab staff’s activity should be applied to inferring the situation of system maintenance. \( c_6 \) and \( c_7 \) illustrate that the remote access event of user’s laptop implies the lab computer’s remote access status.

Since the context-aware engine holds all the policy rules, it is possible at any time to create new policy rules and expand the rule set. The rules for dealing with occasional remote access and system maintenance can be viewed as a complement to time-sensitive power management rules. For the users who prefer to keep computer always running to
facilitate access, applying the rules to computer power management will reduce energy consumption while avoiding interfering their access to the system.

In the same way, the fact that user’s local activity affects computer’s power management can be described with the following rules:

R7: context StartLocalAccess
    condition { Running == NO }
    action {
        turnon()
        setEnergySaving(0)
    }

R8: context EndLocalAccess, NoRemoteAccess
    condition { Running == YES }
    action {
        shutdown()
        setEnergySaving(P \cdot t)
    }

R7 illustrates that when the user is about to use the computer, CAPM should make sure the computer is power-on to allow user access. R8 illustrates that when user’s local access terminates and there is currently no remote connection, the computer should be power-off for energy saving. According to the specification of R7 and R8, the computer’s power state should be managed based on the change of user’s actual computer usage. Compared to the rules which emphasize how user’s working hour affects power management (e.g. R1 and R2), adapting the power management actions to computer-related human activity can make energy saving more productive. In our case study, assuming that the workload of all the computer users are the same, applying these power management rules to their computer can reduce the electricity consumption by 86.6%.

Fine-grained definition of computer-related user activity ensures the accuracy of power management responsive action. According to the optimized context model, the situations of user starting or terminating local access and whether there exists remote access to the computer can be defined by the following expressions.
Chapter 6. Case Studies and Evaluation

6.2.4 Conflict Resolution

The CAPM rule set (i.e. R1~R8) specifies the coordination of multiple services and factors to reduce computer’s electricity consumption while minimizing the interference to users. However, the management of diverse user activities often leads to conflicting requirements, which subsequently materialize into rule conflicts. Conflicting rules act on the same target but the intended actions are different. For the CAPM rule set, if the contexts FinishWork and RemoteAccess occur at the same time, R2 and R5 will establish reconfiguration actions that are opposite to each other, i.e., R2 will shut off computer to save energy, while R5 will turn on the computer to allow remote access. Similarly, if StartWork and EndLocalAccess, NoRemoteAccess occur at the same time, R1 and R8 will issue opposite power management actions.
There are two ways to deal with rule conflict. Firstly, since the user is a principle component in context-aware system who can judge whether the system behavior is satisfactory, the system should allow users to identify any conflict or undesired behavior and configure system behavior according to their current needs. For example, as R2 and R5 were derived from different user needs, when they are about to simultaneously trigger opposite actions, the conflict can be flagged to allow the user to decide whether to allow or reject the execution of R2 or R5.

The other approach for conflict resolution is priority-based automatic judgement. For example, the tradeoff between the goal of energy saving and its impact on user experience needs to be considered to deal with the conflicts of R1 and R8. During working hours, if the user is not accessing the computer and there is no remote connection, R1 and R8 will be triggered simultaneously leading to opposite power management actions. Compared with R8, applying R1 to power management ensures the computer’s fast response to user request at the cost of energy consumption. When there exists the predetermined priorities of energy saving and user experience, the system can resolve such rule conflict and automatically adapt to the power management goal.

Overall, the CAPM rule set demonstrate a flexible and extensible mechanism for context specification. Based on the rule set, the system can provide different configurations according to different situations of the user. Through adaptive rule selection and conflict resolution, the system ensures the satisfaction of various user needs while reducing the unnecessary energy consumption.

6.2.5 Runtime Context Reconfiguration

The CAPM policy rules provide the specification of the context domain as well as the adaptation strategies of system behavior. However, context-aware systems operate in highly dynamic environments: the availability of contexts are unpredictable at runtime, and the user preferences vary in different situations. For example, the policy rule \( R6 \) and
its corresponding context definitions \((c_6, c_7)\), as described in § 6.2.3, specify that the status change of a remote access to “disconnect” should trigger the running computer to execute the power-off action for energy saving. However, at runtime, it could happen that a) the status of remote access is “unknown” to the system, or b) the users who do not access their computers remotely find this rule irrelevant and interfering. The runtime reconfiguration of contexts is therefore necessary for handling the failure of context detection and the change of user preferences. That is, the system should either ignore an irrelevant environmental factor or replace the “unknown” context with some relevant facts to adjust its context-aware behavior.

As mentioned in § 5.1.2, the CCM framework addresses the diversity of human cognition and facilitates the design-phase context space identification. At run time, the end-users could also tune the weights of context views to make the system’s context domain adapt to various situations and preferences. Based on these features, the runtime context reconfiguration can be supported as follows:

- **context detection failure.** Based on the optimized context view and the context-aware action rule set, the adaptive engine searches the context repository for an available context that has a similar effect, i.e., it will trigger the same system actions and produce similar outcome. For example, based on the optimized context view (Figure 6.11) and the rule set specified in § 6.2.3, if the status of remote access is “unknown” at run time, the system can choose the user’s work schedule as context and apply rule \( R2f \) for energy saving.

- **user preference changes.** Based on the repository of context views, the user adjust the view weights to match his own situation and preference. By re-running the AHP calculation, an optimized context view is then generated accordingly. The new optimized scores of the contexts can be used for guiding context-aware responsive actions. For example, Figure 6.11 has shown the optimized context view that takes all context views into account. However, at run time, the users who do not work
remotely can give higher weights to the views of similar situation, i.e., remote access is not considered as a contextual factor. And if remote access is excluded from the context domain, rule $R6$ will not be applied.

Keeping track of the activities of context reconfiguration may help incremental design of context-aware system. As has been illustrated in Figure 4.3, the task of context elicitation and modeling involves an iterative process of data collection, context modeling, context specification and deployment. Runtime reconfiguration provides the feedback from both the environment and the users for continuously developing and evolving the context-aware system.

6.3 Summary and Analysis

This chapter illustrates the application of the CCM framework to context space exploration and context specification within the scenarios of smart meeting room and context-aware power management. Running through the systematic process from data collection, model building, model analysis, to context-aware strategies, the case studies demonstrate the capability of CCM framework for abstracting heterogeneous information with well-designed structure and assisting subsequent context-aware system design.

As has been discussed in § 2.6, although many modeling techniques have been developed to support various aspects of context-aware system design, so far there has been a lack of evaluation criteria for context modeling. Traditional modeling techniques tend to focus on quality of data, support of reasoning and ease of implementation. Considering the characteristics of context modeling, a context model should also emphasize the ability to do context exploration and alternative identification. In evaluating the case studies, we developed the following set of metrics:

**Coverage** This criterion measures the ability of eliciting all types of contextual factors in the environment. A context model should be able to cover basically all relevant
concepts, properties, entities and their interactions. Similar measurement exists in the area of data modeling. Krummenacher et al. use the notion of “applicability” to denote an ontology model’s capability for dealing with heterogeneous data, e.g., time, location, device, etc. [57]. Since context-aware computing usually provides applications that better serve the humans [115], a context model should also be able to build from users’ perspective a compelling story of the world in which an activity situates.

**Visualization** This criterion accounts the use of structures and/or labels to provide system designers with good insights and good design choices. McGee’s criteria for data models [74] stress this criterion with three detailed measures: 1) simplicity, i.e., the model should have the smallest possible number of structure types, composition rules, and attributes; 2) elegance, i.e., the model should be as simple as possible for a given direct modeling capability; 3) picturability, i.e., model structures should be displayable in pictorial form.

**Scalability** It refers to the ability of context model to handle complex and massive context information. Similar to the scalability measurement for ontology models [57], the scalability of context model can be divided into three types: i) cognitive scalability, which refers to the possibility of users to oversee and understand the model; ii) engineering scalability, which refers to the available tool support for large-scale context models; iii) reasoning scalability, which refers to the difficulty of reasoning on large data set.

**Traceability** It refers to ability of context model to abstract individual knowledge and support subsequent context-aware system design. It should assist system designer to decide what situational factors should be involved in the system and how the system will be used. It should also help avoiding the risk of building a sophisticated system that fails to address user goals and cognition. In order to provide ade-
quate control and interpretation, context models should embody diverse individual concepts using fine-grained elements, and provide means to process fine-grained elements and define generalizable terms of context space.

**Flexibility** Flexibility refers to the possibility of extension or adjustment in a context model. According to the studies of context (see § 2.1), the domain of context can be adjusted to include more or less items. An activity’s context subjects to continuous evolution, the context model should allow flexible and low-cost adjustments according to requirement changes.

These five criteria extend existing metrics of data management and knowledge representation, emphasizing the features of context elicitation and manipulation in context modeling. Other general criteria, e.g., formality, reusability and standardization, were also used for the evaluation of context models. It is highly desirable that the model provides a shared understanding of the situation, i.e., the same interpretation of the data exchanged and the meaning “behind” it [102]. It is required that models can be reused by similar tasks and shared among several independent applications.

The case studies have shown that CCM-based context modeling identifies and represents diverse cognitive contexts of a given task. By modeling and integrating context views from different task participants, CCM improves the coverage of context space. The predefined six-dimensional structure of context space facilitates the information extraction process and allows various statistical methods being applied to explore cognitive and behavioral variance. The statistical analysis and optimization on context views provides system designer with good insights and choices of context domain.

The case study on computer power management demonstrates that CCM structures the context domain and provides the ability to extract and summarize massive environmental data. The line plots of context views on time factors captures the intuitive knowledge of contexts and their effects, and CCM guides the application of data anal-
ysis algorithms, such as k-means clustering, for dealing with the large scale contextual information. The AHP-based optimization allows users and system designers to oversee and understand the model.

The CCM framework supports the whole process of context elicitation, representation, optimization, as well as the planning of context-aware actions. Based on an explicit task specification and an optimized context view, the designers can select an appropriate context space and design a context-aware engine that performs seemingly responsive action. The tracing from detailed individual views to abstracted system design can help to reduce cognitive bias in decision making and answer the how and why questions about context selection and context-aware activity.

Compared to traditional context models, CCM is the only context model that takes multiple views or preferences into account. It allows the incremental evolution of context space through the aggregation of independent context views. The specification of context-aware action is also flexible for reconfiguration and evolution. Since the definition of contexts is separated from the specification of policy rules, it is possible to redefine or override existing context variables at low cost. Therefore, it allows the evolution of context definitions with the development of technology and cognition.

A context-aware system developed with the support of the CCM framework may provide benefits to users on several aspects. First, the user’s learning effort should be reduced, since the system’s context domain and context-aware actions are derived from the analysis of the viewpoints and behaviors from potential users. Second, the user’s trust in the system should increase, since the relationship between stakeholder requirements and system design is transparent to users. Third, it allows users to tune the view weights to make the scope of contexts and the priority of policy rules match specific preference or situation. Finally, users new requirements can be satisfied in a timely way after system deployment, since new context views and new context definitions can be captured and added to the context model easily.
Chapter 7

Tool Support

Chapters 4 and Chapter 5 described the CCM-based framework for eliciting context of a task based on the collection of independent context views and presented data analysis techniques for analyzing the variability in human cognition and dealing with cognitive biases in context space identification.

This chapter describes a CCM toolkit for model building, data analysis and context space identification. § 7.1 gives an overview of the system architecture. § 7.2 describes in detail the user interfaces and functionalities of this toolkit. § 7.3 presents the data model for storing and sharing context information. § 7.4 includes the use cases to illustrate how one might use the tool in support of the context modeling process.

7.1 Architecture

Chapter 4 described the framework of cognitive context modeling for eliciting and representing the cognitive context of a task. Some part of the modeling process can be supported by computer-assisted tools. Figure 7.1 provides an overview of the CCM toolkit we developed for CCM modeling. The architecture of the system contains three layers: user interface, the information processing kernel, and the storage of CCM data.

The CCM toolkit provides a graphic user interface to support users in constructing
and storing task-oriented context models. Users can perform the action of model editing, views analysis and context space optimization on a task-oriented context model. Features of the CCM toolkit include:

- A set of user interface elements that enable users to independently model context knowledge and enter data in user-friendly forms.

- A plug-in architecture that manages model elements (e.g., ObjCt, context view, and task) and can be extended with support tools (e.g., for data management, model visualization, inference and reasoning, etc.).

- A XML-formatted storage of context models that allows other applications to access, use, and display models created with the toolkit.

The CCM toolkit uses the CCM parser to process a XML-formatted file into CCM
objects (e.g., ObjCt, context view, task). There are six types of basic objects, e.g.,
time, location, person, etc., as described in Table 4.1. The ObjCt object contains six
dimensions of basic objects and a special object - relation - that links two basic objects.
The task object contains the description of a task. The View object contains a set of
basic objects and relation objects, along with a description of the person who input the
view. This structure corresponds to the definition of CCM in § 4.2.

The kernel of CCM toolkit then manages CCM objects with ObjCt Manager, view
manager and task manager. These object managers handle data inquiry actions and
support data synthesis by dynamically maintaining a body of context knowledge. The
interaction modules, e.g., visualization, AHP calculation and view clustering, respond to
various user manipulations through the GUI widgets. The system keeps track of user
actions so that modifications to a context model can be traced and restored if necessary.

The following sections describe the user interface and the data storage module of the
CCM toolkit.

7.2 User Interface

![Figure 7.2: A screen snapshot from CCM toolkit, showing the interface of adding
contextual elements to a context view.](image)
Chapter 7. Tool Support

The CCM toolkit provides users with three main interaction areas, as shown in Figure 7.2. The menu bar aggregates the plug-in tools, e.g., model editor, analyzer, AHP calculation, etc. Using a menu item, a user can initiate a new action or manipulate existing actions. Currently the toolkit has three tools: model visualization, views clustering and AHP-based context space optimization. As shown in Figure 7.3 (1), a user can use the visualization tool to display the context views in different ways. Figure 7.3 (2) shows the functionalities provided by the AHP tool for optimizing context views. The context model space is used to visualize information retrieved during model manipulation.

Figure 7.4: A screen snapshot from CCM toolkit, showing the interface of adding contextual elements to a context view.

The information space (the right side of the context model space) allows a user to
input information or submit queries using a form-based GUI. The content of the information space changes with different types of actions user choose to perform. Figure 7.2 shows the screen snapshot of the GUI that allow a user to input new contextual factors to the model, and Figure 7.4 shows the screen snapshot of creating a new relation between contextual factors in the model.

7.3 Data Storage

Figure 7.5: A screen snapshot of XML-formatted data storage

The CCM toolkit maintains a database containing user-derived knowledge. Figure 7.5 shows an example of the XML file containing data of a context model. It contains different types of context information and uses XML tags to specify the types of elements in the
CCM structure, e.g., “View”, “ObjCt”, “ObjRel”, etc. Each element is associated with a number of attributes for describing the element and distinguishing it from other elements of the same type. The root of a XML file is the “CCM” element with the attribute of a task description. The CCM element is composed by child elements “CCMViews” and “CCMObjCts”. The data model is shown in Figure 7.6

![Figure 7.6: The data model of CCM toolkit](image)

There are several advantages to keeping things in this format. First, it is easy to share data with all other XML enabled platforms. Second, storing information in a text file make it easy for data analysis techniques being applied directly on the model for mining interesting facts, as a supplyment to the model analysis support provided by the toolkit. Last, the data file is totally extensible. It is possible to encode data that already exists in legacy database and the data contained in the file can also be import to any legacy database, so that different systems or tools can work with each other.

### 7.4 Use Cases

The CCM toolkit provides the data scheme for saving and retrieving the information contained in a CCM context model. Based on the XML-formatted storage model, the toolkit supports various aspects of cognitive context modeling. As shown in the use case
Figure 7.7: The use case diagram of CCM toolkit

diagram (Figure 7.7), system designers, analysts and stakeholders may use this toolkit for data collection, model building, context analysis, and context space optimization.

The toolkit does not support task modeling. Generally, when the task specification is finalized, a system designer creates an empty CCM file which contains only the task description. Then, based on the expertise in the problem domain, an analyst chooses a number of stakeholders for cognitive context data collection. Each stakeholder creates an independent context view of the task. While editing a context view, the stakeholder not only inputs context elements but also identifies the connections between different types of elements. The analyst may also ask the stakeholder to provide the rankings of contextual elements in each category. When all context views are collected, the analyst may check the context views and use his expertise to provide the weight of each view. Then, the analyst applies AHP calculation to generate an optimized context view of the given task.
Based on the task specification and the optimized context view, a set of policy rules can be derived for supporting context-awareness in the system. The policy rule generation is not automated in current release.

As illustrated in the use case diagram, users of the toolkit (participants of CCM-based context modeling) include stakeholder, system designer, and context analyst. The system designer may act as an instance of the stakeholder and provide a context view from the perspective of system design. The context analyst should have expertise in both context modeling and the problem domain. In some cases, the system designer may act as context analyst. However, in order to ensure the independency of context space decision, we suggest that the analyst should not be a stakeholder of the system.

The AHP optimization algorithm (§ 5.1.2) requires that each context view in the CCM model has a weight value, and in each context view, every context element is associated with a score of relevance. As shown in Figure 7.7, the ranking of context elements is provided by the stakeholder who creates the view; while the weighting of context views is provided by an analyst at design phase. In addition to the direct input from user interface, the use case of context ranking can also include measuring the statistical coincidence, when a monitoring environment exists. For example, in the power management case study (§ 6.2), the data of computer usage was used for time factors ranking. The measurement of statistical coincidence can be viewed as a complement to the subjective evaluation on context elements. When dealing with a large number of context views, the analyst may also use clustering algorithms to explore view patterns and assist view weighting. The power management case study (§ 6.2) applied K-means clustering to a set of context views on time factors and divided the context views into three categories. Such applications of statistical analysis are external to CCM toolkit. The analysts may use these external tools to assist model building and model analysis. For example, as described in § 6.2.2, the clustering results helped the analyst to find representative usage patterns, and based on this finding, the analyst interviewed users of each category to explore a variety of
human factors. Clustering context views also facilitates the analyst to decide each view’s relative weight, i.e., the context views of the same category have the same weight value, as described in § 6.2.3.

The AHP optimization produces an optimized context view which can be used as a reference for system designer to decide the context domain of the system and derive context-aware adaptation rules. An optimized context view contains all elicited context elements as well as their connections, the scores of context elements are optimized. The production of a new optimized view also include a log, recording the context view collection (including the ranking of contexts in each view) and the context view weights. When the CCM model evolves, i.e., when views are added or removed, context scores are changed, or context view weights are adjusted, the analyst may re-run AHP calculation and produce an new optimized context view, along with a new log. These logs ensure that the context domain decision and context-aware adaptation rules can be traced back to context view models.

7.5 Summary

This chapter described a CCM toolkit to provide support for building, analyzing and processing CCM-based context model. The toolkit comprises a knowledge base which contains all the context information elicited from individuals, and provides tools for supporting the elicitation of human knowledge and the refinement of context space. The knowledge base is extensible to allow various context views being elicited and incorporated into the context model. The toolkit supports detailed tracing and recording of dependencies throughout the knowledge base and the inference process.
Chapter 8

Conclusions and Future Work

This thesis has presented a systematic context modeling framework which provides guidance for task-oriented context extraction, representation and analysis. The work was inspired partly by the observation that determining the search space or context factors has been taken for granted in most context-aware applications, and they are thus susceptible to incorporate too many or too few facts about the situation. The novelty of this work lies in the support of explicit analysis of cognitive variance in context-aware activities, and in the integration of cognitive contexts and objective contexts into a unified context model which provides a formatted scheme for describing the key situational factors relevant to task execution. With CCM-based context modeling, individuals involved in the task can contribute to context exploration and abstraction, and system designers are provided with the knowledge base for context space decision and context-awareness specification.

This chapter concludes the thesis work and discusses the areas of future research. Section 8.1 presents a summary of the thesis work. Section 8.2 provides a critical review of the framework, while Section 8.3 describes some future work.
8.1 Summary

This section summaries the thesis briefly. We identified a number of difficulties in eliciting
and modeling user contexts, which are important to the design of user-friendly context-
aware system (§ 8.1.1). In addressing these difficulties, we argued that cognitive context
modeling is needed and we set out some objectives based on theoretical analysis and
goal-based evaluation of existing context-aware systems (§ 8.1.2). Finally, we proceed to
develop a framework that meets the objectives based on Activity Theory’s analysis on
the hierarchical structure of an activity and Dewey’s definition of context (§ 8.1.3).

8.1.1 Problem Domain

Context specification, which involves discovering contexts involved in a task and relating
the contexts with specific actions, is a key step in context-aware system design. Context
modeling plays an important role in the context specification process, as a communication
medium between end-users and system developers. In order to provide behaviors that
better match user expectation, the system’s model of interaction should be consistent
with user’s mental model of the system.

Developing unified and precise description of contexts is difficult because it involves
the resolution of trade-offs in deciding the context domain of a context-aware system.
Computers are not currently well enabled to take full advantage of the context of human-
computer interaction. If the context domain is over-simplified and over-objectified, the
context-aware system will ignore the diverse and dynamic mental state of users, resulting
in user frustration and disorientation.

The acknowledgement of contextual facts and the measurement of their correlations
are subject to individual experience. There is a danger that the context specification
represents only the system designer’s perspective and is driven by the ease of system
design and implementation. It is clear that although abstraction, automation and gen-
eralization are important for context modeling, subjective variability and evolution also need to be taken into consideration. Conventional context modeling approaches share a shortcoming that they all concentrate on providing a unified infrastructure to facilitate contextual information management, depending mainly on system designers to decide the domain of context, which is typically the location, identity and state of people, groups and computational and physical objects, while lack of analysis on the trade-offs and alternatives in deciding context domain of context-aware applications. None of current context models explicitly address the importance of balancing situational complexity and technical constraints to achieve maximized user satisfaction.

### 8.1.2 Objectives

The context of an activity can be interpreted as a point in a multi-dimensional context space. The identity of a context is fuzzy and the context space can be adjusted to include more or less elements. Although it is unlikely that context modeling ensures complete description of all dimensions and details of context space, better support can be provided to improve the coverage and accuracy.

Through goal-based analysis on a running example, Chapter 3 emphasizes that context modeling should appropriately reflect end-users' mental state and provide scalable methods of context processing and management. It should allow individual task participants to identify their expected or perceived contexts at a higher, less implementation-specific level. It should provide a mechanism to deal with cognitive bias and form a knowledge base for system analysts to observe and handle the trade-offs and conflicts under social and technical constraint. These objectives help to ensure that context model forms a communication medium between end-users and system designers.

Furthermore, a context model should be well-structured, allowing clustering and comparison algorithms being applied to reveal patterns and assumptions hidden in explicit human activities. Finally, the context modeling framework should provide bi-directional
traceability, i.e., system analysts should be able to answer “why” questions regarding their choice of context space decision, and context specifications can be traced back to individual expectations.

### 8.1.3 Solution

The thesis presented a framework of cognitive context modeling (CCM) to meet the objectives described above. Within this framework, the context modeling process is divided into four steps: i) data collection, ii) model building, iii) context space identification, and iv) context specification. Step i & ii focus on the collection and representation of episodic personal knowledge, while step iii & iv are conducted for deciding the choice of context space and specifying system actions that relate to context changes.

The concepts and structures for context elicitation and representation were described in chapter 4. The CCM context model is formatted as a three-layered structure, representing task, objective surroundings (ObjCt) of the task, and individual views regarding to task participant’s context recognition (CogCt). Task can be interpreted as an activity, which, according to activity theory, contains a flow of operations transforming an objective into an outcome. The individual perspectives, which normally arise from previous experience, cognitive ability, and the subjective role in the activity, emphasize the fuzzy and fluid aspect of context.

The algorithms and procedures concerned with model analysis and context identification were described in chapter 5. There is always a conflict between the infinite, subjective detail of human activity and the finite, objective aspects of system design. This problem is tackled by converting the problem of context space identification into an optimization process, which integrates all context views and associates each objective element with a score of relevance. The output is a context space with interconnected temporal, physical and cognitive factors, which can be applied to the specification of context-aware actions using policy specification language. A toolkit, described in chapter 7, was developed to
Several aspects of the CCM framework are build upon the research in ubiquitous computing, cognitive science and requirements engineering on context-aware/adaptive systems. The structure and classification of contexts are based on the context definitions and models in the literature of context-aware computing [24, 98] and cognitive science [22, 84]. The CCM framework focuses on the exploration of why and how factors for context selection and context-aware system action. Such why and how demands have been addressed in the area of requirements engineering for design space exploration [121] and variability identification of dynamic adaptive systems [65, 66, 123, 40]. Recent work by Lim and Dey also shows that users tend to demand the why and how explanations of system behavior [67, 68]. CCM captures multiple context views to address cognitive bias in context selection. There have been discussions and practices of this feature in both the area of decision making [99, 107] and requirements engineering [27, 106, 71].

8.2 Analysis and Evaluation

The previous section summaries the objectives and the CCM framework developed to meet the objectives. This section examines the strengths and weaknesses of the framework as well as the supporting toolkit.

8.2.1 Advantages

The CCM framework provides systematic support for context modeling. It covers a variety of activities from the initial knowledge elicitation, through the construction of context space, and to the specification of context-aware actions. The framework achieves the objectives as summarized in section 8.1.2. It provides guidance for identifying individual context views of the world in which an activity is situated, analysis of cognitive variances among them, and optimization of context space for ubiquitous computing in
terms of cognitive and technical trade-offs. This allows detailed tracing and recording of dependencies throughout the process, and provides transparency on what the system knows, how the system adapts and why the system knows and adapts. The accessibility of the contextual knowledge can improve the user’s understanding and trust in the context-aware system [68, 105].

Developers of context-aware systems tend to emphasize abstraction, automation and generalization of context models, while marginalizing subjective diversity and system evolution. Cognitive theories suggest the importance of examining social and organizational factors in the process of tasks, but do not provide a methodology that one can readily pick off the shelf and apply to a design problem. By integrating cognitive contexts and objective contexts into a unified context model which provides a formatted scheme for context description and analysis, the CCM framework can likely be used to balance generalization and variability in context-aware system design.

The acknowledgement of contextual facts is subject to individual experience. All context models so far do not provide support for developing and maintaining alternative descriptions from various suggestions. The CCM framework modularizes personal knowledge into independent context views. The diverse context views comprise a knowledge base which contains all the gathered information. It supports a variety of operations carried out on the knowledge base to analyze cognitive and behavioural variance in context-aware activities and assist in the refinement and generation of context-aware adaptation rules. Organizing contextual information as a knowledge base also allows flexible and low-cost adjustments in a context model.

In addition to achieving the objectives, the CCM framework also has advantages on various aspects of context modeling, such as the quality of contextual data, the support for automatic reasoning, and the ease of implementation. Two case studies for illustrating how this approach assists developers in these aspects were presented in Chapter 6, along with descriptions of the criteria - coverage, visualization, scalability, traceability and
flexibility. Running through the systematic process from data collection, model building to model analysis and context specification, the case studies demonstrate that the CCM framework abstracts heterogeneous situational factors with well-designed structure and assists subsequent context-aware system design. The CAPM case study also shows that within the CCM framework, context specification is a continuously developing process and the definition of contexts are separated from the specification of policy rules. This allows modifying context definitions while minimizing the changes on action rules. Compared to current context modeling approaches, the CCM framework not only improves the visualization and coverage of context space, but also provides a scalable and flexible mechanism to handle complex, massive and continuously evolving contextual information.

8.2.2 Remaining Problems

Context modeling is complex, involving knowledge from a wide range of areas while constrained by the limits of cognition and computation. There remain a number of problems with the CCM framework, and a number of aspects which are not tackled by the framework.

The CCM framework focuses on early-stage, task-oriented elicitation, and representation of the domain of context. Other context modeling activities, e.g., data management and context reasoning, are ignored on the basis that they are addressed by existing techniques already. For instance, CCM structures the context space into 6+2 dimensions and provides a XML scheme for storing and representing the contexts, but gives no guidance of formulating the knowledge into a data scheme that can be directly accessed and processed in context-aware system.

A problem related to descriptive representation of contexts is that of recognizing conflicts in terminology. At present, the CCM toolkit handles this issue by forming a terminology base for the problem domain when a CCM is created, and depending on
domain experts to analyze the terminology base and resolve conflicting use of terms between CCM views. Although this approach helps reduce the occurrence of terminology conflicts by encouraging users to choose terms from the terminology base when personalizing their context view, it still requires a lot of human effort to inspect personal terms and refine personal context views. To reduce the possibility of terminology conflicts, the supporting environment provides a context storage which users can navigate and select from, while constituting their view model during the model building phase. Assistance might be provided for this problem by comparing the conceptual structures of isomorphic context descriptions.

The CCM framework provides the guidance for producing policy rules of context-aware system behavior. For the purpose of illustration, the thesis uses a simplified version of policy language. The initial efforts focus on the definition of the necessary policies for context specification, rather than the formal definition of a policy language such as XACML [2], KAoS [109], etc. Currently, the CCM toolkit does not support automatic mapping and generating context specification rules. Although the mappings between CCM elements and policy definitions are provided (see Table 5.2), expertise of policy language and the CCM framework are required for deriving policy rules from an optimized context view.

8.3 Future Work

Various aspects of the CCM framework requires further work. Firstly, the remaining problems discussed above need to be addressed. In addition to that, further tool support is needed to assist model analysis and context space identification. Current K-means clustering and AHP-based optimization only provide basic analysis on the variance of cognitive views. It requires considerable input on element values from users and relies on analysts to identify the variations and patterns. And at present, the context-action
rules are manually produced by analysts. Ideally, the CCM toolkit should allow various data mining and optimization techniques to be applied and automate the rule generation process.

The CCM framework structures the context space as a combination of internal cognitive and external objective factors, emphasizing the description of mental viewpoints on the domains that form the contexts of the system. The “cognitive contexts” are dynamic and complex in nature. The current context view model only captures descriptive knowledge, i.e., a subset of objective surroundings that are known to the user. The cognitive activities or status of users are described by factor weightings. However, while performing a task, people may conceive alternative plans and make decisions based on the imagination or simulation of a series of events. In the literature of cognitive science, several models, e.g., ACT-R, Soar, and MCM (Multi-scale Context Model), have been developed for simulating and understanding the way people perceive and act on the world. In order to develop a schema that completely reflects mental viewpoints and procedural knowledge, the application of such cognitive architectures needs to be explored.

Empirical investigation is needed to test the limits of the framework. Currently the performance of CCM on identifying the quantitative variations is validated for one context dimension in the power management case study. The use of CCM for multi-dimensional context space identification has not been tested. The scalability of this approach for large-scale ubiquitous system (e.g., green transportation and wellness support) also requires further study.

The CCM framework was developed explicitly for the design of context-aware ubiquitous system, but the work in this thesis has potential applications in many areas that require knowledge of context. For example, the CAPM case study has demonstrated that CCM supports the collection, analysis and quantitative measurement of the importance/relevance of environmental factors for a given task, which suggests its potential application in multitask computing to assist resource management and allocation. An-
other example is supporting collaborative activities. The ability of the framework in modeling individual perspectives and in exploring the variance and patterns that exist in diverse cognitive views suggests it has potential applications in improving the awareness in cooperative work, i.e., individuals working together can gain shared knowledge about each other’s situation.
Bibliography


