Sensor Data Fusion for Context-Aware Computing Using Dempster-Shafer Theory

Huadong Wu

CMU-RI-TR-03-52

Submitted in partial fulfillment of the
Requirements for the degree of
Doctor of Philosophy in Robotics

Thesis Committee:
Mel Siegel, Chair
Daniel Siewiorek
Jie Yang
Wolfgang Grimm, Robert Bosch Corporation

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

December 2003

This work is partially supported by Motorola UPR
(University Partnerships in Research) grant

ABSTRACT

Towards having computers understand human users' "context" information, this dissertation proposes a systematic context-sensing implementation methodology that can easily combine sensor outputs with subjective judgments. The feasibility of this idea is demonstrated via a meeting-participant's focus-of-attention analysis case study with several simulated sensors using prerecorded experimental data and artificially generated sensor outputs distributed over a LAN network.

The methodology advocates a top-down approach: (1) For a given application, a context information structure is defined; all lower-level sensor fusion is done locally. (2) Using the context information architecture as a guide, a context sensing system with layered and modularized structure is developed using the Georgia Tech Context Toolkit system, enhanced with sensor fusion modules, as its building-blocks. (3) Higher-level context outputs are combined through "sensor fusion mediator" widgets, and the results populate the context database.

The key contribution of this thesis is introducing the Dempster-Shafer theory of evidence as a *generalizable* sensor fusion solution to overcome the typical context-sensing difficulties, wherein some of the available information items are subjective, sensor observations' probability (objective chance) distribution is not known accurately, and the sensor set is dynamic in content and configuration. In the sensor fusion implementation, this method is further extended in two directions: (1) weight factors are introduced to adjust each sensor's voting influence, thus providing an "objective" sensor performance justification; and (2) when the ground truth becomes available, it is used to dynamically adjust the sensors' voting weights. The effectiveness of the improved Dempster-Shafer method is demonstrated with both the prerecorded experimental data and the simulated data.

Acknowledgements

I am very grateful to my advisor Dr. Mel Siegel for his help and support. It is really one of the luckiest things in my whole life that I can pursue my Ph.D. under the best people I can meet. I really appreciate his kindness towards me as my academic advisor as well as one of my best friends, I can hardly find enough words to express my gratitude to him.

I am also indebted to Dr. Wolfgang Grimm, who gave me lots of support and helped me to go through my hardest time in Carnegie Mellon University. I would also like to thank the other members of my thesis committee, Dr. Jie Yang and Dr. Daniel Siewiorek, for their kind help. It would be very difficult to find another person that is as famous and busy as Dr. Siewiorek, meanwhile is so nice and patient to help me to go through the details with great care. Finally, my thanks are due to Dr. Yangsheng Xu and Mr. Sevim Ablay, who gave me a precious opportunity and assisted me to pursue my goal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1.
Introduction and Motivation

1.1. Sensor, Data, and Information Fusion

Sensing and gathering environmental information is the first step and one of the most fundamental tasks in building intelligent Human-Computer-Interaction (HCI) systems. With the "intelligence" expectation increasing, using multiple sensors is the only way to obtain the required breadth of information, and fusing the outputs from multiple sensors is often the only way to obtain the required depth of information when a single sensing modality is inadequate [157]. However, different sensors may use different physical principles, cover different information space, generate data in different formats at different updating rates, and the sensor-generated information may have different resolution, accuracy, and reliability properties. Thus, how to properly fuse the sensed information pieces from various sources is the key to produce the required information. This is what sensor fusion stands for.

Techniques for multi-sensor data fusion are drawn from a diverse set of more traditional disciplines including digital signal processing, statistical estimation, control theory, artificial intelligence, and classic numerical methods [48]. The characteristics of these commonly used techniques will be examined in Section 2.2.3 in order to find a generalizable sensor fusion solution for a wide range of context-aware computing applications. The following is only a brief discussion about sensor fusion related terminology.

Sensor fusion technology was originally developed in the domain of military applications research and robotics ([20], [53], [54], [60]). Since it is an interdisciplinary technology independently growing out of various applications research, its terminology has not reached a universal agreement yet. Generally speaking, the terms "sensor fusion", "sensor data fusion", "multi-sensor data fusion", "data fusion", and "information fusion" have been used in various publications without much discrimination ([47], [52], [61]). The terminology confusion is well illustrated by a figure in Chapter 2 of [53], which shows a Venn Diagram that purports to represent the relationship among these related terms.

It seems that popular usage has shifted from "sensor fusion" to "data fusion", and it is now moving towards "information fusion[1]" ([51], [59], http://www.inforfusion.org for International Society of Information Fusion). Following robotics convention, however, the term "sensor fusion" is used in this dissertation. In most cases, all the other terms can also be interchangeably applied without causing misunderstanding.

As the lack of unifying terminology across application-specific boundaries had been a barrier historically even within U.S. military applications [48], the U.S. Department of Defense (DoD) Joint Directors of Laboratories (JDL) Data Fusion Working Group was established in 1986 to improve communications among military researchers and system developers. The Group worked out a general data fusion model and a Data Fusion Lexicon ([53] Section 1.6).

The initial JDL Data Fusion Lexicon defined data fusion as ([53] Chapter 2): "A process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and

[1] "Arguments about whether data fusion or some other label best describes this very broad concept are pointless. Some people have adopted terms such as information integration in an attempt to generalize earlier, narrower definition of data fusion …" ([53] Section 2.2).

assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results."

Despite the fact that the concept and implication of this definition can be generalized to encompass a very broad range of application situations, it is also obvious that it is greatly influenced by the patterns of thinking in the military application domain. Revisions of the definition from U.S. DoD and many others choose slightly different words, but basically they all refer to the same theme ([23], [52]).

Some other definitions, however, are more inclusive. For example, in Mahler's definition: "data fusion or information fusion is the problem of refining and pooling multisenor, multitarget data so as to: 1) obtain improved estimates of target numbers, identities, and locations, 2) intelligently allocate sensors, 3) infer tactical situations, and 4) determine proper responses." [56]

Trying to include more generalized situations, Steinberg et al. [61] suggest a formal definition as, "data fusion is the process of combining data or information to estimate or predict entity states."

The work involved in developing this dissertation is in favor of this more inclusive definition class. A formal sensor fusion definition would be: the information processing that manages sensors and collects sensory or other relevant data from multiple sources or from a single source over a period of time and produces (and manages its distribution of) knowledge that is otherwise not obtainable, or that is more accurate or more reliable than the individual origins.

The general data fusion model proposed by JDL Data Fusion Group initially included four differentiating process levels. They are: [Level 1] Object Assessment: estimation and prediction of entity states; [Level 2] Situation Assessment: estimation and prediction of relations among entities; [Level 3] Impact Assessment (Threats): estimation and prediction of effects on situations of planned or estimated actions by the participants; and [Level 4] Process Refinement: adaptive data acquisition and processing to support mission objectives ([52], [61], [62]).

In 1999, the JDL revised the model to incorporate an additional level, [Level 0] Sub-Object Data Assessment: estimation and prediction of signal- or object-observable states, in order to describe the preprocessing at signal level in further detail.

This data fusion model has gained great popularity. However, there also exists the same concern that this model definition is heavily influenced by military operations thinking patterns. As previous described, Steinberg et al. tried to broaden its implication using more general terms [61]. Meanwhile Blasch et al. added a 5th level, [Level 5] User Refinement: adaptive determination of who queries information and who has access to information, to include knowledge management. [62].

This dissertation incorporates this 5-level data fusion model. The ultimate goal of sensor data fusion is to collect and process lower-level signals to extract higher-level knowledge that reveals the "best truth" — in terms of fulfilling the system's mission or providing functional utilities for the targeted applications.

Put it in a simple way, for a specified application, the purpose of sensor fusion is to sense the environmental information of its users or the users' own activity information. From deploying suitable sensors to detect the interested phenomena or parameters, extracting necessary features, combining these features (information pieces), up to dealing with the information storage and distribution, this research studies try to recognize the similarities among different sensor fusion realizations across different situations at different abstraction levels. It aims to form a generalizable solution for building a system architecture that can support fulfilling the tasks of the most popular situations of context-aware human-computer-interaction applications.

The thesis title "Sensor Fusion for Context-Aware Computing" emphasizes that the research focus is on the sensor fusion methodology and its corresponding architectural support, rather than on the context-aware applications themselves.

1.2. Context-Aware Computing, or Context-Aware Human-Computer-Interaction

In terms of providing services to human users, an ordinary service person is much smarter than the smartest computer-controlled machines of today, because the former can react appropriately to the circumstances of the people being served, that is, a human secretary or waiter extensively and implicitly uses "situational" or "context" information. The ultimate goal of context-aware computing research is to have computer-controlled systems behave like smart human secretaries, waiters, or other service personnel.

The idea of "context-aware computing" is to have computers understand the real world so that human-computer interactions can happen at a much higher abstraction level [84], hence to make the interactions much more pleasant or transparent to human users.

The following are some imagined application scenarios that can illustrate what context-aware computing implies and how a "context-aware computing" technology enabled system can enhance service quality or improve human users' personal productivity.

- Example 1: Suppose the user of a context-aware computing system is new to a place (a city, a mall, a tradeshow etc.), and would like to have the system collect the *relevant* information and give him/her a tour. A good context-aware computing technology enabled system should somehow be able to know its user's available time and his/her interests and preferences. It would tentatively plan a tour for the user, get his/her feedback, and then guide him/her point-to-point through the visiting. During the tour, the system should be able to sense the user's emotional states, to guide his/her focus of attention, and to respond to the state changes. According to the user's emotional status change, it would consequently adjust the content description details, adaptively include or omit some contents, and control the content delivery pace – in a manner that a smart human tour guide does naturally.

- Example 2: Today's cellular phone with pager function now has the functions of connecting phone calls and delivering emails. Context-aware computing research is trying to make it smarter ([50], [85]). A context-aware computing enabled personal information management system should further know what its user is doing and adjust its own behavior accordingly. Some examples of good behaviors are: only the time-sensitive e-mails should be delivered to the cellular phone set; the text-voice function should be automatically triggered to read out the email contents whenever it is appropriate; it should be able to sort out priority of the calls and use appropriate ringing methods, etc.

- Example 3: A context-aware computing enabled home service system should be able to detect the activities of its occupants: the room temperature should be adjusted not only based on the time of a day but also based on the occupants' current activities and preferences. Some other potentially additional functions are: it can tell whether young children or senior adults are doing well, it can notice and remind the occupants regarding important evens, and it may recognize and remember where things have been misplaced, etc.

The term "context-aware" was first introduced by Schilit et al ([2], [3]) in 1994 to address the ubiquitous computing mode where a mobile user's applications can discover and react to changes in the environment they are situated. The three important aspects of context that he identified are "where you are", "who you are with", and "what resources are nearby".

While the basic idea of context-aware computing may be easily understood using some examples like the previously described ones, Dey and Abowd [1] tried to formally define it as: "a system is context-aware if it uses[2] context to provide relevant information and/or services to the user, where relevancy depends on the user's task." They further suggested formally classifying context-aware computing into three categories: (1)

[2] We might suggest "… if it senses context …" because sensing context is more to the point the characterizes such situation.

presentation of information and services to a user; (2) automatic execution of services for a user; and (3) tagging of context to information for later retrieval.

Of Dey and Abowd's three categories of context-aware computing applications, it is obvious that they all deal with serving human users. The third category is slightly different from the first two in that the context information is used as a means to aid human memory [10]. Nevertheless, they are all for human-computer interactions. Schmidt even described using context as one of the most basic benefits that enables the change from explicit human-computer interaction mode to implicit human-computer interaction mode [66]. Hence, perhaps the better term to describe this concept would be ***context-aware human-computer interaction***, or ***context-aware HCI***. However, since the terminology "context-aware computing" is well established, it will be used throughout this dissertation.

Roughly speaking, the three terms "context-aware computing", "ubiquitous computing", and "pervasive computing" have been interchangeably used in the computer science research domain without much discrimination. The term "pervasive computing" is slightly more popular among the many researchers who often cite Mark Weiser's paper "The Computer of 21 Century" [81] as the seminal document ([43], [76], [77], [78]) that triggered the current global interest in developing the concept. Many other researchers, however, regard context-aware as a subset of pervasive computing, or, only the necessary means to realize the pervasive computing concept ([70], [73]).

This dissertation builds on the Context Toolkit system ([22], described in Section 1.3.3), so it follows the terminology used there. Thus the term "context-aware computing" is used interchangeably with "pervasive computing" and "ubiquitous computing" concept.

1.3. Supporting Context-aware Computing

1.3.1 Sensing context information

For a context-aware computing system to work, obviously it must be able to sense and manage context information. However, the term “context” can imply an extremely broad range of concepts. Almost any available information at the time of interaction can be regarded as a piece of context potentially relevant to the human-computer interaction.

To give a formal definition, Dey et al. [1] did a survey of existing research work regarded as context-aware computing and suggested: “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 application themselves.”

The following list is just an example of some commonly considered information contents based on Korkea-aho’s [5] enumeration of context:

- Identity of the user
- Spatial information: locations, orientation, speed, acceleration, object relationship in physical space, etc.
- Temporal information: time of the day, date, season of the year, etc.
- Environmental information: temperature, humidity, air quality, light, noise pattern and level, etc.
- Social situation: whom the user is with, the nearby people, family relationships, people that are accessible, etc.
- Nearby Resources: accessible devices, hosts, other facilities, etc.
- Resource usability: battery capacity, display resolution, network connectivity, communication bandwidth and cost, etc.
- Physiological measurements: blood pressure, heart rate, respiration rate, muscle activities, tone of voice, etc.

- User's physical activity: talking, reading, walking, running, driving a car, etc.
- User's emotional status: preferences, mood, focus of attention, etc.
- Schedules and agendas, conventional rules, policies, etc.

Since context is such a broad concept and the ultimate goal of context-aware computing is to provide a ubiquitous computation model to *ordinary human users' daily usage* [42], sensing context is very different from traditional sensing and sensor fusion tasks in the following aspects:

- The goal is to provide computational services to human users anywhere at anytime. Thus context-sensing needs to be implemented in mobile environments using whatever sensors that are available, i.e., the sensor set is highly dynamic;

- Context-aware computing is for human computer interactions, therefore the context-sensing capabilities need to be commensurate with human perception capabilities;

- The context information is for context-aware computing applications — running programs in the computers — as well as for human users' reference directly. The humans often prefer a descriptive semantic format over the numerical parameter of most sensors' outputs; and

- For the system to be used for ordinary users' daily life, the sensors being used cannot be very expensive.

In addition, compared with traditional desktop computing applications, context-aware computing is a new computation mode that is much more complicated ([70], [76], [77], [78]). For such a system to function correctly, it needs system architectural support, which is much different from that of the traditional computer systems ([66], [79]).

1.3.2 Context-aware computing research

The current context-aware computing research and development is still at its infant stage [76]: typically, in most published research projects, only one or very few pieces of context information are sensed and used.

The most successfully used contextual information pieces thus far are human user's identity and location information [148]. Among those early successful location-aware computing research projects, some commonly referenced are the Active Badge (1992-1998) of Olivetti Research Ltd. (now AT&T Labs, Cambridge, U.K.) ([6], [88]) the Xerox's ParcTAB (1992-1993) [7], the Georgia Institute of Technology's CyberGuide (1995-1996) [8], and the University of Kent at Canterbury's Fieldwork or Stick-e (1996-1998) ([9], [10]).

In early — and in many recent — "Active Map" or "Tour Guide" application, the user's current location is the only context information being used: the vicinity map is updated or the nearby point-of-interests are displayed blindly — meaning that the system does not know its user's actual focus of attention, preference, intention or interest at that time ([90], [91], [92], [94]).

More advanced context-aware computing research integrates more context information. Examples include Microsoft's EasyLiving, Georgia Institute of Technology's Aware Home, and Carnegie Mellon's Aura project.

The Microsoft EasyLiving project (http://research.microsoft.com/easyliving/) aims at developing prototype architecture and the necessary technologies for intelligent office environments, where a group of dynamically collected smart devices can automatically collaborate to provide human users a convenient interface to personalized information and services. By the end of year 2001, the EasyLiving system could handle a single room with about a dozen dynamically available devices, and one to three users can using the facility simultaneously ([11], [12], [80]).

The Aware Home Research Initiative (http://www.cc.gatech.edu/fce/ahri/) in the Georgia Institute of Technology creates a living laboratory for research in ubiquitous computing for daily activities. The application target is to provide services to home life of a typical small family or a couple. Many sub research projects have been conducted since then (http://www.cc.gatech.edu/fce/ahri/projects/index.html). Generally speaking, these ongoing projects and experiments are still case-studies with carefully controlled environmental conditions ([13], [27], [86]). The initial goal of having hundreds of sensors ubiquitous deployed has apparently been cut back to a few dozens at most, indicating the practical difficulty of implementing this sort of advanced applications.

Carnegie Mellon's Aura Consortium consists of a series of ubiquitous computing research projects in Human-Computer-Interaction, wearable computers, intelligent networking, and software composition etc. (http://www.cs.cmu.edu/~aura/). Emphasizing minimizing distractions to users' attention, the research goal is to provide each user with an invisible "halo" of computing and information services that persist regardless of location so that the users can interact with their computing environments in terms of high-level tasks (http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/aura/services/www/). The Aura deployment has focused on two main areas. One is a set of contextual information services, which provide information about entities of interest such as devices, people, physical spaces, and networks. The other is to develop applications that exploit the Aura infrastructure, such as predicting users' next activities and preparing relevant equipment for the next task ([43], [107]).

Towards integrating more context information pieces and more complex context, one important step is to modularize (and eventually standardize) the system building blocks or components. Many context-ware computing research projects address or include this topic in course of providing system architecture support ([44], [67], [71], [73], [79]). The Context Toolkit system (http://www.cc.gatech.edu/fce/contexttoolkit/) developed in the Georgia Institute of Technology GVU (Graphics, Visualization & Usability) Center is regarded quite successful in supporting modularizing system components. It effectively separates concerns of context from its usage [22]. This dissertation seeks to expand the

Context Toolkit system with context-sensing and context information management modules.

1.3.3 Georgia Tech Context Toolkit

It is now a common practice for applications to use standard I/O device driver interfaces and GUI functions, though it took many years to achieve this standardization. Based on this observation, Dey et al. ([17], [21], [22]) developed the Context Toolkit system to facilitate building context-aware applications with standard components. As illustrated in Figure 1 from [16], the Context Toolkit consists of three basic building blocks: context widgets, context aggregators and context interpreters.

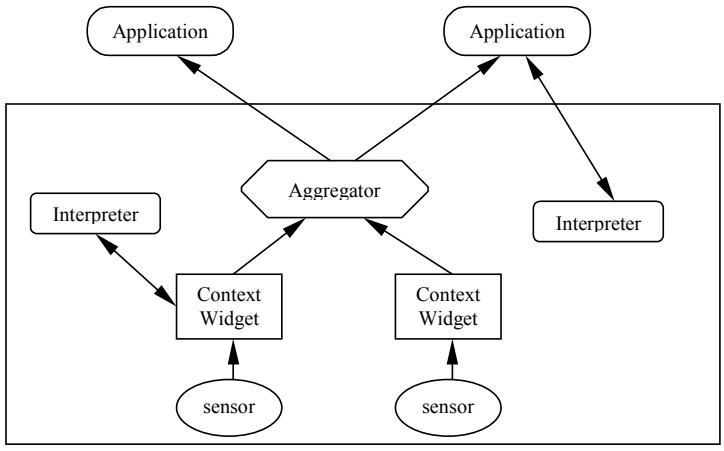

Figure 1. Georgia Tech Context Toolkit component and context architecture

Figure 1 also shows the relationship between sample context components: arrows indicate data flow. The context components are intended to be persistent: instantiated independently of each other in separate threads or on separate computing devices, they are executed all the time.

A Context Widget is a software wrapper or agent of a sensor. It provides a piece of context through a uniform interface to components or applications that use the context. Using Widgets hides the details of the under-lying context-sensing mechanism(s), and allows the system to treat implicit and explicit input in the same manner. Widgets maintain a persistent record of their sensed context, to be polled or subscribed by context-consuming components in the system.

A Context Interpreter is a software agent for abstracting or interpreting context. For example, a Context Interpreter may transfer a location context in form of latitude and longitude to the form of a street name. A more complex interpreter may take context from multiple Widgets in a conference room to infer that a meeting is taking place.

A Context Aggregator is a software agent that collects context from multiple sources, usually for comprehensive information about a particular entity (person, place, or object). Aggregation facilitates the access of distributed context about a single entity.

The Context Toolkit promotes hiding the details of the lower-level sensors' implementation from context extraction, thus allowing the underlying sensors to be replaced or substituted without affecting the rest of the system. However, since the "context" usage is still far away from having any established conventions — in contrast to the highly-evolved computer GUI and keyboard/mouse usage — to actually insulate sensors' implementation from context sensing is now very difficult when many sensors are deployed[3].

[3] Abowd et al say: "After a couple of years' experience using Context Toolkit, we still contend that the basic separation of concerns and programming abstractions that it espouses are appropriate for many situations of context-aware programming, and this is evidenced by a number of internal and external applications developed using it. However, in practice, we did not see the implementation of the Context Toolkit encourage programmers to design context-aware applications that respected the abstractions and separation of concerns." [27]

1.3.4 To fill in the missing part — sensor fusion

Dey, the Context Toolkit author, listed seven benefits (or seven requirements that have to be fulfilled [22]) to use the toolkit. They are: (1) applications can specify what context they require to the system; (2) separation of concerns and context handling; (3) context interpretations convert single or multiple pieces of context into higher-lever information; (4) transparent distributed communications; (5) constant availability of context acquisition; (6) context widgets and aggregators automatically store context they acquired; (7) the Discovers support resource discovery.

From the context-sensing point of view, because sensor fusion support was not among its original design goals, the Context Toolkit system also has the following limitations:

- No intrinsic support for context uncertainty indication: by default, any context information was regarded as correct and unambiguous
- No direct sensor fusion support: an application needs to query or subscribe to all available sensor widgets that can provide the context contents of interest, and it is up to the application itself to decide whether there is any overlap or conflict between any two pieces of the sensed context
- Difficult to scale up: when the sensor pool is large, it is not easy for an application to track all sensors and to make all possible context-providers collaborate.

A context-aware computing application system typically has many sensors in mobile status: old sensors may disappear and new sensors may appear at any time. For sensors to work in such a dynamical configuration, sensor fusion support is necessary. The direct motivation of this dissertation is to provide the Context Toolkit system with the missing part — the sensor fusion support component.

The sensor fusion component obviously needs to provide the sensor fusion functionality. It also needs to perform related administrative functions, such as tracking the currently available sensors and their specifications, collecting relevant data, integrating and maintaining the system information flow, etc. The developed infrastructure has a long-term goal to provide a *generalizable* sensor fusion solution with regard to two goals. First, the system configuration and architecture building blocks can be easily reused for different context-sensing tasks in the same context-aware application system or in different context-aware application systems. Second, the developed sensor fusion algorithm is applicable to as many context-sensing tasks as possible, and its implementation is modularized for reuse.

To achieve this goal, a systematic sensor fusion methodology is proposed and demonstrated. This top-down approach suggests a two-step process: the first step is to define a context information structure for a given context-aware application; using it as a guideline, the second step is to design the information flow inside the sensor fusion architecture.

Dempster-Shafer evidence theory is chosen as the first core module to implement the sensor fusion algorithm. This approach is shown to provide a sensor fusion performance advantage over previous approaches, e.g., Bayesian Inference approach, because it can better imitate human uncertainty-handling and reasoning process.

Compared to the original Context-Toolkit, the two-step with Dempster-Shafer theory implementation approach further separates the concerns regarding the context-sensing process from the usage of the sensed context. This work demonstrates the thesis that synergistic interaction between sensor fusion and context information facilitates the sensor fusion processes, which in turn provide more context information with higher accuracy.

1.4. Outline of the Dissertation

Figure 2 illustrates the key features of this dissertation. Sensor fusion in traditional context-aware computing systems was done in an ad hoc manner, so context-use was highly coupled with the context-sensing sensors; the Context Toolkit system promotes separating them by wrapping the sensors with widgets. This dissertation further standardizes the context-sensing process by specifying a context information architecture and by adding sensor fusion supporting modules. It is intended that this approach will advance context-aware systems toward imitating human sensing and understanding context in ways consistent with human intuition.

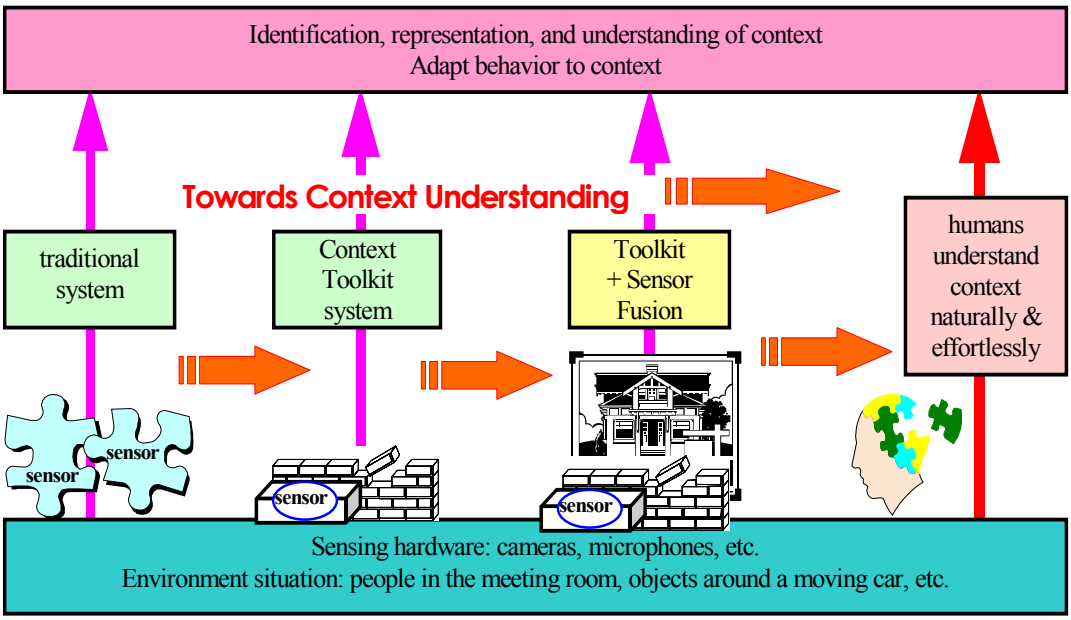

Figure 2. Towards context understanding: this dissertation provides sensor fusion support

This dissertation is organized as follows:

Chapter 1 introduces the background, goals, and terminology of context-aware computing and the new sensor fusion challenges. It outlines the goal of this dissertation: to provide a *generalizable* sensor fusion framework based on the Context Toolkit system.

Chapter 2 first discusses context information classification and representation issues, presenting preliminary research results in context classification; then regarding context sensing, typical sensor fusion methods are analyzed in seeking for a most generalizable method. The context classification, representation, and the sensing technology discussion ultimately leads to proposing a top-down methodology pursued throughout this dissertation.

Chapter 3 addresses realization of the top-down methodology from two perspectives: software architecture development and Dempster-Shafer algorithm implementation. The system architecture discussion analyzes the architectural style characteristics of typical context-aware computing systems, and justifies why the proposed system is an improvement over existing systems. The Dempster-Shafer algorithm research describes the existing typical conflict-handling proposals and introduces a weighting scheme that mitigates conflicts.

Chapter 4 describes the experiments and results of a concept demonstration system. It illustrates how the proposed top-down methodology was used in a meeting-participant's focus-of-attention analysis case study. The outcome demonstrates the concept feasibility, and quantitative sensor fusion results compare the effectiveness of different sensor fusion algorithms.

Chapter 5 further studies the adaptation of the Dempster-Shafer sensor fusion method theoretically and numerically. The discussion of different interpretations of Dempster-Shafer formalism helps to provide a deeper insight into the Dempster-Shafer theory, and based on that, it explains in what situations the Dempster-Shafer method would be more suitable than the most commonly used classical Bayesian inference framework. Using

artificially generated sensory simulation data, the numerical results compare how different sensor fusion algorithms perform.

Chapter 6 summarizes the dissertation results in terms of system architecture improvement and in terms of general sensor fusion advancement in context-aware computing. The dissertation contributions are elucidated in two areas: (1) the system development work adds a sensor fusion module to the Context Toolkit system, and this promotes further separating concerns of context information usage from context-sensing implementation; (2) introducing Dempster-Shafer Evidence Theory into context-aware computing research area solves otherwise very difficult problems, and extending the Dempster-Shafer method practically enhances sensor fusion performance. Finally further research suggestions are given.

The Appendix mainly serves as the technical report documentation for the Motorola University Partnership in Research program, which in part supported this work. It comprises three parts. After an introduction of software architectural support issues for context-aware computing, the first part describes the development process of the system software package. Using the focus-of-attention case study as an example, the second part illustrates how to build a context information database and how to use the software package that was developed. The third part is the Dempster-Shafer sensor fusion module API description manual; this software module is relatively independent, and can be used in other applications with the help of this manual.

Chapter 2.
Context and Context-Sensing

Sensing context for context-aware computing faces a two-fold problem: to represent context properly, and to map sensor outputs into this context representation. To address this two-fold problem, context taxonomy, representation, and its uncertainty management issues are addressed in the first part of this chapter. The second part of the chapter addresses development of a generalizable sensor fusion method.

2.1. Context Contents and Presentation

2.1.1 Context classification

As described in subsection 1.3.1, the trend of context-aware computing development is to integrate more context information. To provide a reference for better managing the context elements, a taxonomy of context information needs to be developed. At the beginning stage such a taxonomy reference can help to identify the most cost-effective context sensing technology for implementation, in the long run it will help to properly choose a course to scale the system up (integrating more context information) towards developing a human-like context-aware system.

There is not much research published on classification of general context information. The reason is perhaps that, at the current development stage, even the-state-of-the-art context-aware computing research does not seriously think of a general context model [109].

The first attempt towards a generalizable context classification is Schmidt et al's scheme [29], which suggests organizing context into two general categories — "human factors" and "physical environment", with three subcategories each. The scheme suggests defining "history" as an additional dimension of context information, orthogonal to the "human factors" and the "physical environment" categories.

The human factors' category is organized into three subcategories: (1) information on the users (e.g., instance knowledge of habits, mental state, or physiological characteristics, etc.); (2) information on their social environment (e.g., proximity of other users, their social relationship, collaborative tasks, etc.); and (3) information on their tasks (e.g., goal-directed activities, higher-level abstraction about general goals of the users, etc.).

The physical environment category also has three subcategories. They are: (1) location information (absolute location, e.g., GPS-coordinates; or relative location, e.g., inside a car, etc.); (2) infrastructure information (e.g., surrounding computing and communication equipment, etc.); and (3) physical conditions information (e.g., level of noise, brightness, vibration, outside temperature, room lighting, etc.).

Not directly addressing context classification, but attacking a related problem from an implementation perspective, Dey et al [1] suggested categorizing general contexts into a two-tier system. The primary tier has four pieces of information ("location", "identity", "time", and "activity") to characterize the situation of a particular entity. The secondary tier is considered as contexts to be indexed by the primary tier contexts.

If a candidate list of all feasible context-sensing technologies and their usage situations is required to find and organize "killer" applications for implementation, or if a reference is required to help find and organize context-sensing technology development directions to pursue, neither Schmidt's nor Dey's classification can help very much. A more detailed classification scheme is needed.

Intuitively, a systematic way to classify a user's context information would be to explore its contents in physical and temporal dimensions, and to attempt classification in terms of intention, mood, etc. The ultimate organizing scheme, well beyond the practical

scope of this dissertation, might be something like the one described by Takeo Kanade in CMU Robotics Seminar of November 1, 2002. Kanade suggested a model to describe human functions and behaviors within the computer's virtual world. His model comprises three classes of functions: physio-anatomical, motion-mechanical, and psycho-cognitive. The physio-anatomical sub-model sees the human body as a living entity that regulates and controls various parts, organs and circulatory systems. It describes the shapes, material properties, physiological parameters, and their relationships to internal and external stimulations. The motion-mechanical sub-model sees the human as a machine that can walk and run, move and manipulate objects. It concerns kinematic, dynamic, and behavioral analysis of human motions. Finally, the psycho-cognitive sub-model deals with human's psychological and cognitive behaviors as they interact with events, other people, and environments.

The true difficulty in context classification is not how to define some *orthogonal* dimensions that can categorize the context contents, but rather its origin is in the nature of the far-reaching implications of context information itself. For example, at different abstraction levels, a user's activity context description can be "fingers tapping the keyboard", "typing on a computer keyboard", "typing a report", "preparing a report for a task", "working", etc. The number of ways to describe an event or object is unlimited, and there is not a standard or even a guideline regarding granularity of context information.

Because classification of general context information is not feasible at the current research stage of context-aware computing, a pragmatic approach is adopted. The adopted context classification scheme is a user-centered approach that groups context into three categories: (1) the physical environment around the user; (2) the user's own activity; and (3) the user's physiological states. The relationship of the three context categories is illustrated in Figure 3, and the context elements are listed in Table 1, Table 2, and Table 3.

As discussed in Section 1.2, context-aware computing is for human-computer interaction purposes, so using the user-centered scheme to classify context is a natural evolution in context information management. Notice that because the state-of-the-art

context-understanding level varies in different aspects of human user context, the boundaries among the three context classification tables may not be crisply clear. However, the three classification tables per se should be valuable for providing a reference to context-aware computing application developers, because they include all the context information elements that have be used or even mentioned in all the literature the author have found.

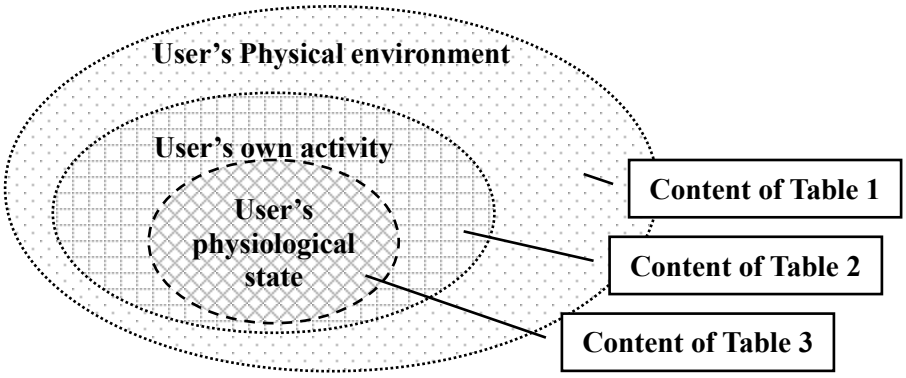

Figure 3. The user-centered scheme to group context information

Referring to Table 1, environmental context description includes the following aspects of information:

- Location related information in terms of large area, absolute physical position, or geographic and climate implication etc.; when it changes, it means the user is traveling.
- Proximity related information in terms of small area, which includes where the user is in at the current time, what the environment means to humans, and what facilities and devices he/she can reach and possibly use; when the proximity is changing, it means that the user is walking or running away from one place to another.

- Time related information which can be either in the absolute sense such as time of the day (it implies darkness of the sky, etc.), season in a year (it implies feelings humans would have in out-door activities, etc.); or in the sense of the expected activities (including other people's activities) such as time for work, for lunch, for a vacation etc.

Table 1. A human user's physical environmental context description

perspectives	contents	transition	extension
location (absolute)	community, street, city, geographic information, (altitude etc); weather (cloudiness, rain, snow, temperature, humidity, barometer pressure, forecast)	Location-change: traveling, speed, heading,	
Proximity (relative)	orientation, building (name, structure, facilities, etc.), room, car, devices (function, states, etc.), vicinity temperature, humidity, vibration, oxygen-richness, smell, etc.	Proximity-change: walk or run, speed, heading	
time	time, day, date, season; occasion (show, meeting, cock tail party, etc.); function-time (work, vocation, etc.)	Function-time change: lunch time, etc.	history, schedule, expectation
people	individual, group; interaction (casual chatting, formal meeting, eye contact, attention arousing); non face-to-face interaction	interruption: incoming calls, encounter, leave, etc.	social relationship
audiovisual	man-made voice (bearing information), music, etc.; noise-level; in-sight objects, surrounding scenery; brightness		
computing connectivity	computing environment (processing power, information availability & cost), network connectivity, communication bandwidth, communication cost		

Table 2. A human user's own activity context description

Mental activity	Work, rest, play, etc; Task-ID, value, objective, means, interruptible? etc.
Physical action (global)	Corporal: drive, walk, sit, …
	Visual: read, watch TV, sight-seeing, …, people interaction - eye contact
	Aural: information content - work, entertainment, living chore, etc., …
Physical action (local details)	Hands & other body-parts: type, write, use mouse, etc., …

Table 3. A human user's state context description

Feeling	Sensation: cold, hot, warm, tight, loose, painful, etc.
	Mood: tranquil, exciting, happy, merry, sad, grief, etc.
	Agitation & tiredness: curious, energetic, numb, sleepy, etc.
	Stress: serene, composing, nervous, anxious, etc.
	Concentration: focus of attention, indulging, interesting, indifferent, etc.
Preference	Habitual bias: love, hate, passion, etc. Current likeness: appreciate, enjoy, disgust, suffer, etc.
Physical characteristics	Name, address, contact information, etc. Height, weight, heart rate, metabolism, health condition, etc.

- People around the user, which mostly decides what interactions the user might engage in; the users' social relationship will also be reflected here.
- Audiovisual information, which describes what the user can hear and see.
- Computing and connectivity information, which describes what computation services (perhaps other context-aware services) the user may use.

Without an appropriate granularity reference for specific application or domain, it is not easy to describe the user's activities. Table 2 shows the structure of a model with three abstraction layers. The highest layer is a description of the user's conceptual activities. It uses abstractive vocabularies (such as "working", "resting", or "playing" etc.) to convey information regarding the activities' value, objective, means etc. The middle layer is a description of the user's physical activities from the global status. It conveys the information regarding the user's physical engagement in such activities — the actions the user takes, and the contents the user handles, etc. The lowest layer is a description of the user's physical activities from local and detailed interaction. It conveys information regarding the user's body posture, head orientation, hands and feet movement, using tools and interaction with objects, etc.

With regarding to the user's personal character and feeling, if the user is cooperative to the system, at the physical embodiment level, his/her personal information such as address, height, weight, health condition etc. can be recorded and used without much difficulty [149]. At the middle level, this user's general preferences or altitude towards certain objects, events, or stimuli can also be recorded and used — although his/her current likeness cannot so be so easily sensed, it is still possible to collect and use such information if the user is willing to cooperate. However, at the most intangible level, it would be even too difficult for an ordinary user to describe his/her feelings or emotional status verbally to another person consistently. Given that this is one of the most important aspects of context information worth further investigation, a few items are tentatively listed as candidates in Table 3 for consideration.

When the information granularity is not specified, there can be many ways to describe the same event or situation. To avoid such possible confusion, a practical solution is to carefully pre-specify an information description vocabulary set for given applications. The resulted context classification tables are thus further elaborated for the given application scenarios. The tables and their content representation are called "information architecture" thereafter.

2.1.2 Context representation

A knowledge-based system (KBS) has two essential parts: the reasoning or problem-solving process (R part) and the knowledge (K part). According to Edward A. Feigenbaum, building the K part is a much more difficult task (Forward of [13]). A successful context-aware system is also a knowledge-based system in that it includes managing and extracting a large scope of information. Building a context information architecture that can be sustainable in the long run is a really big challenge, and it may be a reciprocal process as the result being improved with each trial. It would be beneficial to draw the experiences gained from building knowledge-based systems.

2.1.2.1 Basic requirements for context representation

Analogous to the fact that an object has molecular properties under microscopic scale observation whereas it has geometric shape and texture properties under human perceptible or palpable scale observation, the same context information can have different property descriptions at different detail levels. Also analogous to the fact that an object can be observed from different perspectives, from different distances, and under different lighting conditions, the same context information can be represented in many different ways. Building a *general* context information model to capture all aspects of a user's information is an impossible task.

However, analogous to the fact that perspective machine drawings are valuable in making complex machine, by limiting information granularity and confining its

vocabulary, a complexity-reduced context set would be valuable to facilitate human-computer interactions [111].

Using symbols to describe knowledge or facts (referred as the "context" here) is a registration and recognition process ([19] page 32). The key is to make the information representation consistent over different applications and perhaps endurable for future development, thus, to make it *generalizable*.

For the representation to be a generalizable, it has to be explicit or declarative ([19] page 80). It should have the following three basic properties. First is modularity: the representation needs to be self-contained and autonomous. In other words, there should be an identifiable and bounded set of symbol structures that make up the representation. The symbols ought to be distinct and separate from the interpreter programs that use them, and there should be a set of well-defined and narrowly prescribed interfaces by which other parts of the system can access and manipulate the symbol representations.

Second is semantics: the representation must have well-understood semantics. Properly choosing presentation formats (using descriptive words, figures, etc.) so that their meaning is obvious to users is especially important for direct human-computer interaction applications.

Third is causal connection: there must be a causal connection such that changing the representation causes the system to change its behavior in a way that is appropriate for the change to the representation and its semantics. This causal connection provides the basis by which the representation governs reasoning and behavior of the system.

The combination of modularity, semantics, and causal connection makes it easier to change explicit representations than implicit representations. In reality, whether an explicit context representation is generalizable also depends on how much the context information architecture can correctly anticipate the nature and extent of the different situations in which the system must operate [89].

2.1.2.2 Modeling context

Context-aware computing strives to bridge the gap between the real world and the virtual computational world ([77], [78]). Modeling the connection between real world objects and their counterparts in the digital virtual world will influence system architecture realization.

To model a context-aware application situation is the necessary first step to represent its complex content systematically. To model a situation is to analyze the human users, objects, and events to identify relevant properties and relationships of interest, in order to form a simplified abstract world that corresponds to the real application situation.

Since "context" is the "circumstances in which an event occurs"[4], a sensible way to model a simplified but expandable world of concern is to set up a *stage* on or in which context-aware applications play themselves out. On the other hand, since context-aware computing applications are for human-computer interactions, so, naturally the context model developed in this dissertation contains two basic semantic entities[5]: "STAGE" and "USER", defined as follows.

A STAGE entity is an abstract representation, or a software agent, of a place where context-aware applications take place. It physically corresponds to a central facility or place that has the desired functionalities relevant to the expected context-aware applications. Different applications play out on different stages. Some examples stages are an individual's home office, a small conference room used by a small group of colleagues, a computer cluster room, etc.

[4] American Heritage Dictionary, 3rd Edition, 1992

[5] "Entity", "primary key", "foreign key", "attribute", "group attribute" etc. are conventional modeling and database terminologies. Their meanings are quite self-explanatory in the contexts used here; more and detailed explanation can be found in information modeling and database literatures, e.g., [19], [34].

A USER entity is an abstract representation of, or a software agent on behalf of, a human user. For context-aware system targeted human users, their applications commonly required information is usually pre-registered.

User identification and location information are the most commonly used context in today's context-aware applications. To include more context elements in a generalizable model, two more kinds of entities are defined, the "OBJECT" and "EVENT" entities.

An OBJECT entity is an abstract representation of, or a software agent of, an object in real world. A "real world object" is an object in a broad sense that collectively consists of the settings in the STAGE. It can be a piece of furniture, a device, or a software program, etc.

An EVENT entity is an abstract representation of an interaction among USER, OBJECT, or STAGE entity. Some examples of an EVENT are a USER appears in a STAGE, an incoming phone call, an automatically executed program, etc.

Considering that a USER's one EVENT would be a piece of "context information" to other EVENT's of the same USER or other USER's, this information modeling approach is the right way to work with the user-centered context information classification scheme described in subsection 2.1.1. With this modeling scheme to analyze and simplify context-aware situations, new context information items are expected to be more easily included. Figure 4 illustrates the relationships among these entities.

Figure 5 illustrates an example of a context information model that comprises a STAGE and a USER basic semantic entity. Each of the entities can contain some basic properties and some child entities (OBJECT's and EVENT's), explained as follows. The STAGE entity may contain two kinds of EVENT sub entities: USERS (appearance of user or users) and USERS-ACT (users' group activity description), and two kinds of OBJECT sub entities: EQUIPMENT and SENSOR objects. The USER entity may contain instances of an EVENT sub entity ACTIVITY and two kinds of OBJECT sub entities: PREFERENCE and SCHEDULE.

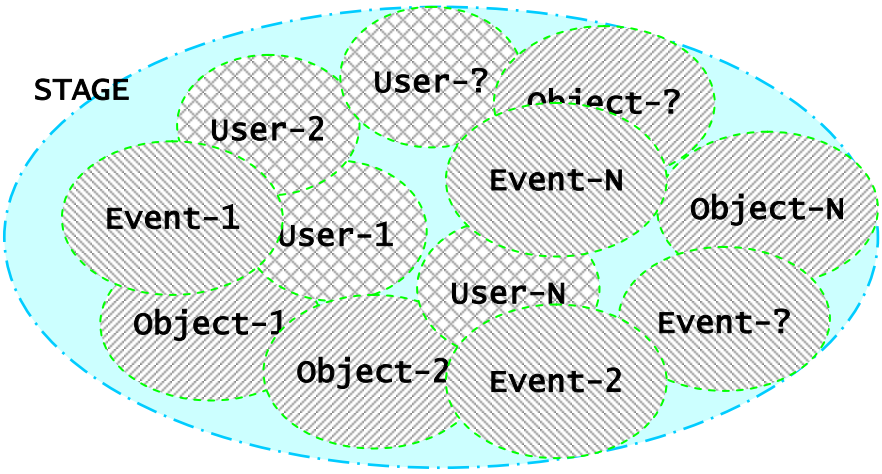

Figure 4. Context model: stage, users, objects, and events

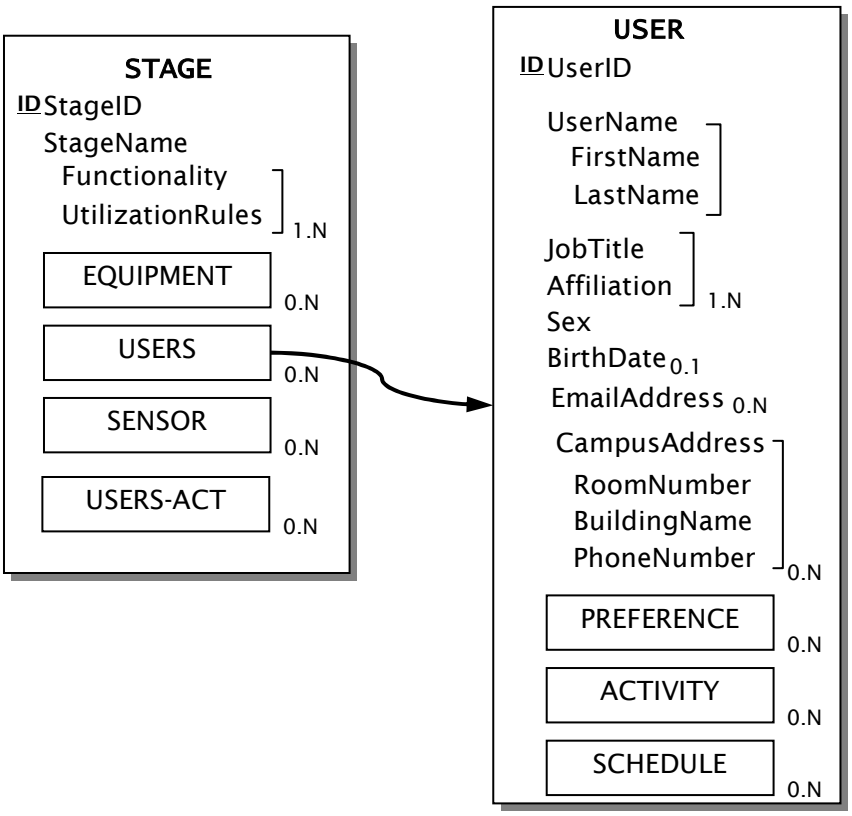

Figure 5. The two basic semantic objects in a context information model

The notation of "m.n" in Figure 5 means that the described item has at least m and up to n ("N" means an unlimited number) instances. The STAGE entity has a unique attribute (the primary key) StageID as its identifier, a name attribute of StageName, and at least one, maybe an unlimited number of, group-attributes of Functionality and UitilizationRules. The STAGE may have many (with its maximum cardinality, N, unlimited), or may not have any (with its minimum cardinality being equal to 0) USERS, EQUIPMENT, SENSOR and UESERS-ACT entities.

The USER entity in Figure 5 has UserID as its identifying attribute (the primary key), a UserName group attribute, a Sex property attribute, and at least one but maybe an unlimited number of JobTitle and Affiliation group attributes, etc. It may contain an unlimited number of PREFERENCE, ACTIVITY and SCHEDULE entities, but it might also contain none.

Generally speaking, the STAGE root entities and their sub entities are abstract representation of items chosen from the context classification Table 1, whereas the USER root entities and their sub entities are abstract representation of items chosen from context classification Table 2 and Table 3. The resulting model is an explicit representation of complex context information, and the representation is generalizable because it can adapt to a broad range of context-aware application scenarios, where new context can be added easily. Context-aware research thus far has demonstrated that such a very simplified and incomplete context information model can still be very useful to many applications ([69], [70], [74], [75], [77]).

2.1.2.3 Context database implementation

To use this context model, implementing it in a central database format is preferred because the information is then clustered together, easily identifiable, and separated from interpreters ([19] page 81-83). Transferring the model into context database format is relatively easy by following the well-established conventions [34].

The database has two root tables, corresponding to the STAGE and USER entities. Roughly speaking, each entity will have a table representation, and the attributes with limited cardinality number will be shown directly in their corresponding entity tables, whereas the unlimited repeatable multiple-value attributes will be represented in separate tables using their corresponding entity foreign key as one of their primary keys.

Figure 6 shows a database that has partially implemented the model described in Figure 5, where the STAGE entity root table has two secondary-indexed tables — the EQUIPMENT table and the USERS table, and the USER entity root table has one secondary-indexed table — the FOCUS-OF-ATTENTION table. The FOCUS-OF-ATTENTION is an instance of ACTIVITY event entity in Figure 5. Here a conventional graphic representation of database components is used, where tables are shown with their column heads and their relationships are illustrated in lines with crossing dash, oval, and/or crow-feet symbols indicating cardinality of 1, 0, and N respectively, detailed explanation can be found in [34].

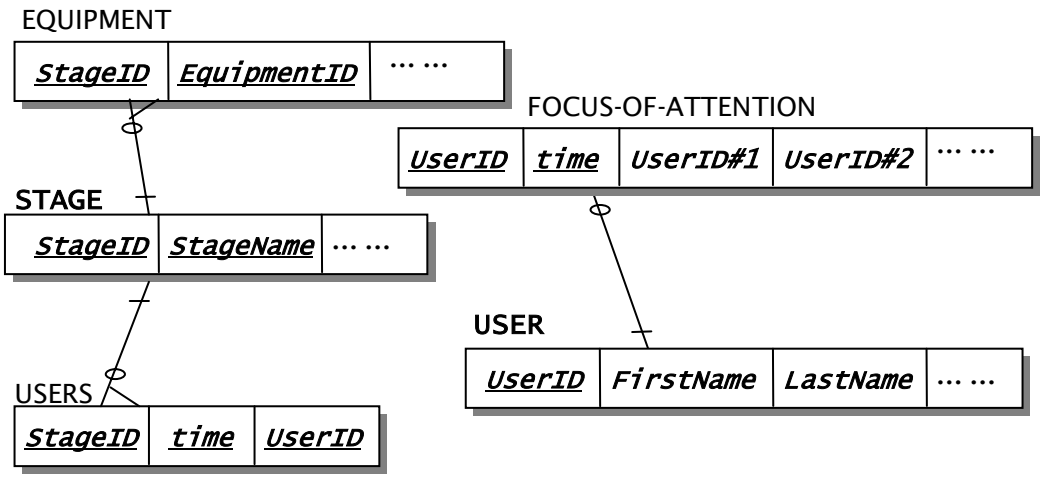

Figure 6. Transfer context information model into relational database

As part of building context information architecture process, context modeling and representation may need reciprocal trials too. In designing the context model, careful consideration should be excised regarding how to incorporate the time information. In Figure 6, the `EQUIPMENT` table records the available equipments in the `STAGE` object, which may not change very often over the time of interest, so it has a joint primary key of `StageID` and `EquipmentID`. However, the `USERS` table describes which `USER` is, or `USER`'s are, on the `STAGE`, which may be quite dynamic, so it has its joint primary key of `StageID` and `time`.

In concluding subsection 2.1.2, this context modeling and representing scheme practically treats the context items as if they are in a single layer. Treating the presumably hierarchical information structure (at different abstract level) as a single layer can mitigate impact due to possible classification scheme change, which is very likely to happen [49] in the research process[6]. The disadvantage of this practice is that it would be difficult for human users to find their interested information pieces directly if the information is not properly rearranged on HCI interfaces.

2.1.3 Managing uncertainty information

The context database fields can be filled either by direct human input or automatically by sensor output. In either cases, uncertainty is an intrinsic property that has to be properly managed by the context information architecture [28].

In traditional parameter measurement, every sensor has its own measurement accuracy and precision limitation specifications, and every measurement can have an error estimation (often denoted as σ). Uncertainty management in traditional parameter measurement is well understood. In simple cases, the usual rules of error propagation and statistical weighting of redundant and complement measurements constitute the simplest

[6] In reality, it not seems possible to come out with a complete list of classified contexts without many reiterations, because of the nature of information contents and because of lacking information granularity standards.

sensor fusion. In more complicated cases, more sophisticated sensor fusion methods are still available to combine multiple measurements to generate higher-quality estimation.

One important error source is the drift that exists in all sensors. Originating from slow change of the sensor's characteristics, often related the sensor's aging or adapting to its environment, it adds a slowly changing offset[7]. Sensor calibration is the only way to overcome drift. A way to implement continuous calibration will be discuss in Section 3.2.

However, the context information of interest is not confined to traditional parameter measurement. In fact, most of the data handled by today's context-aware computing systems are non-parametric. Discrete environmental facts and human-interaction events, in addition to measurement per se, are of major concern in the context-aware computing. To manage uncertainties involved in the discrete facts and events, for each such context information item, all possible values that it can take, as well as their ambiguous combinations, are listed first. Then by assigning a probability value to each set in the list, an efficient uncertainty representation can be achieved. This process is further illustrated in the following example.

Imagine a scenario in which exactly one person is detected in an instrumented room. It has already been concluded that this person can only be User-A, User-B, or User-C. For this specific user identification situation, the complete list of possible values that can occur is enumerated in Table 4.

Table 4. Context uncertainty management user-identification example

	{A}	{B}	{C}	{A, B}	{B, C}	{C, A}	{A, B, C}	{Ø}
Probability	P_A	P_B	P_C	P_{AB}	P_{BC}	P_{CA}	P_{ABC}	$P\emptyset$

[7] For most sensors and instruments provided by reputable manufacturers, their specifications would usually include initial accuracy and precision, and guarantee the drift rate to be within certain range for specified period, e.g., one year.

In Table 4, {A} means that User-A is present; {B, C} means that either User-A or User-B (but it is not clear which one) is present; and so forth. The symbols P_A, P_{BC}, etc. stand for the corresponding probabilities. Notice that the set {A, B, C} means that either User-A, User-B, or User-C is the case. This is actually an acknowledgement of ignorance regarding the user identification situation given the constraint. The symbol {Ø} refers to an exception that the constraint is violated, for example, this is neither User-A, nor User-B, nor User-C, or even there is actually nobody present.

With this scheme of information uncertainty management, all possible situations and human reasoning cases can be easily captured. This representation allows incorporation of artificial intelligence techniques using frame reasoning symbol systems, thus it provides a generalizable solution suitable for future research.

In practice, the major source of uncertainty originates from transferring qualitative context information to the quantitative representation. This is especially true for situations that involve human activities. For example, if an ordinary user is asked to describe his/her preferences regarding some objects or events, most likely the responses would be some qualitative descriptions such as “very much positive”, “positive”, “somewhat positive”, “neutral”, etc., which would then have to be transformed into a numerical quantity resembling a probability description.

With the context information architecture being defined and with each of its possible values being specified, the whole system’s information flow is largely decided. It will become clear in section 3.2 that this uncertainty management scheme naturally leads to a generalizable sensor fusion method using the Dempster-Shafer Theory of Evidence reasoning mechanism.

2.2. Context Sensing

Context sensing is to realize populating context database with context pieces derived from sensors’ lower-level outputs. This section deals with the information mapping issues

in the process, especially emphasizing how to use sensor fusion techniques to properly handle the competing sources.

2.2.1 Mapping sensory data into context information space

Sensors measure physical parameters of their environment, whereas context-aware computing systems usually need to include context information at various levels, from current environment to human users' intention and activities [72]. There is a gap between what sensors can now provide and what is needed [68], and the task of context sensing is to bridge the gap ([14], [26]).

There can many ways to bridge the gap, i.e., to realize the mapping from sensors' lower-level output into the context information architecture described in previous sections. This dissertation follows the so-called "widget" approach to fulfill the required mapping functionality. Analogous to software drivers that interface peripheral devices with computers, widgets [8] are software agents that transfer sensors' lower-level information into predefined higher-level context. Table 5 lists currently available sensor and data processing (performed by widgets) technologies that can sense the commonly used context information [87].

The simplest format of information mapping is one-to-one (one source to one destination) with a monotonic mapping function. For example, the room temperature measured by a thermometer sensor can be mapped into one of the three states as of "cold", "warm", and "hot".

However, most context sensing requires a network of sensors to work in concert. These sensors may collaborate in different modes (to be described in next subsection), and their mapping mechanism can be described as a many-to-many relationship. With vector $[s_1, s_2, \cdots, s_m]^T$ representing the sensor array's output data, and vector

[8] "Widgets" are interchangeably called "context widget" or "sensor widget" in this document. Widget for context collecting implementation in the Georgia Tech's Context Toolkit system is discussed in subsection 1.3.3.

$[x_1, x_2, \cdots, x_n]^T$ representing the desired context description, the mapping function $sf(\cdot)$ relates sensors' output to the context information as shown in EQ. 1.

Table 5. Context sensing achievable with commonly used sensors

sensors	widget	context information
microphone	sound processing	sound pattern recognition speaker recognition, speaking understanding
cameras infra-red sensors	image processing	object recognition 3-D object measuring face recognition
GPS, DGPS serverIP, RFID gyro, accelerometers dead-reckoning , network resource	map registration	location altitude speed orientation
thermometer barometer humidity sensor photo-diode sensors accelerometers gas sensor	direct registration	physical & chemical environment
biometric sensors: heart-rate blood-pressure GRS, temperature respiration	registration & psychophysiological mapping	personal physical state: heart rate respiration rate blood pressure blink rate Galvanic Resistance body temperature sweat

EQ. 1

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \xleftarrow{sf(\cdot)} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}, \text{ or } \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = sf\left(\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}\right)$$

When more than one sensor contributes to a context information item, the mapping operation is actually a sensor fusion function. The simplest sensor fusion operation is a linear combination of inputs to a single output; in this case, the operator can be expressed in a format of matrix multiplication as the following:

EQ. 2

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} sf_{11}(\cdot) & sf_{12}(\cdot) & \cdots & sf_{1m}(\cdot) \\ sf_{21}(\cdot) & sf_{22}(\cdot) & \cdots & sf_{2m}(\cdot) \\ \vdots & \vdots & \ddots & \vdots \\ sf_{n1}(\cdot) & sf_{n2}(\cdot) & \cdots & sf_{nm}(\cdot) \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}$$

Most context sensing applications, however, require mapping and sensor fusion functions that are more complex [110] than merely a linear combination. The next subsection lists, compares, and contrasts the types of sensor fusion algorithms in common use.

2.2.2 Sensor fusion architecture for context sensing

Context sensing technology can advance in two directions: an individual sensor's functionality and performance improvement or, intelligently connecting multiple sensors to provide higher-level information. The later is increasingly needed with today's ever-growing computer networking development ([39], [40], [65], [70]).

Toward intelligently combining sensor outputs, all issues eventually boil down to the fundamental question: given the context information architecture, how should the sensor fusion system manage the mapping mechanism so that various sensors can *properly* contribute to the results in the predefined information space. The adverb "properly" here means to resolve conflicts so that the information accuracy, confidence, or reliability is

improved ([37], [41]). There is no simple solution to fulfill this goal, because different sensors have different resolutions and accuracies as well as different formats and update-rates. In addition, the sensed data may have overlaps or conflicts.

Sensor fusion can be classified from different perspectives, such as information level, implementation architecture, algorithms being used etc. ([36], [59], [60]). Durrant-Whyte et al ([24], [156], [158]) suggested classifying sensor fusion into "competitive", "complementary", and "cooperative" types according to the nature of information input-output relationships. They are explained in the following paragraphs.

Competitive type sensor fusion combines sensor data that represent the same measurement to reduce uncertainty and resolve conflicts. This is the basic sensor fusion type. It is often regarded as the "traditional" or "classical" sensor fusion technique. Some examples of this type are: taking multiple measurements and then applying the weighting average algorithm to accurately evaluate the size of a machine part, manufacturing a certain number of standard parts and measuring the products to evaluate the status of a machine tool, etc.

Complementary type sensor fusion combines incomplete sensor data that do not dependant on each other directly to create a more complete model. For example, combining sensor data according to a predefined physical model could enable the sensor outputs collectively to estimate the state of a higher-level measured physical process. A more specific example is combining measurements of pressure and airflow to estimate the propulsive force of a jet nozzle.

Cooperative type sensor fusion combines sensor observations that depend upon each other to deduce higher-level measurement. In stereovision, for example, image components (pixels, featured spots) depend on each other in pairs to estimate object distances.

Roughly speaking, competitive sensor fusion enhances measurement reliability or confidence, whereas complementary and cooperative sensor fusion lead to higher-level measurements. The three sensor fusion types are not exclusive though, as many sensor

fusion processes can belong to more than one type. Moreover, of the three sensor fusion types, the complementary and the cooperative types are generally domain specific — means that their methods are often valid only under specific conditions where specific knowledge based artificial intelligent inference techniques can apply.

From the information process model point of view, however, sensor fusion can be roughly grouped into three categories [158]: (a) direct data fusion, (b) feature level fusion, and (c) decision level (identity declaration) fusion, as shown in Figure 7 ([53] Chapter 1).

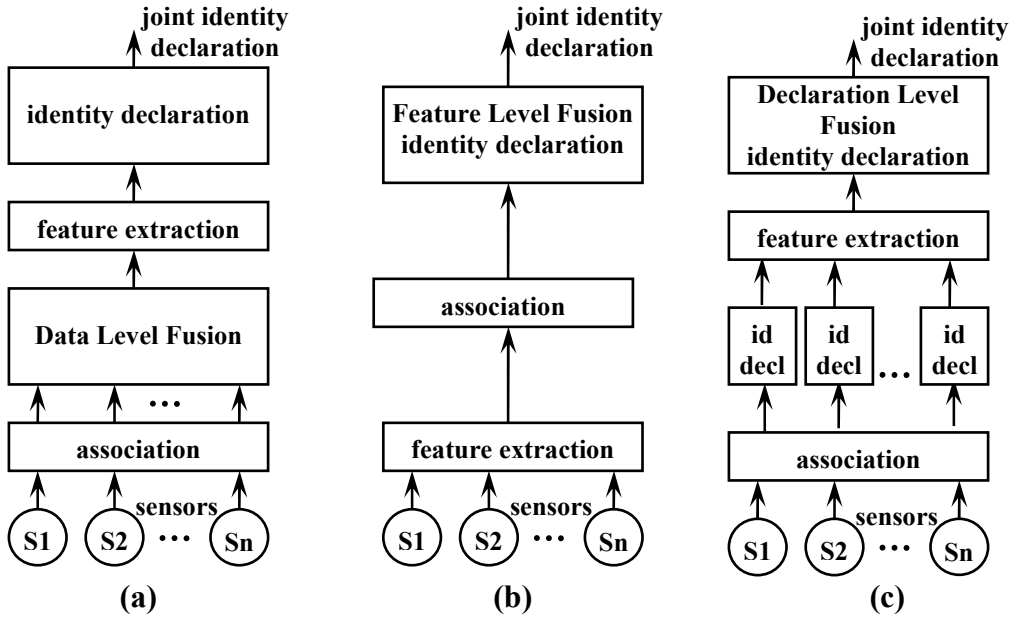

Figure 7. Sensor fusion process model: (a) direct data fusion, (b) feature level fusion, and (c) declaration level fusion [53]

If the sensors are measuring the same physical parameter, the low-level sensory data can be directly combined. Otherwise, the information can be fused only at feature or decision level.

Representative features from sensors provide signature elements for object recognition. In feature fusion, features are extracted from multiple sensor observations and combined.

In decision level fusion, each sensor makes a preliminary determination of targeted objects' identity and other attributes, and the fusion algorithm combines these to generate more accurate results or higher confidence ([82], [83]).

In image processing to recognize objects scenario, for example, sensor-level direct data fusion works at the image pixel level, whereas feature level fusion deals with extracted features such as colored areas, boundaries and edges etc., and decision level fusion handles declaration of objects from other object recognition algorithms.

From sensor-, feature-, to decision-level sensor fusion, the communication bandwidth and computation power requirements become less demanding at the price that all available information is potentially less fully used[9], so the system configuration flexibility increases. The decision level fusion is most suitable for systems that have physically distributed components and require these components to work more independently.

On the other hand, as discussed earlier, of the three sensor fusion types, complementary and cooperative sensor fusion types are usually highly domain and task specific. They are implemented typically as feature-level data processing in a complex physical model, to use the information meanwhile without consuming too much communication bandwidth. In fact, sensor fusion is usually implemented in only a few system architecture patterns [155]. Table 6 shows the commonly implemented sensor fusion architectural patterns and identifies the application characteristics that make them desirable ([60], [156]).

[9] Extracting features from raw sensory data and synthesizing features to make decisions gain higher-level information, but the condensed higher-level information has lost other potential means of utilizing the lower-level information. Conceptually, the earlier the lower-level information is discarded, the higher risk is the potentially useful information is lost.

Roughly speaking, the competitive type of sensor fusion algorithms are more suitable for the sensor data-level and decision-level fusion problems. The lower-level information is kept longer and potentially better used in the sensor data-level fusion case, but the decision-level fusion model provides more system configuration flexibility in implementation. So to form a generalizable context sensing solution, a competitive-type, decision-level model of sensor fusion architecture (shown with a different cell border in Table 6) is selected as the system architectural support scheme.

Table 6. Property highlight of commonly implemented sensor fusion architectural patterns

		sensor fusion implementation model		
		sensor level	feature level	decision level
sensor fusion type	competitive	enhance reliability		flexible system configuration
	complementary & cooperative		allow complex model, to fully use information, while control bandwidth	

2.2.3 Sensor fusion methods for context-aware computing

Many well-developed algorithms can be applied to the competitive type of sensor fusion. The commonly used sensor fusion methods are: classical inference, Bayesian inference, Dempster-Shafer theory of evidence, voting, and fuzzy logic. This section examines these commonly used sensor fusion methods in order to choose one as a module for building a generalizable sensor fusion system.

2.2.3.1 Classical inference and Bayesian inference method

The classical inference method and Bayesian inference network method are often referred as the "classical" or "canonical" sensor fusion methods because not only are they the most widely used, but also they are the bases of, or the starting points for, many new methods.

Classical inference methods seek to judge the validity of a proposed hypothesis based on empirical probabilities. Given an assumed hypothesis H_i (a contextual fact is true or an event has happened), the joint probability P that an observation E_k would be reported by the sensors is:

EQ. 3 $$P(E_k \text{ would be observed} \mid H_k \text{ is true})$$

Many decision rules can be used to form the judgment in the classical inference method ([63] Section 6.2). For example, the likelihood comparison rule suggests accepting the hypothesis H_i if the probability relationship satisfies EQ. 4, otherwise, the system should believe that the contextual fact or event is not true or has not happened.

EQ. 4 $$P(E_k \mid H_i) \cdot P(H_i) > P(E_k \mid \neg H_i) \cdot P(\neg H_i)$$

Another example of the decision rules is to use statistical significant test techniques. In the case where there are several alternative hypotheses, then the joint probability for each hypothesis needs to be computed and the results compared.

The classical inference method quantitatively compares the probability that an observation can be attributed to a given assumed hypothesis. But it has the following major disadvantages ([23] page 53): (1) difficulty in obtaining the density functions that describe the observables used to classify the object, (2) complexities that arise when multivariate data are encountered, (3) its capability to assess only two hypotheses at a time, and (4) its inability to take direct advantage of a priori likelihood probabilities.

Bayesian inference overcomes some of these limitations by updating the likelihood of a hypothesis given a previous likelihood estimate and additional new observations. It is applicable when two or more hypotheses are to be assessed.

Given the observed phenomena or evidence E, Bayesian inference calculates the likelihood $P(H_i|E)$ that the contextual fact or event H_i should be true or should have occurred in the form of ([23] Section 4.2):

EQ. 5
$$P(H_i \mid E) = \frac{P(E \mid H_i)P(H_i)}{\sum_j P(E \mid H_j)P(H_j)}$$

where, $P(H_i)$ is the a priori probability that the contextual fact or event H_i has occurred; $P(E|H_i)$ is the likelihood that the phenomenon or evidence E can be observed given the contextual fact or event H_i has occurred.

Compared with the classical inference method, the Bayesian inference network method provides the following advantages: (1) given new observations, it incrementally estimates the probability of the hypothesis being true, (2) the inference process can incorporate the a priori knowledge about the likelihood of a hypothesis being true, and (3) when empirical data are not available, it permits the use of subjective probability estimates for the a priori of hypotheses[10].

Despite these advantages, Bayesian inference method also has some disadvantages that prevent it from being used in many situations. The key limits ([23] page 53) are: (1) difficulty in defining a priori probabilities, (2) complexities when there are multiple potential hypotheses and multiple conditionally dependent events, (3) mutual exclusivity required for competing hypotheses, and (4) inability to account for general uncertainty. The meaning of the last two points will be clarified when the Dempster-Shafer Evidence Theory is discussed in next subsection.

[10] This situation happens very often in practice, where the a priori probabilities of hypotheses cannot be easily obtained and thus are guessed. However, the output of such a process is only as good as the input a priori probability data ([23], page 85).

Because of the limitations, the Bayesian inference method is considered unsuitable for the main sensor fusion method for context-aware computing applications. There are two practical challenges that need to be addressed in this domain. First, the system users' context information is typically not repeatable in nature[11], which means that the prior probability information $P(H_i)$ is hardly available. Second, when multiple sensors are used to sense some specific contextual information, the symbol E in EQ. 5 represents the joint observation over all the sensors; however, since the available sensor set's configuration is typically highly dynamic, the joint probability distribution function $P(E|H_i)$ is usually unavailable.

2.2.3.2 Dempster-Shafer Theory of Evidence method

The Dempster-Shafer method generalizes Bayesian theory to allow for distributing support not only to single hypothesis but also to the union of hypotheses. This way, it easily includes uncertainty in the likelihood function and acknowledgement — and even quantification — of ignorance [31]. The Dempster-Shafer and Bayesian methods produce identical results when all the hypotheses are singletons (not nested) and mutually exclusive [143].

In a Dempster-Shafer reasoning system, the possible basic hypotheses — all the hypothesis elements that are not further dividable, mutually exclusive, and exhaustive — are collectively called "the frame of discernment" (T). The system inference space is the power set (Θ) of T, which includes all the possible combinations of the elements of T. For example, in an instrumented room, a human user is detected, and from the reality constraints, this detected user can normally only be one of the three registered users: User-A, User-B, or User-C. The task is then to discern this user's identity. In this case, the "frame of discernment" is T = {User-A, User-B, User-C} and the valid hypothesis set space consists of the eight possibilities listed as EQ. 6:

[11] For example, a group of people would noticeably behave differently the second time they play a game — even they are playing the same game.

EQ. 6 $\Theta = \{A, B, C, \{A, B\}, \{B, C\}, \{A, C\}, \{A, B, C\}, \{\phi\}\}$ **.**

Symbols in EQ. 6 indicate the eight situations of: (1) User-A, (2) User-B, (3) User-C, (4) either User-A or User-B, (5) either User-B or User-C, (6) either User-A or User-C, (7) any one of the User-A, User-B, and User-C, and (8) neither User-A, nor User-B, nor User-C. Notice that the situation (7) is actually an indication of ignorance and situation (8) is an indication of exception.

With the frame of discernment *T* defined, the system's possible hypotheses Θ defined, "belief" can be assigned over Θ. Analogous to probability, the total belief equals to value 1, and each sensor S_i reports its observation by assigning beliefs. This assignment function is called the "probability mass function" of S_i, denoted as m_i. The belief assignment is based on the observed "evidence" (*E*) that supports the belief.

For any given hypothesis *H*, the system's belief in *H* is the sum of all the evidence E_k objects that support *H* and the sub-hypotheses nested in *H*:

EQ. 7 $$Belief_i(H) = \sum_{E_k \subseteq H} m_i(E_k)$$

On the other hand, the evidence objects that support *H*'s exclusive hypotheses (i.e., the hypotheses that do not include any sub-hypotheses nested in *H*) are then actually the evidences that are against *H*. Therefore, the "plausibility" of hypothesis *H* should include all the observed evidence objects that do not argue against *H*:

EQ. 8 $$Plausibility_i(H) = \sum_{E_k \cap H \neq \phi} E_k = 1 - Belief_i(\overline{H}) = 1 - \sum_{E_k \cap H = \phi} m_i(E_k)$$

Thus in the Dempster-Shafer reasoning system, according to a sensor S_i's observation, the belief regarding a hypothesis is measured by a "confidence interval" bounded by its basic belief and plausibility values (as shown in Figure 8):

EQ. 9 **[** $Belief_i(H)$, $Plausibility_i(H)$ **].**

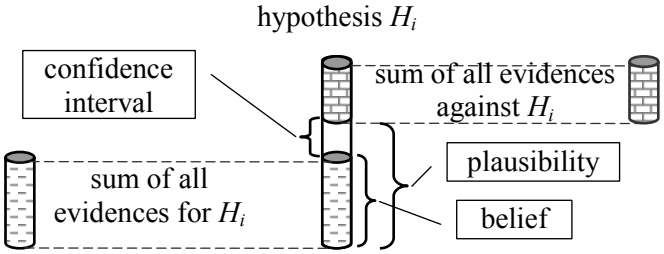

Figure 8. Confidence interval is between "belief" and "plausibility"

Of course, there can be more than one sensor in a system. When there are multiple sensors in a system and the sensors' observations are assumed independent of each other, the Dempster-Shafer Evidence combination rule provides a means to combine these observations. For each hypothesis in Θ — e.g., a proposition, such as "the detected person is User-A" — the rule combines sensor S_i's observation m_i and sensor S_j's observation m_j as[12]:

EQ. 10
$$Belief(A) = m_i \oplus m_j(A) = \frac{\sum_{A_k \cap A_{k'} = A} m_i(A_k) m_j(A_{k'})}{1 - \sum_{A_l \cap A_{l'} = \phi} m_i(A_l) m_j(A_{l'})}$$

In equation EQ. 10, the combined proposition A stands for the intersection of the sensor S_i observed hypothesis A_k and sensor S_j observed hypothesis $A_{k'}$, whose associated probability mass functions are represented as $m_i(A_k)$ and $m_j(A_{k'})$ respectively. For example, {User-A} ⋂ {User-B} = $\{\phi\}$, {User-A} ⋂ {User-A, User-B} = {User-A}, and {User-A, User-B} ⋂ {User-B, User-C} = {User-B}.

In equation EQ. 10, the numerator calculates the probability mass function value of the products of two sensors' observed evidence objects that generate proposition A, and

[12] Adopted from [18], symbol ⊕ has a high precedence: $m_i \oplus m_j(A) = \{m_i \oplus m_j\}(A)$.

sums them up. To properly normalize the result, this sum of the products is divided by the denominator, which accounts for all the impossible proposition {Ø} combinations that have been assigned with non-zero probability mass functions.

Notice that this evidence combination rule is both associative and commutative [128]. This means that the probability mass functions $m_i(A_k)$ in EQ. 10 can be the results of previously combined evidence, so the process of combining evidence from multiple sources can be chained, and the order in which the sources are combined does not affect the final results.

Bayesian methods and Dempster-Shafer methods are the most commonly used formalisms in MSDF (Multi-Sensor Data Fusion) in the military applications domain. The main reason that these two formalisms in particular have received so much attention is that both are associative and commutative, so the results are independent of the order in which the data are received and incorporated [54].

While both the Bayesian inference method and the Dempster-Shafer method can update a priori probability estimation with new observations to obtain a posteriori estimations, the Dempster-Shafer method relaxes the Bayesian method's restriction on mutually exclusive hypotheses, so that it is able to assign evidence to "propositions", i.e. unions of hypotheses.

The underlying concept is that Dempster-Shafer uses a general level of uncertainty ([63] page 167, [134], [135], [136]). This topic will be addressed in more detail in Chapter 5, briefly now, the main point is that it does not require an exhaustive set of hypotheses, each of which is defined in terms of definite probabilities. Thus, as an extension of the Bayesian inference method, the Dempster-Shafer method largely overcomes the Bayesian's limitations listed in the previous subsection. Because of its capability to solve general problems, in this dissertation, the Dempster-Shafer method is selected as the sensor fusion method for building context-aware computing systems.

2.2.3.3 Voting fusion method

Voting sensor fusion imitates voting as a means for human decision-making. It combines detection and classification declarations from multiple sensors by treating each sensor's declaration as a vote, and the voting process may use majority, plurality, or decision-tree rules. The most commonly used voting architecture is a Boolean combination of outputs from multiple sensors, where additional discrimination can be introduced via weighting each sensor's specific declaration ([23] Chapter 7, [63] Section 6.6).

The principle of the underlying mechanism of voting fusion is estimation of the joint detection probability based on the participating sensors' detection confidence levels, which are in turn based on predetermined detection probabilities for an object or an event. Given that all sensors' observations are independent and non-nested, the probability that a hypothesis is true can be estimated as illustrated by the following example.

For the proposition the context fact H_k is true or event H_k occurs, the inputs of voting fusion are the sensor $\boldsymbol{s_i}$ and $\boldsymbol{s_j}$'s detection probabilities $P_i(H_k)$ and $P_j(H_k)$, and their false alarm probabilities $P_{fai}(H_k)$ and $P_{faj}(H_k)$. The outputs of the voting algorithms are the detection probability $P(H_k)$ and the false alarm probability $P_{fa}(H_k)$, as shown in EQ. 11 and EQ. 12.

EQ. 11 $$P(H_k) = P_i(H_k) + P_j(H_k) - P_{i \cap j}(H_k)$$

EQ. 12 $$P_{fa}(H_k) = P_{fai}(H_k) + P_{faj}(H_k) - P_{fai \cap j}(H_k)$$

The voting method greatly simplifies the sensor fusion process, and it can provide a prediction of object detection probability as well as false alarm probability. However, voting fusion is more suitable with "yes/no" problems like the classical inference method. This granularity of reasoning, generally speaking, is not good enough for multiple status context discrimination, which is often required in context-aware computing applications. For a multiple status problem to be solved using the voting method, it has to be converted

into multiple “yes/no” problems first. Further, the more serious disadvantage inherent in the voting fusion method is that it treats each “yes/no” problem separately rather than taking them as a whole package, as the Dempster-Shafer method does (thus, the available information can be better utilized in the Dempster-Shafer sensor fusion framework).

2.2.3.4 Fuzzy logic method

The fuzzy logic method accommodates imprecise states or variables. It provides tools to deal with context information that is not easily separated into discrete segments and is difficult to model with conventional mathematical or rule-based schemes. One example of such information kind is the room temperature: though it is commonly referred to with some descriptive words like “cold”, “warm”, or “hot”, it does not have hard boundaries between these states.

There are three primary elements in a fuzzy logic system, namely, fuzzy sets, membership functions, and production rules.

Fuzzy sets consist of the imprecisely labeled groups of the input and output variables that characterize the fuzzy system, like the “cold”, “warm” and “hot” status in the above example of room temperature.

Each fuzzy set has an associated membership function to provide a representation of its scope and boundaries. A variable of a fuzzy set takes on a membership value between the limits of 0 and 1, with 0 indicating the variable is not in that state and 1 indicating it is completely in that state. An intermediate membership value means a “fuzzy” state, somewhat between the “crisp” limits. A variable may belong to more than one fuzzy set. For example, a room temperature of 90°F may be regarded simultaneously as 0.25 “warm” and 0.65 “hot”.

Production rules specify logic inference in the form of IF-THEN statements, which are also often referred to as fuzzy associative memory. The basic algorithm is that the “AND” operation returns the minimum value of its two arguments, and the “OR” operation returns the maximum value of its two arguments. The output fuzzy set is

defuzzified to convert the fuzzy values, represented by the logical products and consequent membership functions, into a fixed and discrete output that can be used by target applications.

Regarding human-users' contextual information, there is a broad range of "fuzzy" situations, where the boundaries between sets of values are not sharply defined, events occur only partially, or the specific mathematical equations that govern a process are not known. With its capability of dealing with this kind of information, and with its cheap computation to solve very complicated problems[13], the fuzzy logic method is expected to develop extensively in some context-aware computing applications.

The fuzzy logic sensor fusion method provides an effective tool to handle requirements of human daily-life, where imprecision is an inherent property in nature. However, the fuzzy logic sensor fusion method cannot be the main sensor fusion method in a generalizable architectural solution in building a context-aware computing system for two reasons. First, it is not applicable to situations where the objects inherently have clear-cut boundaries (e.g., it does not make sense to say, this is 0.6 person-A and 0.4 person-B). Second, the fuzzy set, membership function assignment, and production rules are usually extremely domain- and problem-specific, making it difficult to implement the method as a general approach.

2.2.3.5 Neural network method

Neural networks open a new door for fusing outputs from multiple sensors. A neural network can be thought of as a trainable non-linear black box suitable for solving problems that are generally ill defined and that otherwise require large amounts of computation power to solve.

A neural network consists of an array of input nodes to accept sensors' output data, one or a few output nodes to show sensor fusion results, and sandwiched in between the

[13] Fuzzy logic is especially successful in solving control problems of very complicated non-linear systems.

input and output nodes is a network of interconnecting data paths. The weights along these data paths decide the input-output mapping behavior, and they can be adjusted to achieve desired behavior. This weight-adjusting process is called training, which is realized by using a large number of input-output pairs as examples.

The neural network training process can be simplified as follows. From the input nodes to output nodes, the data-path network provides many ways to combine inputs: those that lead to the desired output nodes are strengthened, whereas those that lead to undesired output nodes are weakened. Thus, after using the large number of input-output pair as training examples to adjust weights, the input data are more easily transferred to desired output nodes through the strengthened paths.

The neural networks can work in a high-dimensional problem space and generate high-order nonlinear mapping. Many successful applications have been reported. However, it has some well-known drawbacks too. The three major problems are (1) it is difficult to select a network architecture that reflects the underlying physical nature of the particular applications; (2) training a network is typically tedious and slow, and (3) training can easily end up with local minima, as there is no indication whether the global minimum has been found [55].

The neural network method is not suitable for the main sensor fusion method mainly because of the drawbacks. First, the mapping mechanism is not well understood even if the network can provide the desired behavior — only in the simplest toy-like problems does examination of the weights in the trained network give any clue as to the underlying analytical connection between the inputs and outputs. Thus, such a solution cannot be easily generalized. Second, the neural network method is, generally speaking, not suitable to work in a dynamic sensor configuration environment, because each sensor needs a unique input node and each possible sensor-set configuration needs to be specifically trained. Third, the neural network sensor fusion method has the “local minimum problem” during its training process, which cannot be easily overcome.

2.3. Chapter Summary

Since *context* can be an extremely broad term including anything from low-level parameters (e.g., time, temperature, etc.) to highly abstract human concepts (e.g., intention, social relationship, etc.), state-of-the-art technology can only sense a small fraction of what we call context. To approach this daunting task, this dissertation proposes a top-down methodology via a divide-and-conquer strategy. First, decompose complex and abstract context information into an information architecture comprised of lower-level discrete facts and events. Second, use the Dempster-Shafer Theory of Evidence as the sensor data fusion means to resolve conflicts in the process of mapping sensor outputs into the context information architecture.

Building the context information architecture, i.e., decomposing complex context information into discrete facts and events, has no unique solution, and there is no standard or guideline for a conventional approach. Using the results of the described information taxonomy development as a reference, the strategy adopted is to carefully model any given context-aware application scenario, and then to arrange its content into one flat layer so that new information can be added with minimum impact.

Once the context information architecture is completely defined, the possible mapping from sensors' output to the context information entries is also mostly defined, so sensor fusion becomes a less difficult problem. The main remaining challenge in sensor fusion for context-aware computing is that, because the sensor sets' content, characteristics, and configuration are dynamic, and because human-computer interactions are not precisely repeatable, the sensors' joint observation probability distribution is unknown, so the classical sensor fusion methods (e.g., the Bayesian inference network method) cannot be used.

The Dempster-Shafer Theory of Evidence method is the proposed way to meet these challenges. For every context item in the context information architecture, all possible hypotheses, including ambiguous cases and ignorance, are enumerated. The sensors report their observations by assigning mass probability values to all possible hypotheses.

The reports are collected by sensor fusion mediators — each sensor fusion mediator in charge of each context item. The Dempster-Shafer Evidence Combination Rule allows quantitative chaining of separate inferences, cumulatively, regardless of the order in which the information is received, and robustly against changes in the sensor set.

In implementing sensor fusion, the earlier the lower-level information is condensed and the details discarded, the less demand will there be on system communication bandwidth. To define the context information architecture is thus to choose a balance point between how the totality of available information can be best used versus how flexible and robust the sensor fusion system can be.

The context sensing methodology set out in this chapter is a generalizable solution: it is non-domain-specific, non-application-specific, and easily scalable.

Chapter 3.
Implementing Context Sensing

To realize context sensing, there are two major concerns that need to be addressed carefully. One is how to make the system architecture efficient in collecting and distributing context information. The other is how to use sensor fusion technology to deal with uncertainty in the sensed context information, that is, how to handle information overlap and to resolve conflicts. This chapter first discusses issues regarding how to implement a layered and modularized system architecture using the Context Toolkit system, then, it addresses the questions regarding how the Dempster-Shafer Theory of Evidence concept can be applied in practice to the sensor fusion tasks.

3.1. System Architectural Support

At current stage of context-aware computing research, because the number of sensors being used is not very large, sensor fusion is still manageable in an ad hoc manner. That is, for a given set of sensors, some system infrastructure components, or the applications that directly use these sensors, are able to perform sensor fusion functions, as well as to take care of the sensors' administrative management issues.

With the number of sensors increasing, and with the contextual information becoming more complicated, it soon will be cost-prohibitive to maintain and extend such a system. System architecture support is required to fulfill the increasingly more complex context sensing tasks [4].

3.1.1 System architecture style for context-aware computing

Over years of practice and research, experiences are accumulated and summarized in building complex software systems. They are described as system architecture styles: a system architecture style is believed to be effective for some typical application situations [151]. Because context-aware computing is a relatively new research area, the characteristics of its applications have not been fully understood yet, thus systematically developing system architecture styles specifically for context-aware computing applications has not yet been done [46].

Winograd [115] summarized that there are three major styles (or, "models", in his term) for context-aware computing that have been developed thus far. They are the "blackboard", the "infrastructure", and the "widget" style. The key advantages and drawbacks of these three architecture styles are listed as follows.

3.1.1.1 The blackboard-style system architecture

The blackboard style of system architecture is characterized by its component communication style, analogous to the scenario that a group of human experts collaborate to solve difficult problems using a blackboard as their communication means. The system consists of three components: a "blackboard", many "knowledge sources", and a "control shell". The blackboard is a shared memory where context information is written, read, and erased. The "knowledge sources" are independent software agents that provide specific expertise to process the context information; each knowledge source behaves as if it were a human specialist watching the blackboard, looking for an opportunity to apply its expertise to synthesize information pieces or to extract higher-level context. The "control" is the component that monitors the changes in the blackboard and decides what actions (e.g., notifying knowledge sources of change) to take. ([152], [153])

The blackboard architecture style systems are built around databases that coordinate information across the components, their implementation includes a rich group memory, where integrating artifacts (rationale, stakeholders, etc.) into a common knowledge space

is relatively easy. The systems' uniform communication path is advantageously simple. In addition, because each knowledge source can have partial information, and new sources can be added easily, the blackboard-style architecture is robust to configuration change.

The blackboard architecture style is very successful in opportunistic problem solving (especially in artificial intelligence research of 1970s and 1980s), but it is relatively less adopted for context-aware computing systems. The reason is that context-aware computing systems typically are highly distributed, the high communication bandwidth requirement and the difficulty to implement shared memory over distributed components are the key drawbacks that prevent the blackboard architecture style from being widely used in context-aware applications.

Winograd et al are the proponents of this style, and in the implemented system in their Interactive Workspace research project, all communications go through a centralized server, where routing information to different components is accomplished by matching message content to a subscriber's pattern. Braun et al's [108] iBistro project uses a "distributed concurrent blackboard" architecture, where there are many local blackboards in the system and small data objects are completely replicated whereas larger data objects are accessed directly in small chunks. This can mitigate the communication difficulty in some situations – depends on how much cross interactions are needed among the local blackboards, however, a new problem may arise as data objects' version control would be very difficult.

3.1.1.2 The infrastructure-style system architecture

"An infrastructure is a well-established, pervasive, reliable, and publicly accessible set of technologies that act as a foundation for other systems" [113]. The "infrastructure" style architecture approach for context-aware computing tries to simplify the tasks of creating and maintaining context-aware systems by shifting the functions of context-aware computing realization onto network-accessible middleware infrastructures. By providing uniform abstractions and reliable services for common operations, such a

service-oriented infrastructure should make it easier to develop robust context-aware applications even on a diverse and constantly changing set of devices and sensors [113].

Most of the context-aware systems developed thus far use a form of service-based (e.g., client-server) mechanism. The reason that "infrastructure architecture style" was not explicitly stated as one of their system design goals (as declared by Hong and Landay in [113]) is that context-aware computing is a new research area, where "uniform abstractions" have not yet been established and "common operations" have not been clearly identified by the context-aware computing developers' community.

The key advantage of the infrastructure-style system architecture is the independence of the components. Through pre-configuration or resource discovery, a client can find the services it needs and then set up a connection. The drawback is that, in a distributed environment, finding services' location (host and port) and communicating with the independent services are inherently higher-cost compared with component-tightly-coupled systems [115].

3.1.1.3 The widget-style system architecture

"Widgets" can be thought of as an extension of device drivers. For example, a scroll-bar widget on a graphical user interface is a device driver at a higher abstraction level — the program using it can treat it as an abstract device that visualizes and controls a 1-dimensional position. The inside-components' interaction is based on messages to the widgets and callbacks from the widgets.

Traditional widgets belong to some controller; and the widgets, the controller, and other underlying components are tightly coupled. For example, a GUI (Graphic User Interface) software package with widgets is compiled together and is an interface to one operating system. This tight coupling is highly efficient in message dispatching and callback looping. Because of the tight coupling of the system components, there is less flexibility for system components' evolution.

The Georgia Tech's Context Toolkit system [22] is regarded as the exemplary architecture of widget style systems for context-aware computing. As thus, the system is not flexible enough to evolution [113] and not robust enough to tolerate component failures [115]. However, this is not necessary the final status, this dissertation uses the Context Toolkit system as a middleware building blocks to build a service-based system that is of infrastructure architecture style with some blackboard-style advantages in context information management.

3.1.2 Improving the Context Toolkit system

Software *toolkits* are collections of reusable software components suitable to support a class of applications [106]. For example, to build a GUI (graphic user interface), toolkits may provide buttons and checkboxes that can handle event and message dispatching tasks. Using this approach, the "Context Toolkit" system developed by the Georgia Institute of Technology offers a small set of generic base classes for context-aware computing.

Previously developed context-aware computing systems were typically built in such a way that their system architectures heavily depended on the sensors they used. In other words, the sensors and sensed context information are highly coupled in those systems. As described in Subsection 1.3.3, the "Context Toolkit" system supports separating context information from the sensors that sense the context information through using sensor widgets, modularizing and standardizing software interfaces. Compared with previous ad hoc approaches, this system design concept makes it easier to replace existing sensors and/or to add new sensors.

However, as discussed in the previous subsection, the Context Toolkit system can further evolve. The point is that the Context Toolkit system needs not necessarily be regarded as a system architecture style of its own — it can be used as middleware to build a new system architecture style that combines the advantageous features of the infrastructure and the blackboard system architecture styles.

The Context Toolkit system architecture improvements start with sensor fusion support. Since supporting sensor fusion was not one of the primary goals in its original design, the Context Toolkit system lacks the means to address the uncertainty and ambiguity problems, thus it cannot easily handle information overlapping and conflicts from multiple information sources. To address this problem, described in detail in Chapter 2, a top-down methodology is proposed.

The first step is to define the context information architecture, where uncertainty and ambiguity are treated as intrinsic properties of the context information. In this new information management scheme, beliefs are assigned to all possible values regarding the given context information pieces. Then, in the second step, sensor fusion mediators use the Dempster-Shafer evidence combination rule.

To realize this uncertainty management scheme, the Context Toolkit system architecture is modified as illustrated in Figure 9. Given that, according to a simplified context model (the information architecture), the context information has been decomposed into discrete facts and events, the sensor widgets of the Context Toolkit system are modified to generate only specific contextual observations, the hypotheses, according to the context information model. All the possible hypotheses are estimated by the sensors in terms of assigned beliefs (probability mass functions). For each of the predefined context information items, there is a sensor fusion mediator to combine the hypotheses' beliefs from multiple sources. Overlapping and cross-verifying information will increase the overall estimation confidence, whereas conflicting signals will decrease the estimation confidence.

For every piece of the context information starting from the low-level sensor output, there is always an associated time stamp to indicate when this information piece was sensed or deduced. Therefore, even the sensor hypothesis estimates may arrive at any time in any order, based on the nature of the sensing task and the time stamps, the sensor fusion mediators can decide whether it makes sense to fuse the information pieces of different time stamps, and when and how to update the information.

The sensor fusion mediator behaves as a special functionally enhanced Context Aggregator in the original Context Toolkit system. It also keeps and updates a list of all sensors that can generate its specific hypotheses. The sensor list is acquired from the system Resource Discoverer at initialization process. Subsequently, for the periodically activated sensors, their status is known by their occurring reports, and for the event-triggered sensors, their status can be known via a polling operation.

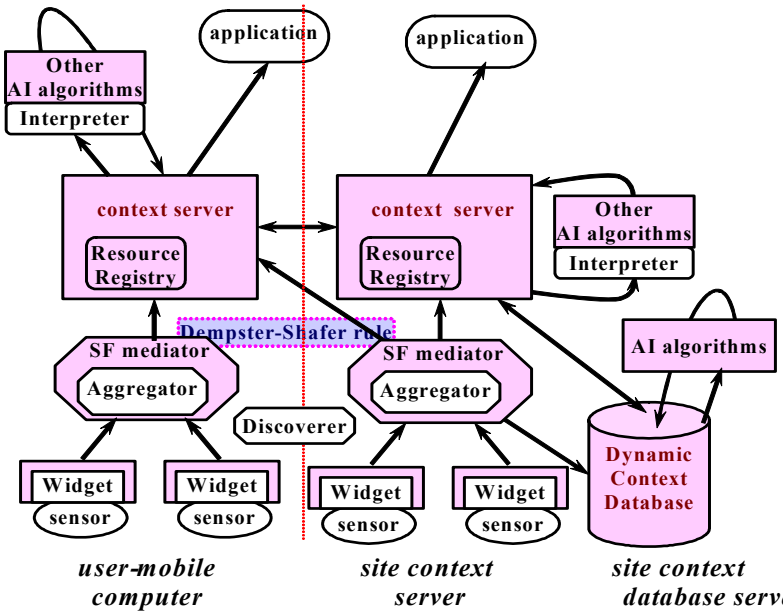

Figure 9. System architecture to support sensor fusion in context-aware computing

The sensor widgets and their corresponding sensor fusion mediator form a client-server configuration, meanwhile the relationship between the sensor fusion mediators and their corresponding context information server is also client-server. Thus, the modified system is a service-based infrastructure-style architecture processing information at two abstraction levels. Consequently, referring back to the discussion in last subsection, it should have the advantageous characteristics of the infrastructure system architecture style.

The context information servers illustrated in Figure 9 are agents of significant entities in the context information model, which collect all information regarding those corresponding entities from the relevant sensor fusion mediators. The context information servers provide a central storage buffer for their entities. In addition, all context information is centrally stored in a dynamic context database as shown in Figure 9, as the context interpreters thus no longer directly work with context widgets. The context information servers are the central information repository regarding specific objects; as such the dynamic context database is the central information repository for all available context. These information repositories resembles a blackboard-style system architecture where sensor fusion mediators write their output to the repositories and context interpreters read from and write to the repositories to transfer context information description formats or to extract higher-level context information via synthesizing multiple lower-level context information pieces. As discussed in the previous subsection, this system configuration would have the same advantages as the blackboard architecture style.

3.1.3 Benefits from the system architecture improvement

Because the original Context Toolkit system design did not address measurement uncertainty, adopting the uncertainty management scheme (using the Dempster-Shafer statistics reasoning framework) enables the system to migrate from the yes/no binary realm into a “statistical reasoning” realm.

Figure 10 shows the enhanced system architecture from an information flow perspective. It can be seen that information processing occurs at three layers.

The lowest layer consists of the sensor widgets that behave as the sensors’ agents — collecting sensors’ raw data to generate hypothesis estimations as predefined by the system information architecture.

The second layer is the sensor fusion mediators that consolidate the hypothesis estimations from relevant sensor widgets, i.e., re-evaluate associated confidences, combine compatible evidences and resolve conflicts.

The highest layer is the context information servers and the dynamic context database. The context information servers collect relevant context information regarding specific objects and the dynamic context database provides a central storage for all available context. with the context database, interpreters transfer context description formats and derive higher-level contexts, and the system provides context information services. Though not shown in Figure 10, the Dempster-Shafer machinery can also be called to aggregate information in the context database directly.

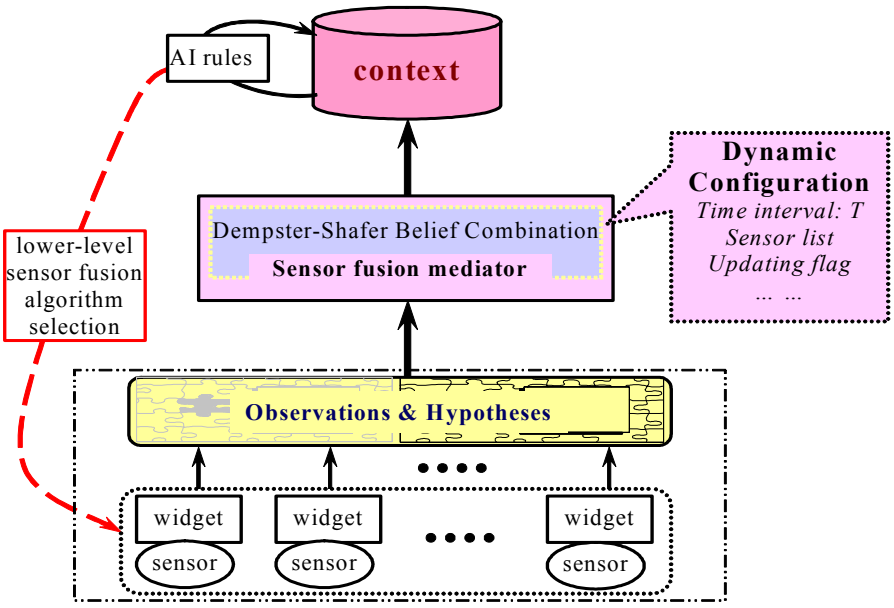

Figure 10. Information layered structure and sensor fusion support

System performance benefits, versus the original Context Toolkit system, from the middle sensor fusion layer, and from some reinterpretation of component functionality is as follows:

- The system can directly support the competitive type of sensor fusion in information mapping: for most applications, the previously difficult sensor fusion task is reduced to just recalculating confidence.

- The system gains the advantageous features of the blackboard architecture style: adopting centralized context aggregation (context information server, backed up with dynamic context databases) makes it easier for artificial intelligent agents to access the context data to derive more abstract, higher-level context information.

- Context information usage is further separated from context data acquisition. The separation makes applications less sensitive to system hardware change, e.g., it becomes easier to replace existing sensors or to incorporate new sensors.

- The dynamic context database service makes it easier to form complex context situational abstraction because all the context data are better organized in a central repository.

Using the Context Toolkit components as the building blocks, the new system is an infrastructure-style architecture, and since the Dempster-Shafer algorithm implementation is an independent module for providing information fusion services in the architecture, this modularized architectural support of context-aware computing is a generalizable one.

3.2. Sensor Fusion with Dempster-Shafer Theory

For given application scenarios, with their context information architectures being defined as discussed in Section 2.1, using the system architecture implementation scheme discussed in Section 3.1, the system can handle arbitrary sensor having different resolutions, accuracies, data rates, and data formats. This section shows how to apply Dempster-Shafer Evidence Theory to effect the sensor fusion, especially to handle

conflicts, given that the mapping between the sensors' output to the context information architecture is already established.

3.2.1 Evidence combination in Dempster-Shafer frame

In the Dempster-Shafer framework, the information sources are called "evidence", and a standard process to combine the items of evidence (described in subsection 2.2.3.2) was established in Shafer's original work [18]. However, not all Dempster-Shafer proponents agree with the original evidence combination rule. The controversial point is how to treat the conflicts among evidence from multiple sources.

3.2.1.1 Challenge to the Dempster-Shafer evidence combination rule

As discussed in Subsection 2.2.3.2, the Dempster-Shafer theory has two main parts: (1) the association of quantitative "belief" and "plausibility" with hypothesis based on evidence that partially supports it and additional evidence that partially refutes it; and (2) an algebra for propagating "belief" and "plausibility" when new evidence is folded into prior evidence. In the original Dempster-Shafer Evidence Combination Rule EQ. 10, for any one of the propositions A, the products of the beliefs, whose associated hypothesis intersection would coincidently be A, are summed together, and the result is then normalized by the denominator as rewritten in EQ. 13:

EQ. 13
$$1 - \sum_{A_l \bigcap A_{l'} = \phi} m_i(A_l) m_j(A_{l'}) = 1 - K$$

where the K stands for the sum of all the probability mass functions that have been assigned to conflicts, or in other words, K accounts for the beliefs that lead to the conflicts.

Using this normalization scheme, the Dempster-Shafer evidence combination rule attributes the conflict-associated probability mass to the null set. Thus, in effect it ignores the conflict completely [128]. However, some researchers argue that this combination rule is catastrophically counter-intuitive when the conflict is significant. The most

commonly used example by the opponents is an extremely significant conflict in disease diagnosis case as described following ([128]).

Suppose that a patient is diagnosed by two physicians for neurological symptoms. The first doctor believes that the patient has either meningitis with a probability of 0.99 ($m_1(meningitis) = 0.99$) or a brain tumor with a probability of 0.01 ($m_1(tumor) = 0.01$). The second doctor believes that the patient actually suffers from a concussion with a probability of 0.99 ($m_2(concussion) = 0.99$) but admits the possibility of a brain tumor with a probability of 0.01 ($m_2(tumor) = 0.01$). So the conflict probability mass is:

EQ. 14

$$\begin{aligned} K &= m_1(meningitis)m_2(tumor) + m_1(meningitis)m_2(concussion) \\ &\quad + m_1(tumor)m_2(concussion) \\ &= 0.99 \times 0.01 + 0.99 \times 0.99 + 0.01 \times 0.99 \\ &= 0.9999 \end{aligned}$$

Now apply the Dempster-Shafer evidence combination rule, either meningitis or concussion will have zero probability mass, but brain tumor will have 0.0001 probability mass, and after normalizing it with *1-K*, the final diagnosis will conclude that the brain tumor has the probability mass of 1.0. This is considered as a surprising, or counter-intuitive, interpretation for combining multiple-source information[14].

Significant conflicts may happen most likely in situations where the application's frame of discernment is not well defined. As the frame of discernment is essentially educated guesses supplied by human experts, the human has a tendency to hedge his bet by assigning a small probability to an unlikely alternative conclusion, which expands the overall frame-of-discernment.

To address this problem, a number of compromised, or modified, evidence combination rules have been proposed, for example the one proposed by Yager and the Inagaki described in the following subsection [131].

[14] However, to be fair, if the patient has no background knowledge regarding neurological disorder but solely relies on the advice of these two equally trusted but contradictory doctors, what can the patient conclude — the compromised solutions discussed later are also questionable.

3.2.1.2 Yager's and Inagaki's modification to evidence combination rule

Instead of assigning the conflicting evidence to the null set, thus in effect it ignoring the conflicts, Ronald Yager suggests assigning the conflicts to the whole frame of discernment, i.e., the ignorance associated set that contains all possible outcomes of the situation.

To rationalize this new combination rule, Yager introduced a new term referred to as "ground probability mass assignment":

EQ. 15 $$q(A) = \sum_{A_k \bigcap A_{k'} = A} m_i(A_k) m_j(A_{k'})$$

which is essentially the Dempster-Shafer's basic probability mass assignment without the normalization by *(1-K)*. This allows non-zero value to be assigned to the null set $q(\phi) = K = \sum_{A_l \bigcap A_{l'} = \phi} m_i(A_l) m_j(A_{l'}) \geq 0$.

To convert the ground probability mass to basic probability mass, Yager adds the conflict-associated null set ground probability mass to the ignorance-associated frame of discernment basic probability mass:

EQ. 16 $$m_Y(\Theta) = K + q(\phi) = q(\Theta) + q(\phi)$$

While Yager's evidence combination rule admits ignorance whenever items of evidence conflict, it is also criticized as not intuitive as each hypothesis' probability mass tends to become very small with new conflicting evidences being combined. In the above classic neurological symptom diagnosis example, Yager's evidence combination rule will conclude that the probability of brain tumor is 0.0001, which the critics argue is too small.

Toshiyuki Inagaki tried to take advantage of Yager's ground probability mass idea, but instead of assigning conflict entirely to ignorance set, he defines a continuous parameterized class of operations that subsumes both Dempster-Shafer's and Yager's

evidence combination rule [128]. Inagaki's unified evidence combination process can be expressed in the following equations:

EQ. 17 $$m_u(A) = [1 + kq(\phi)] \cdot q(A) \qquad \text{for all } A, A \neq \phi,\ A \neq \Theta$$

EQ. 18 $$m_u(\Theta) = [1 + kq(\phi)] \cdot q(\Theta) + [1 + kq(\phi) - k] \cdot q(\phi)$$

EQ. 19 $$0 \leq k \leq \frac{1}{1 - q(\phi) - q(\Theta)}$$

The interpretation is that, while Inagaki's method firstly uses Yager's ground probability mass to combine evidences as in EQ. 15, the result is converted to basic probability mass in a different way using EQ. 17 and EQ. 18. The parameter *k*, whose range is specified in EQ. 19, is used for normalization. At the high extreme, when $k = \frac{1}{1 - q(\phi) - q(\Theta)}$, Inagaki's evidence combination rule is the same as the Dempster-Shafer rule; whereas at the low extreme *k = 0*, it is the same as the Yager rule.

3.2.1.3 Practical solution to resolve evidence conflicts

Inagaki's evidence combination method, parameterized by *k*, may provide a widely acceptable solution to combine significantly conflicting items of evidence from various sources. However, the process of choosing an optimal value for the parameter *k* itself becomes an open research topic that needs experiments, simulation, or good intuition with a deep understanding of the specific applications. However, because the make of the sensor set is volatile, it is difficult to develop a good intuition about the quality of the sensor-based evidences.

On the other hand, given that the information architecture is already defined (Section 2.1), the frame of discernment is already decided, so it is unlikely that the sensor based evidence will contain extreme conflicts. Furthermore, since the applications are for human-computer interactions, even if significant conflicts occur, human intervention is easily available.

The next subsection describes a procedure for attaching a weight to each sensor's judgment. This effectively mitigates conflicts among items of evidence.

3.2.2 Weighting means non-democratic voting

The fundamental Dempster-Shafer combination rule essentially requires that all the items of evidence, or the sensors' reported observations in a sensor fusion system, be independent, consistent, and rational [18]. In other words, in sensor data fusion applications, simply applying the Dempster-Shafer evidence combination rule implies that any two sensors S_i and S_j are trusted equally. While the issues of independence and rationality are critical factors in applying the Dempster-Shafer theory and are still important research subjects [131], in practice, misplaced trust can produce problematic outcomes. Here, a practical solution is proposed, which tries to take advantage of knowledge already gained regarding the sensors' expected performance [31].

In most situations, a sensor's performance can be estimated to some extent based on already known factors. Knowledge about the sensor's general performance (based on its technical specifications) and its current working status, statistical data about its behavior evolution as sensors age, and historical performance records (based on a regular stream of occasional ground-truth observations) can all contribute to estimate the sensor's expected performance. Such performance expectation information should be used to decide the trust of each sensor [35].

This differential trust scheme is realized simply via justifying each sensor's basic probability mass assignments before they are submitted to the Dempster-Shafer reasoning system. Given a sensor S_i's probability mass function m_i, its corresponding trust factor w_i, the weight adjustment process is expressed in EQ. 20 as:

EQ. 20

$$m_i'(A) = w_i m_i(A) \quad \text{for all } A: A \subset \Theta, \text{ and } A \neq \Theta$$
$$m_i'(\Theta) = w_i m_i(\Theta) + 1 - w_i$$

where, the weighting (i.e., trust) factor w_i is in the range between 0.0 and 1.0, the $m_i(\Theta)$ stands for the probability value assigned to the acknowledgement of ignorance, and the term $m_i'(A)$ indicates the adjusted probability mass function to be submitted to the Dempster-Shafer reasoning system.

As the weighting factor is always equal to or less than 1.0, this differential trust scheme effectively lowers the basic probability mass assigned to the non-zero hypotheses by each sensor and correspondingly raises the value assigned to the ignorance set. This will mitigate conflicts among groups of evidence items. Furthermore, the scheme of differential trust of sensors makes it convenient for humans to intervene in cases where mechanical application of the method causes counter-intuitive outcomes.

3.2.3 Dynamic weighting means constant calibrating

When the ground truth is available, e.g., shortly after current measurements or from an additional information channel, it can be used to make the weight factor w_i become a function of time t to account for the sensor's expected performance change [150].

A simple but effective practical implementation is to define the weight function $w_i(t)$ (with backward-looking time step Δt) as:

EQ. 21
$$w_i(t) = (1-\rho)\sum_{n=1}^{\infty} c_i(t - n\cdot\Delta t)\cdot\rho^n$$

where the $c_i(t)$ is the function that describes the correctness of the senor S_i's estimation at time t:

EQ. 22
$$c_i(t) = \begin{cases} 1 & \text{correct estimation} \\ 0 & \text{incorrect estimation} \end{cases}$$

and the ρ, in the range 0.0 to 1.0, is the "remnance factor" that corresponds to how rapidly past performance will be discounted.

This dynamic weighting approach resembles the Kalman filter method in terms of dynamic averaging effects, where recent performance plays a more important role and past performance is gradually forgotten. The dynamic weight $w_i(t)$ is largely decided by the remnance factor ρ: the smaller the ρ is, the faster the past performance is forgotten. The $(1-\rho)$ term in EQ. 21 normalizes the $w_i(t)$ value to the range of 0.0 to 1.0: in the case that the sensor S_i is completely reliable, the right-hand sum will approximate to $\frac{1}{1-\rho}$ and the weight will be equal to 1.0.

To summarize Section 3.2, the original Dempster-Shafer evidence combination rule requires that all combined evidence groups be independent and consistent. But in practice, these conditions cannot be guaranteed. The outcome would be especially problematic when two groups of evidence are in significant conflict. Alternative evidence combination rules have been proposed by other researchers, but they theoretical justification is questionable and there are no generally agreed upon practical solutions. This dissertation argues that, in practice, this is not a big hindrance to applying the Dempster-Shafer method to sensor fusion for context-aware computing because: (1) the sensors observations typically do not contain serious internal conflicts, and (2) the proposed dynamic weighting scheme (of differential trust among sensor observations) actually mitigates the existing conflicts effectively.

Chapter 4.
Concept-Proving Experiments and Results

Following the top-down methodology described in Chapter 2, this chapter describes concept-demonstrating experiments and their results. The experimental task is to discern the instantaneous focus-of-attention of the meeting participants given the video and audio sensors' judgment (probability mass functions, i.e., belief assignments). The main purpose of the exercise is to demonstrate the utility of the system architecture implementation and the effectiveness of the proposed Dempster-Shafer sensor fusion scheme.

4.1. Application scenario and the sensory data

The application scenario can be described as follows: there is a small round table in a small meeting room, where a few people sitting around the table participate in a discussion. For each meeting-participant, the context information of interest is how likely this meeting-participant's focus-of-attention is on each of the other meeting-participants.

With the meeting facility instrumented, several meetings were recorded and analyzed by Rainer Stiefelhagen ([30], [33]). The experimental settings seen by an omni-directional camera at the center of the table is shown in Figure 11. During meeting discussion, the omni-directional camera captures the activities of the four participants. With the omni-directional camera's panoramic video image sequences, a skin-color-based

face detector [32] is used to detect the face locations and then each participant's head pose is estimated from the perspective user view via neural network algorithms.

Figure 11. Meeting-participant's focus-of-attention analysis experimental settings seen from the central omni-camera

For each meeting-participant, a Gaussian model is assumed to describe the head pan angle distribution. The head pose estimated from the video image sequences are thus used to estimate this meeting-participant's focus-of-attention at each moment.

Besides the panoramic video sequences being recorded during the meeting, there is one microphone set up in front of each meeting-participant to record who was speaking at that moment. Based on the relative sound strength, the microphone sensors working together can sense who is speaking at each moment during the meeting. Based on the assumption that the non-speakers focus their attention on the speaker, the information regarding who is talking now and who was speaking in a short-time history then can be used as a hint to deduce whose focus-of-attention is on whom at the current moment.

So a meeting-participant's focus-of-attention is estimated independently by two different sensing modalities, one using the omni-directional camera's image sequences, the other using the microphone sensors' relative sound strength. Their estimation reports were recorded in the format of a series of time-stamped three belief-assignments indicating probabilities that this meeting-participant's focus-of-attention is on his/her left-side, straight-ahead, and right-side person. For the recorded meetings, the video tape footages were hand classified to label the nominal ground truth[15].

The sensor fusion challenge is how to combine the information from the two sources to estimate each meeting-participant's focus-of-attention more accurately.

4.2. Building the context information architecture

This dissertation advocates using a top-down methodology to build context-aware computing systems, and to provide system architectural support for sensor fusion. Given the application requirements, the suggested first step is to build a context information architecture.

Following the methodology described in Section 2.1, for the context sensing system with the small conference room's user activities, or more specifically the meeting-participants' focus-of-attention, as the major concern, it is natural that the conference room should be set as the "stage" object.

To illustrate how to build and use the context information architecture using this focus-of-attention application as an example, the following are some of the tables in the context database:

```
STAGE(StageID, StageName);
STAGE_USAGE(StageID, Functionality, UtilizationRule);
EQUIPMENT(StageID, EquipmentID, Status, … …);
```

[15] The ground truth is "nominal" because even human observers will not agree on each participant's focus-of-attention 100% of the time.

STAGE_USERS(*StageID*, *Time*, *UserID*);

STAGE_USERS_ACTIVITY(*StageID*, *Time*, *GroupActivityID*);

USER(*UserID*, FirstName, LastName, Sex, BirthDate);

USER_TITLE(*UserID*, *JobTitle*, Affiliation);

USER_ACTIVITY(*UserID*, *Time*, *UserActivityID*);

USER_ACTIVITY_MEETING_FOCUS_OF_ATTENTION(*UserID*, *Time*, UserID#1, UserID#2, UserID#3);

where, each item in parenthesis stands for a column attribute in its parent table, those in italic font style are foreign keys, and those in italic and underlined font style are primary keys. The domain definition and constraints are not shown here, details can be found in Appendix B.1.

In the case that there are four people participating in a meeting discussion around the small conference table as shown in Figure 11, for each meeting-participant, his/her focus-of-attention hypothesis can be "on the left-side person", "on the person straight across the table", and "on the right-side person". Correspondingly, regarding each meeting-participant's focus-of-attention context information, the observation report from the omni-directional video camera sensor and from the microphone-set sensor can be in a format of three probability assignments $[p_L, p_S, p_R]$.

This is a good example that illustrates how the context information architecture defines the hypotheses or propositions of the context information ("frame of discernment" in Dempster-Shafer theory) and its format, with which the sensors report their observations (or, called "evidence" in the Dempster-Shafer theory). The next step is to build the system architecture that can support combining these evidences.

4.3. Implementing context-sensing architecture

According to the defined context information architecture, the omni-directional camera video sensor behaves as four image sensors; each one is in charge of sensing a

meeting-participant's focus-of-attention. Similarly, the microphones working together behave as four sensors; each one is in charge of sensing a meeting-participant's focus-of-attention also.

As previously described (Section 3.1), the Georgia Tech's Context Toolkit system is the basic building blocks to construct new context-aware computing systems. The configuration block diagram of a complete system with complicated context information architecture and context-sensing technologies was illustrated in Figure 9. The specific implementation of this general architecture, as shown in Figure 12, is significantly simpler because the scenario of our experiments is correspondingly simple.

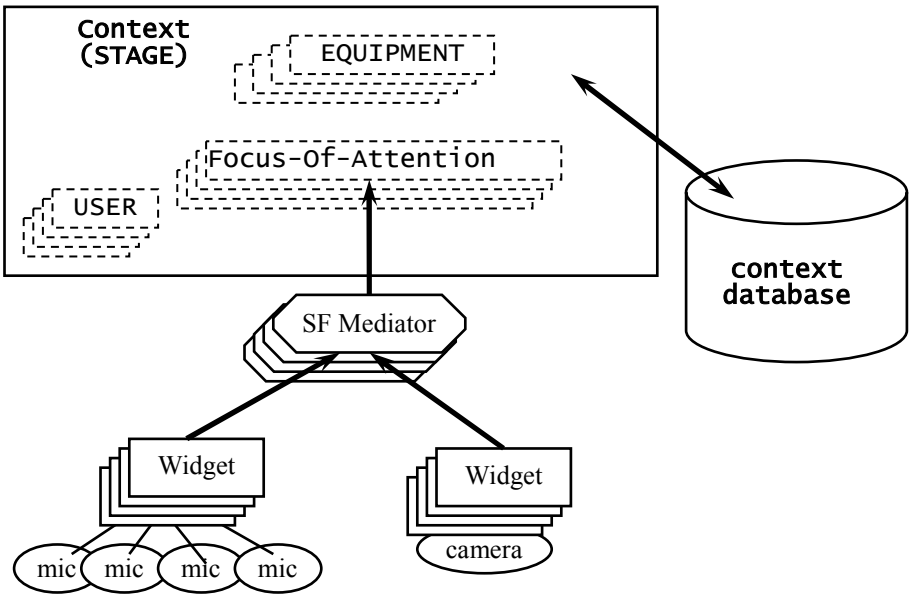

Figure 12. Concept-demonstration system architecture implementation using the focus-of-attention scenario as central application

The partially overlapped blocks in Figure 12 indicate that there are multiple instances of such components in a practical implementation system if the application scenario

requires more than one meeting-participants' focus-of-attention to be sensed, although the concept-demonstrating implementation only realizes one such context-sensing component set.

In this demonstration system, there are two context widgets for the meeting-participant of current concern. One widget is in charge of video image-sensing modality: its input may be raw video image raster data or preprocessed visual features, and its output is an XML encoded ASCII stream consisting of a series of time-stamped focus-of-attention hypotheses and beliefs. The other one is in charge of audio-sensing modality: its input is audio waveforms from the microphones, and its output is an XML encoded ASCII stream consisting of a series of time-stamped focus-of-attention hypotheses and beliefs too.

Separating concerns of context sensing and context usage was one of the primary goals in the development of Context Toolkit system. Thus, how the context widgets interact with sensors to collect raw sensory data is independent of the rest of the system. Consequently, for the purpose of system implementation demonstration, in principle it does not make any difference to the rest of the system whether real physical sensors or artificially simulated sensors are used — as long as the widgets-to-context interface is well defined and properly implemented. The demonstration system actually uses simulated sensors that read prerecorded data (as described in Section 4.1) from a text file as their information source.

As previously described, the audio and video widgets report their estimation of the meeting-participant's focus-of-attention in the format of three belief (probability mass function) assignments. The reports for each participant are aggregated and fused by a focus-of-attention sensor fusion mediator. The demonstration system only needed to implement one instance of a focus-of-attention sensor fusion mediator, as the data streams for the four meeting-participants are processed sequentially by these three agents.

The three software agents in the demonstration experiments are: the audio sensor widget, the video sensor widget, and the focus-of-attention sensor fusion mediator. They

are hosted in three separated computers. These three computers are connected in a LAN. For the experiment to be repeatable to test various challenges to robustness, the two sensor widgets use the same data file and the simulated audio and visual sensing processes are triggered simultaneously at a preset time.

To simulate possible packet loss in data communication over a network, for each sensor widget, observation reports are not sent out only when a randomly generated number fails to exceed a specified threshold value. Furthermore, with the initial triggering being synchronized so that the "correct timing" is known for both widgets, each sent signal packet's transmission time is randomly delayed from the "correct time" within some limit, simulating network communication latency.

With the described experimental setups, many experiments with various conditions of packet loss and latency have demonstrated the feasibility of the system implementation methodology; Figure 13 shows some screen-shots of the GUI during an experiment.

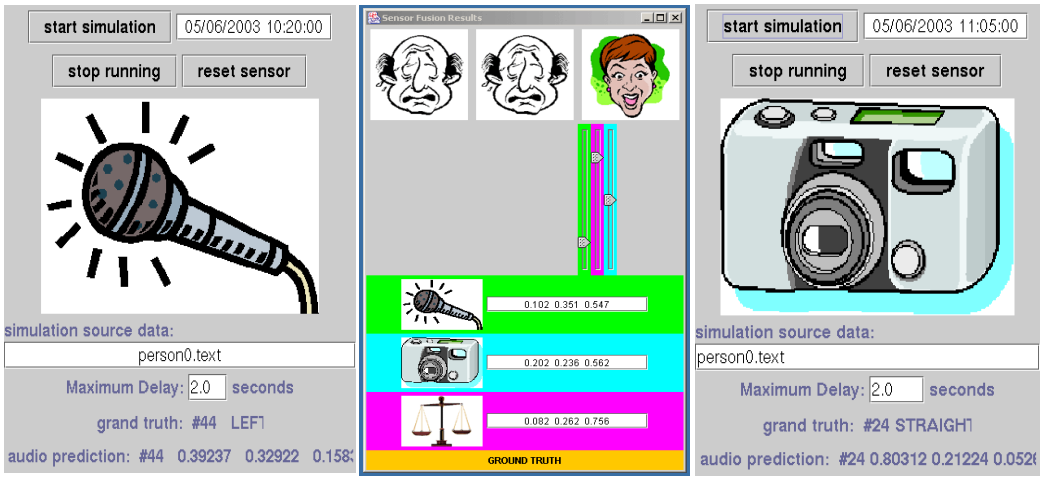

Figure 13. System architecture concept-proving demonstration experiment screen shot, using prerecorded meeting-participant's focus-of-attention data,

In Figure 13, the left and right frames are the interfaces of the audio and video sensor widget respectively, and the middle frame is the sensor fusion mediator interface. The ground truth is shown in the sensor fusion mediator interface via the face icons. As an example, the status shown in Figure 13 is that the currently monitored meeting-participant's focus-of-attention is at his/her right-side person.

The estimates and related confidences of the sensor widgets and their fused result are shown as different colored bars-tickers under the face icons. It can be seen that at this specified time, both audio and video sensors were working properly and their focus-of-attention reports had arrived at the sensor fusion mediator. Both the audio and video sensor widget concluded that this meeting-participant's focus-of-attention was on the person to his/her right-side, and the Dempster-Shafer method powered sensor fusion mediator confirmed this conclusion. Thus, the fused result was the same as both individual estimates but with higher confidence.

The confidence measurement c in these experiments is defined as the biggest probability number $p_{\max}$ over the next-biggest probability number $p_{\max-1}$ in the probability mass function m_k (from sensor S_k, or from fused result F_k).

EQ. 23

$$c = \frac{p_{\max}}{p_{\max-1}}$$

The demonstration system provides an optional context information presentation format, denoted as $\{H; c\}$, meaning the proposition or hypothesis H that has the highest probability number in basic probability mass and the confidence ratio c.

4.4. Sensor fusion effectiveness comparison

Rainer Stiefelhagen reported his original sensing meeting-participant's focus-of-attention research in [30], where he used an ad hoc weighted sum of probability method to fuse the video and audio sensor focus-of-attention outputs:

EQ. 24 $$P(\mathrm{Foc_S}) = (1-\alpha)P(\mathrm{Foc}_S \mid \mathrm{Pan}_S) + \alpha P(\mathrm{Foc}_S \mid A^t, A^{t-1}, \cdots, A^{t-n})$$

In EQ. 24, regarding a meeting-participant's focus-of-attention $\mathrm{Foc_S}$, $P(\mathrm{Foc_S} \mid \mathrm{Pan}_S)$ is the video sensor probability estimation given the observed head pan angle Pan_S, and $P(\mathrm{Foc}_S \mid A^t, A^{t-1}, \ldots, A^{t-n})$ is the audio sensor estimated probability estimation given the observed who-has-been-talking facts.

The relative audio/video weight parameter $\alpha \in [0.0, 1.0]$ in EQ. 24 was arbitrarily chosen as 0.5 by Stiefelhagen. He reported that the linearly combined estimation method consistently yielded more accurate results than what could be achieved from either single source.

Using this linearly weighted combination of audio and video probability estimates method as a baseline, several alternative sensor fusion methods were tested in this dissertation for comparison. The methods include the plain Dempster-Shafer method, the Weighted Dempster-Shafer method (subsection 3.2.2), and the dynamically weighted Dempster-Shafer (subsection 3.2.3) method. For the four sets of prerecorded focus-of-attention analysis experimental data, each of the meeting-participant's audio and video focus-of-attention estimation is fuse, and the results are shown in Table 7.

Table 7 shows the percentages that the non-fused and fused estimations have correctly predicted the meeting-participant's focus-of-attention. In the weighted Dempster-Shafer method, the total correctness rate for each meeting-participant in each experiment is used as its corresponding weight. The dynamically weighted Dempster-Shafer method uses the same weighting scheme initially but dynamically adjusts the weights using the remnance factor $\rho = 0.9$ as described in EQ. 21 and EQ. 22.

Table 7. Sensor fusion method comparison with prerecorded focus-of-attention experimental data

	person	valid frames	audio	video	linear sum	DS	weighted DS	dyna. weighted DS
Experiment Set 2	#0	1229	55.6%	70.5%	70.1%	70.0%	71.4%	74.9%
	#1	1075	61.5%	66.2%	69.8%	70.0%	69.4%	73.0%
	#2	1098	66.1%	78.3%	80.2%	80.8%	80.2%	80.9%
	#3	991	68.8%	60.3%	65.6%	66.6%	70.0%	72.1%
Experiment Set 5	#0	768	73.8%	74.4%	76.8%	77.0%	77.0%	80.1%
	#1	956	67.6%	68.5%	72.0%	72.3%	72.1%	77.0%
	#2	1006	73.2%	83.0%	84.1%	84.2%	83.9%	85.1%
	#3	929	53.3%	67.9%	75.7%	76.9%	73.2%	79.1%
Experiment Set 6	#0	799	59.1%	70.6%	71.2%	71.5%	71.0%	74.5%
	#1	751	63.3%	84.6%	85.5%	85.8%	85.2%	86.2%
	#2	827	75.4%	82.2%	83.3%	84.3%	83.4%	83.8%
	#3	851	60.8%	80.6%	81.9%	82.3%	81.7%	82.8%
Experiment Aufname 2	#0	653	73.2%	84.2%	85.0%	85.0%	84.2%	86.2%
	#1	653	57.3%	53.5%	54.2%	54.2%	54.5%	63.1%
	#2	681	72.7%	65.6%	69.5%	69.3%	70.3%	76.1%
	#6[16]	435	85.3%	75.6%	78.2%	78.4%	79.8%	83.9%
total		13702	64.6%	72.8%	75.1%	75.4%	75.4%	78.4%

For these experimental data sets, the linear combination of probabilities method has an overall fused estimation correctness rate of 75.1%. The classic Dempster-Shafer

[16] This "#6" appears as shown in the historical data set. The surmise is that his meeting might have more than four participants but only four of them appeared in the recorded period.

method produces a slightly higher estimation accuracy of 75.4%. The weighed Dempster-Shafer method shows no significant improvement in these experiments. The dynamically weighted Dempster-Shafer method shows an apparently significantly higher estimation accuracy rate of 78.4%. There is significant case-to-case (experimental run and meeting participant) variation, but there seems to be generally consistent improvement from linear probability combination method, to classical Dempster-Shafer method, weighted Dempster-Shafer method, and to dynamically weighted Dempster-Shafer method.

By carefully examining the focus-of-attention estimation data from the two sensing modalities, it is clear that the video sensor is usually (12 out of 16 cases) more accurate than the audio sensor in estimating the meeting-participant's focus-of-attention. Therefore, a better question than "how much can sensor fusion improve estimation accuracy over the best sensor alone" to ask is "what fraction of the best sensor's mistakes can sensor fusion correct", because the latter reflects "improvement" versus "possible improvement room" explicitly.

Let n_0 stand for the total number of valid cases in an experiment, n_v stand for the number of cases in which the video sensor correctly estimated focus-of-attention, and n_f stand for the number of cases in which the fused result correctly estimated the focus-of-attention. Then, out of the mistakes that the best sensor made $n_0 - n_v$, the fraction p_{impv} that the sensor fusion method corrected is:

EQ. 25

$$p_{impv} = \frac{n_f - n_v}{n_0 - n_v}$$

Figure 14 provides a graphic representation of effectiveness, according to the measure of EQ. 25, of the four sensor fusion methods.

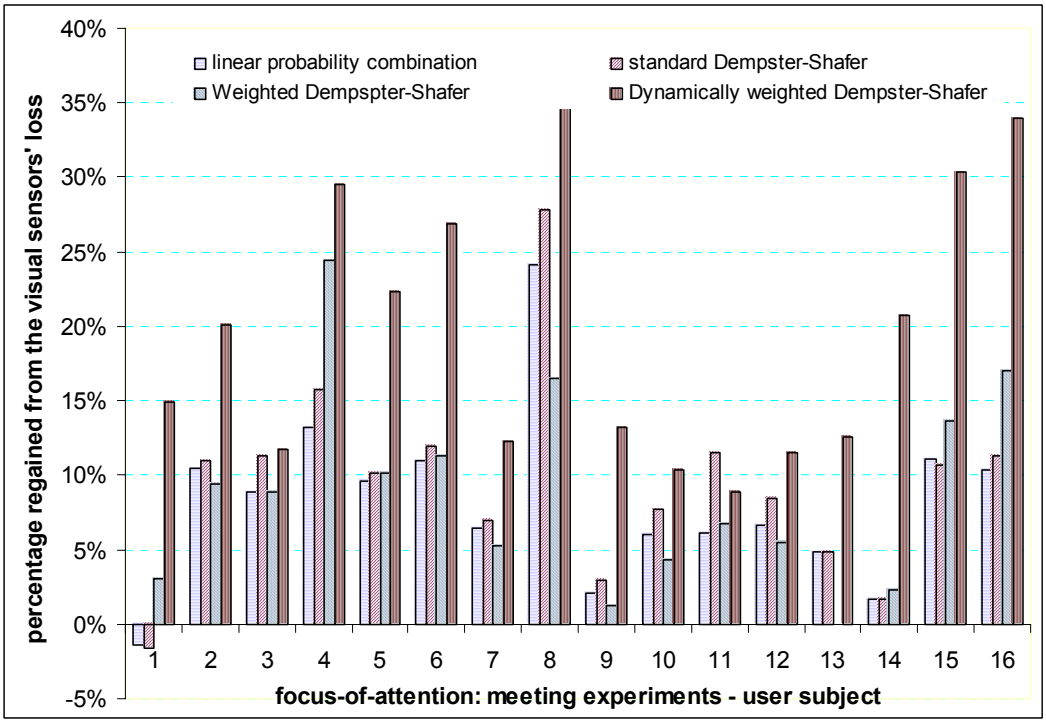

Figure 14. Sensor fusion effects in terms of correcting visual sensor misclassification

4.5. Conclusions from the experiments

4.5.1 Experiments testing methodology and system architecture

A concept-demonstration employing the described top-down methodology for building context-aware systems was implemented. Referring back to Section 3.1, there are three software architecture styles in use or under development in this research and application area: blackboard, service infrastructure, and widget style. Each style has relative advantages and disadvantages. This dissertation suggested an approach that uses the widget-style Context Toolkit system as the building blocks — called “middleware” in the software industry — to configure a system that is of service infrastructure architecture

style, but also delivers some blackboard-style advantages. The system implemented inherited all communication mechanisms from the Context Toolkit. It adds a "fusing" module that consists of a sensor fusion mediator sub-module and a sensor fusion algorithm sub-module. This implementation provides functional advantages over the original Context Toolkit system. Details and possible future extensions are described in Appendix A.2.

In the experiments conducted, the context widgets and sensor fusion mediator were separately hosted in a Java platform built on top of two operating systems, SGI IRIX 6.5 and Microsoft Windows 2000 Professional. With the sensors generating data every 3 seconds, and a simulated random network latency taking up of to 2 seconds, the focus-of-attention context-aware application system that was developed handled the data processing robustly.

Successfully fulfilling the desired context sensing and information management functionalities demonstrated that this approach is practical and useful. The system meets the challenges that sensor set is dynamic and joint probability distribution is not available. It is robust to sensor change or failure, and it is easily scalable to accommodate new sensors and context items.

4.5.2 Experiments testing sensor fusion algorithm effectiveness

In the experiments conducted, the previously published linear weighted probability combination, and the new standard Dempster-Shafer, the weighted Dempster-Shafer, and the dynamically weighted Dempster-Shafer sensor fusion algorithms were compared for fusing user's high-level focus-of-attention reports delivered by audio-based and video-based sensors.

With regard to sensor fusion effectiveness, from the data in Table 7, it is concluded that: any of the four sensor fusion algorithms achieves a noticeable improvement in correctness rate over any single sensor modality. The linear weighted probability combination method, the standard Dempster-Shafer method, the weighted Dempster-

Shafer method, and the dynamically weighted Dempster-Shafer method systematically show progressive performance improvement, but the absolute improvement from worst to best is too small to be of practical significance.

However, one thing especially worth mentioning is that, while the relative performance of the other three methods varies from one to another of the sixteen experimental data sets, the dynamically weighted Dempster-Shafer method outperforms the other three methods in all but one case. The dynamically weighted Dempster-Shafer method achieves this performance advantage by adaptively incorporating knowledge of sensors' recent success or failure. Thus, if ground truth is available, albeit after some delay, the dynamically weighted Dempster-Shafer algorithm is clearly the sensor fusion method of choice.

The overall prediction correctness rate is systematically but not significantly improved in the four progressively more sophisticated sensor fusion methods investigated, however, further comparing the fused results with their originals can reveal additional information that is interesting and useful. For example, if at some instant both audio and video context widgets estimate that a meeting-participant's focus-of-attention is on the person to his/her left, then the Dempster-Shafer fusion method seemly just confirms the proposition. But in-depth analysis reveals that the fused judgment has a much higher confidence estimate than either individual judgments, as shown in Figure 12. Similarly if the audio context widget estimates with weak confidence that the meeting-participant's focus-of-attention is on the person to his/her left, whereas the video context widget concludes with a strong confidence that the focus-of-attention is on the person to his/her right, then the Dempster-Shafer method will agree with the higher-confidence video context widget's opinion, but the fused conclusion will be with a correspondingly lower confidence.

Chapter 5.
Adaptation of Dempster-Shafer Sensor Fusion Method

The Bayesian inference network has been regarded as the classical, or even the canonical, method to deal with statistical inferences in sensor fusion applications. This chapter first describes different interpretations of Bayesian and Dempster-Shafer formalisms to give a better understanding about their similarities and their distinct interpretations. This helps to answer the question of "under what situation, is each method more appropriate for sensor fusion?" Then, using artificially generated simulation data, the performances of different methods are compared, thus, beside the theoretical discussion, the numerical results can provide a more intuitive explanation regarding how these methods work under different conditions.

5.1. Methodology and theoretical explanation

5.1.1 Objective and subjective Bayesian statistics

The term "probability" can have different interpretations[17, 18] [154]. The most commonly used, or the so-called "classical", interpretation is that, it is the ratio of the

[17] Besides the subjective and objective interpretations described in this subsection, there is also another one called "logical probability" which has been generally abandoned.

[18] Furthermore, statistics can be built based upon a few axioms, either the "subjective" or the "objective" can go with different axioms ([141]Chapter 2).

number of times that a particular outcome is observed to the total number of possible outcomes in a repeatable experiment. The "long run", or "empirical" probability, of an outcome is the limit of the proportion of times the event will occur as the number of trials in the repeatable (or conceptually repeatable) experiment increases towards infinitum [141]. This interpretation is often referred to as "frequency / frequentist" or "objective" probability.

The frequentist interpretation of probability was dominant in the 1950s ([143] p.2), perhaps because it matches the ethics that being non-emotional, impartial, and objective should be the basic scientific merit. The frequentist probability has become the standard of correctness with many scientists as their reference for experimental design and data analysis, where the techniques of hypothesis test and significance estimation etc. comprise the bulk of the so-called methods of statistical inference ([140] Chapter 5).

In contrast to the frequentist interpretation, probability is also regarded as an individual's personal degree of belief about some proposition or about any quantity unknown to the person making the probability statement. This personal probability interpretation is also referred to "subjectivist/subjective" or "epistemic" probability.

The subjectivist interpretation is now believed to be more inclusive and consistent. It does not require the process to be repeatable practically or conceptually, and this concept applies to a large number of compelling applications, e.g., as trying to answer a question like "what is the probability that Mary will attend this afternoon's seminar?" In fact, according to the proponents of subjectivist interpretation, frequentist probability is only a special case of subjectivist probability.

The Bayesian approach is free to interpret a probability as a frequency probability if one thinks it is appropriate in a given situation, but in other situations, the probability can be interpreted as a purely personal belief[19]. As described in Subsection 2.2.3.1, the Bayesian inference rule EQ. 5 provides a formal way to combine prior probability

[19] As a matter of fact, there exists a third interpretation, the so-called "objective Bayesian", which sees probability as a rational degree of belief, not as the degree of belief of an actual person ([143] Chapter 2).

distribution with new observations to produce a posterior probability regarding a statistical process. While the new observations are generally regarded as "objective" or "objectively determined" facts, the prior probability distribution is subject to different interpretation [138].

So corresponding to the subjective and objective interpretation of probability, the Bayesian inference rule can also have subjective and objective interpretation although the original interpretation (until the early part of the twentieth century) was subjective. As it has become increasingly clear that the frequentist approach to scientific inference is fraught with technical problems and inconsistencies ([140] Chapter 5), scientists schooled in the Bayesian methodology have been departing from the frequentist approach and returning to the subjectivist approach since the 1970s ([143] Chapter 1, Chapter 2).

This work takes the subjective interpretation of the Bayesian theorem, and the Dempster-Shafer reasoning system is generally regarded as a subjective approach, comparison of the two subjective methodologies would answer the question of under what situations which method is more appropriate to be used.

5.1.2 Different explanations of the Dempster-Shafer theory

The Dempster-Shafer belief functions provide a new way of using mathematical probability models to quantify subjective judgments in that probabilities for related questions are assessed and then their implications are considered ([18], [143] Chapter 7). The potential of this approach was soon recognized and many interpretations of it were proposed ([129], [130], [144]), most interpretations try to use a more generalized framework to integrate various sensor fusion techniques together, where the Dempster-Shafer method is regarded only as a special case. The relatively influential and well-developed interpretations are the random set theory ([56], [57], [58], [64], [139]), the generalized Bayesian ([143] Chapter 7), the upper and lower probability ([132], [133], [146]), and the transferable belief model ([144], [145]).

Of all the Dempster-Shafer interpretations, the random set theory is perhaps the most popular. While the underlying mathematics of random set theory is rather complex, its core concept can be summarized by the so-called “finite-set statistics”, which has its element data consisting of finite sets of ordinary observations rather than individual point or vector observations ([64] page 7). The probability density function in finite-set statistics describes the comprehensive statistical behavior of an entire sensor suite, thus sensor data fusion algorithms may be interpreted as statistical estimators of a finite-set parameter.

Within the context of random set theory, the Dempster-Shafer theory of evidence can be formally expressed (the probability measure is no longer additive because one subset may include another subset), and the Dempster-Shafer evidence combination rule would correspond to finding the probability characteristics of the intersection of non-empty independent random sets. Referring back to EQ. 10 in Subsection 2.2.3.2, the numerator denotes the intersection of non-empty independent random sets, being counted together, whose summation represents the cumulative degree to which the two bodies of evidence do not contradict each other.

When all masses occur on singleton subsets, then the belief function is an additive measure and the evidence combination formula is equivalent to the Bayesian inference formula with conditional independence. From this perspective, the Dempster-Shafer theory is also regarded as a generalization of the Bayesian formalism. To more formally state it, Shafer points out that the Bayesian formalism has two elements, the idea of a probability and the rule of conditioning, and both these elements have their place in the belief-function formalism ([63] Section 7.1). A Bayesian probability measure is a special kind of belief function in the Dempster-Shafer system, and conditioning a belief function on a subset of its frame is equivalent to combining it with a special belief function that represents the knowledge that the subset is true.

At its very beginning stage, the Dempster-Shafer theory was developed as a theory of lower and upper probabilities by Dempster in his effort to reconcile Bayesian statistics

with Fisher's fiducial argument[20] ([130] Preface). Roughly speaking, lower and upper probability is used to denote the low and high boundary respectively in a statistical model such that the situations where the probability of events itself is uncertain can be handled. The Dempster-Shafer theory can be regarded as a special case of lower and upper probability system in that the belief and plausibility are the lower and upper probability respectively[21].

There are variants of Dempster-Shafer theory interpretations too. For example, the transferable belief model is very different from other interpretations in that it is a two-level model [144], which is closer to Shafer's original interpretation in his book [18]. The "credal" level is intended to model subjective personal beliefs and it is completely dissociated from any probability functions, whereas the "pignistic" level derives probability distributions from the credal state to support making reasonable and consistent decisions. These do not now seem to provide any additional utility for solving sensor fusion problems addressed in this dissertation.

Many interpretations of Dempster-Shafer theory may conceptually add complexity, but they actually do not necessarily add any technical difficulty. In reality, from the practical utilization perspective, they only beneficially add flexibility. The random set theory interpretation is slightly favored in this research work, although other interpretations are not rejected provided the interpretations help to realize consistently and rationally behaved sensor fusion algorithm implementation.

---

[20] "Roughly, Fisher's methods and outlook were non-Bayesian (in particular, 'frequentist') … . Fisher's infamous 'fiducial argument' … was Fisher's attempt to provide a 'frequentist' alternative to Bayesian account of inverse probability." Book review by Branden Fitelson (http://www.fitelson.org/howie.pdf). More information can be found in the University of Adelaide Library's Collected Papers of R.A. Fisher, Relating to Statistical Mathematical Theory and Applications, http://www.library.adelaide.edu.au/digitised/fihser/stat_math.html.

[21] Shafer's book ([18] Preface) "offers a reinterpretation of Dempster's work, a reinterpretation that identifies his lower probabilities as epistemic probabilities or degrees of belief, takes the rule for combining such degrees of belief as fundamental, and abandons the idea that they arise as lower bounds over classes of Bayesian probabilities."

5.1.3 Where is Dempster-Shafer method more suitable?

From the discussions thus far, it is clear that both the Bayesian inference method and the Dempster-Shafer method are based on mathematical probability, but both sides are able to accept subjective judgments as probabilities. The Dutch book argument has been used in history for conservatively checking the consistency of human probability judgments and for proving the rationality and justification of the Bayesian method[22] ([137], [142]), and it has been proved that the Dutch book argument cannot be used to criticize the Dempster-Shafer method [145]. Therefore, "We believe both formalisms (Bayesian and Dempster-Shafer) are useful. Their usefulness in a particular problem depends on the nature of the problem and the skill of the user" ([143] page 482).

Table 8 lists the situations where the Bayesian method or the Dempster-Shafer method is more suitable to be used as the sensor fusion algorithm. More detailed explanations follow.

In the Bayesian framework as well as in classical statistics, all events or hypotheses are assumed mutually exclusive, meaning that only one hypothesis can be true at any given time. To the contrary, the Dempster-Shafer framework is suitable to apply to situations where information pieces of different granularity need to be considered at the same time. For example, if a face recognition system has to deal with information, like {"male", "female", "Tom" ("Tom" $\subset$ "male"), "John" ("John" $\subset$ "male"), "Mary" ("Mary" $\subset$ "female"), etc.}, simultaneously, then using the Dempster-Shafer formalism may be a better choice. As a more specific example, in an image-based people-tracking system, after two already identified people approach closely and separated gain, the tracking system may have lost the exact identity of each moving person but still has

[22] The basic idea is briefly explained here. Suppose a bookie sets odds on all subsets of a set and accepts bets in any amount on any combination of the subsets. Unless the odds are computed from a prior probability and updated according to the Bayesian inference rule, a Dutch book can be made such that anyone can refer to it to make a series of bets against the bookie to win the game no matter what may come out.

vague information as “this is either person-A or person-B”. In such situations, the Dempster-Shafer method is more suitable to be used.

Table 8. Situations where Bayesian or Dempster-Shafer method is more suitable for sensor fusion

Bayesian method preferred	Dempster-Shafer method preferred
All hypotheses are mutually exclusive	Information pieces of different granularity are included, some hypotheses may include others
Prior probabilities of all hypotheses are known, and new observations unambiguously related to probability	Prior probability distribution is unknown, and/or new observations partially correlate to probability distribution, ignorance needs to be counted
Joint probability distribution is known, or observations are conditionally independent	Joint probability is not known, observations are independent
Direct correlation with probability helps maximizing expected utility	Difficult to correlate evidence with probability distribution, thus weak in decision-making support

Corresponding to the fact that all hypotheses in a Bayesian framework are mutually exclusive, assigning a probability number p to one hypothesis implies that $(1 - p)$ is

assigned to this hypothesis's complement. Therefore, the Bayesian inference system is suitable to be used in situations where all hypotheses' prior probabilities are known, and new observations would directly contribute to probability distribution updating. To the contrary, in the Dempster-Shafer formalism, the concepts of negation and ignorance are clearly separated, meaning that when probability p is assigned to a hypothesis it does not imply anything regarding how the remaining $(1 - p)$ is assigned. Therefore, the Dempster-Shafer method is especially suitable to be used in situations where hypotheses' prior probability distribution is not available, observations cannot directly relate to complete probability assignment, and partial ignorance regarding the hypothesis probability distribution has to be counted. As an example, if an application needs to describe information such as "it is very likely that (e.g. with a confidence of 0.8) this person is **not** Tom but no further information regarding this person's identity is available at this time", then the Dempster-Shafer method may be more suitable to be used.

As described in subsection 2.2.3.1, when observations from multiple sensors are used to evaluate the probability of a hypothesis using EQ. 5, the symbol E represents the joint observation over all sensors. This means that the Bayesian method should be used in situations where both the hypotheses' prior probability distribution and the sensors' joint distribution are available. This practically implies the situations that the being observed process is stable and the sensors' set configuration is stable. In such cases, the Bayesian method can maximally use all available information. However, for situations where the available sensor set's configuration is highly dynamic, so the joint probability distribution of the sensor observations is unavailable, then probably the Dempster-Shafer method is more suitable to be used for sensor fusion.

The reason that the Bayesian inference method has been widely used in numerous scientific and social applications' data analysis is that it directly connects mathematical probability concepts with human subjective judgments. Thus, it can be very easily incorporated into utility functions to support making judicious decisions. For the Dempster-Shafer framework to be used to solve reasoning problems and to help make a decision, a deep understanding of the application situations and special skills are often

required to correctly correlate the observed evidence and the hypothesis probability distribution of interest. This is perhaps the biggest hurdle to overcome for the Dempster-Shafer theory to be widely used practically.

Besides the properties illustrated in Table 8, in contrast with the classical Bayesian inference framework, for its practical usage, the Dempster-Shafer method has a very good property that should be noted — its computational implementation is very simple and robust compared to many other sensor fusion methods. In a data fusion process, the initial probability mass function (from the first arriving reports) data are kept in memory first. Referring to EQ. 10, for the arriving probability mass function m_i from sensor S_i, all the evidence combination rule does is to calculate the numerical products for each possible proposition (intersection of the newly arriving and the original probability mass function in memory) and then to normalize the results. Since the probability mass function data in memory are always valid and can be updated with newly arriving sensory data at any time, obviously, this sensor fusion process is very robust to sensor configuration change. If the given sensor set is predefined, this additive property ensures that the final sensor fusion result will not be affected by the sensory data's arrival order.

5.2. Experiments with artificially generated data

Analyzing the prerecorded experimental data (Section 4.4) provides a good feeling for how the sensor fusion methods work. Given the theoretical comparison in the last section, testing the sensor fusion algorithms against data with known probability distribution would help making the conclusions more clear and convincing.

5.2.1 Design of simulated experiments

For the comparison of sensor fusion algorithms to be objective and convincing, the experiments should test a typical sensor fusion procedure that can represent a large range of data synthesizing situations. Without losing generality, a scenario of focus-of-attention

estimation with three head-pan-angle detection sensors is simulated in the imagined experimental settings.

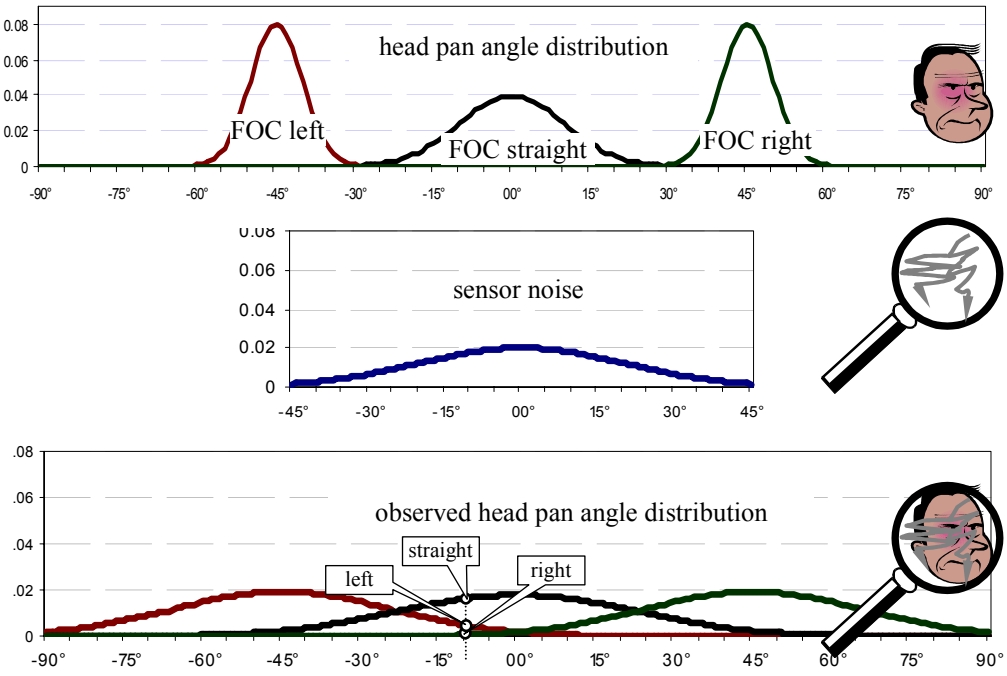

Figure 15. Simulation of a focus-of-attention estimation scenario

Suppose at time *t*, the meeting-participant's head pans an angle $\theta(t)$ complying with one of the Gaussian distribution functions $N[-45°, \sigma_0]$, $N[0°, \sigma_{00}]$, $N[45°, \sigma_0]$:

EQ. 26 $\theta_L \sim N[-45°, \sigma_0]$

EQ. 27 $\theta_S \sim N[0°, \sigma_{00}]$

EQ. 28 $\theta_R \sim N[45°, \sigma_0]$

if his/her focus-of-attention is on the left-side, the straight-forward, or the right-side meeting-participant respectively.

Because of sensor S_i's measurement noise at time t: $n_i(t) \sim N_i(t)$, which has a Gaussian distribution $N[dft_i(t), \sigma_i]$ ($dft_i(t)$ is the drifting effects in measurement) independent of the angle being measured

EQ. 29 $$n_i(t) \sim N[dft_i(t), \sigma_i],$$

the sensor S_i observed pan angle will be

EQ. 30 $$\theta_i(t) = \theta(t) + n_i(t).$$

Because the observed head pan angle is the "true" head pan angle plus measurement noise, sensor S_i observation complies with the combined probability distribution function as:

EQ. 31 $$\theta_{Li}(t) \sim N[-45° + dft_i(t), \sqrt{\sigma_i^2 + \sigma_0^2}]$$

EQ. 32 $$\theta_{Si}(t) \sim N[dft_i(t), \sqrt{\sigma_i^2 + \sigma_{00}^2}]$$

and

EQ. 33 $$\theta_{Ri}(t) \sim N[45° + dft_i(t), \sqrt{\sigma_i^2 + \sigma_0^2}]$$

when the meeting-participant's focus-of-attention is on the left-side, the straight-forward, or the right-side person respectively.

5.2.2 Simulation data and data processing

In this simulated focus-of-attention sensing scenario, assuming a meeting-participant's focus-of-attention has a probability[23] of 0.3, 0.4, and 0.3 on the left-side, the one straight-across the table, and the right-side person respectively, the "ground truth" can be generated randomly, and the result is recorded as a series of "left", "straight", or "right" data. To simulate the focus-of-attention changing over time, each occurrence of the ground truth information is then repeated 10 times per second for a random-length (in the range of 5 to 15 seconds) duration, this way, the ground truth time series is generated.

Given the ground truth defined in the time series, complying with the probability distribution function EQ. 26, EQ. 27, and EQ. 28 with $\sigma_0 = 5°$ and $\sigma_{00} = 10°$, the meeting-participant's head pan angle θ is generated for the time instance.

For every generated instantaneous head pan angle $\theta(t)$, sensor S_i will have its observed angle according to EQ. 30. For two typical sensor fusion situations, where (I) the sensors are of the same precision and (II) the sensors are of conspicuously different precision, with different relative drifting $dft_i(t)$ properties, each sensor S_i observed pan angle will be generated as $x_i(t)$.

The sensor S_i observed head pan angle $x_i(t)$ complies with one of the probability distribution functions EQ. 31, EQ. 32, or EQ. 33 corresponding to the situation that the meeting-participant's focus-of-attention is on the left, straight, or right person respectively. But because the drifting effect in measurement $dft_i(t)$ cannot be easily estimated, the sensor S_i would reasonably estimate the focus-of-attention as if there were no drift. Therefore, given the head pan angle is observed as $x_i(t)$ (depicted by x for brevity), the sensor S_i's rational focus-of-attention estimation can be calculated based on the relative strength of its assumed probability density function $\{f_{iL}(x), f_{iS}(x), f_{iR}(x)\}$ as illustrated at the bottom part of Figure 15:

[23] This is the prior probability $\{p_{L0}(t), p_{S0}(t), p_{R0}(t)\}$ of the focus-of-attention event; to get a fair comparison, suppose this information is not known to any sensor fusion method.

EQ. 34

$$p_{iL}(x) = \frac{f_{iL}(x)}{f_{iL}(x) + f_{iS}(x) + f_{iR}(x)}$$
$$p_{iS}(x) = \frac{f_{iS}(x_i)}{f_{iL}(x) + f_{iS}(x) + f_{iR}(x)}$$
$$p_{iR}(x) = \frac{f_{iR}(x_i)}{f_{iL}(x) + f_{iS}(x) + f_{iR}(x)}$$

where the three numbers $\{p_{iL}, p_{iS}, p_{iR}\}$ correspond to the sensor S_i assigned probabilities that the focus-of-attention is on the left, straight, or the right person.

Since the three sensors' observations are conditionally independent, their joint probability distribution equals to the product of all three probability distribution functions. Using $\{p_{0L}, p_{0S}, p_{0R}\}$ to denote the ground truth probability distribution, then Bayesian inference method can be used to calculate the posterior probability given the three sensors have observed $x_1(t)$, $x_2(t)$, and $x_3(t)$ respectively. Suppose the prior probability distribution is not known, then according to conventions in Bayesian inference practicing, an even distribution will be assumed:

EQ. 35

$$p_{0L} = p_{0S} = p_{0R} = p_0 = \frac{1}{3}$$

Under these above described conditions, omitting the derivative process, the Bayesian inference method would generate sensor fusion results as:

EQ. 36

$$p(L \mid x_1, x_2, x_3) = \frac{1}{p_0^{\,2}} p_{1L}(x_1) p_{2L}(x_2) p_{3L}(x_3)$$
$$p(S \mid x_1, x_2, x_3) = \frac{1}{p_0^{\,2}} p_{1S}(x_1) p_{2S}(x_2) p_{3S}(x_3)$$
$$p(R \mid x_1, x_2, x_3) = \frac{1}{p_0^{\,2}} p_{1R}(x_1) p_{2R}(x_2) p_{3R}(x_3)$$

By examining the Bayesian inference formula EQ. 36 carefully, one will notice that it is directly proportional to the Dempster-Shafer evidence combination rule, and after taking account of the normalization process, they will produce exactly the same results.

This comparison result can be expected because this is a special situation that the observations of all sensors are conditionally independent and their reported estimations are singletons, i.e., the meeting-participant's focus-of-attention can be on only one of the three persons.

Because the Bayesian inference method and the Dempster-Shafer evidence combination rule are going to produce exactly the same results, only the Dempster-Shafer method calculations are numerically carried out in the following comparisons.

5.2.3 Experiments and their result analysis

Two typical sensor fusion situations regarding a sensor's precision variation are simulated in the following numerical experiment sets: (I) the sensors are of the same precision and (II) the sensors are conspicuously of different precision.

5.2.3.1 Case I: sensors are approximately of the same precision

$$(\sigma_1 = \sigma_2 = \sigma_3 = 20°)$$

The experimental duration is chosen for about 60 minutes (the sensory observation time series has a total of about 36,000 data sets, as each of the three sensors will generate 10 reports per second), since measurement drift is the most difficult part to be properly handled in real practices, the following typical sensor measurement drift effects are experimented:

I. the sensor's drift cycles are relatively long compared with the experiment duration:
   dft1(t) = 5°·sin(0.001·t), (drift cycle: ~105 minutes);
   dft2(t) = 5°·sin(0.0007·t), (drift cycle: ~150 minutes); and
   dft3(t) = 5°·sin(0.0003·t), (drift cycle: ~345 minutes);

II. The drift cycles are relatively short:
   dft1(t) = 5°·sin(0.01·t), (drift cycle ~10.5 minutes);

dft2(t) = 5°·sin(0.007·t), (drift cycle ~15 minutes); and
dft3(t) = 5°·sin(0.003·t), (drift cycle ~35 minutes);

III. The sensors' drift amplitudes are relative large compared with their built-in measurement noise:
dft1(t) = 10°·sin(0.01·t);
dft2(t) = 5°·sin(0.007·t); and
dft3(t) = 15°·sin(0.003·t).

With the above-specified sensor drift situations, two random sets of experimental data are generated for each situation, and four sensor fusion methods are applied to process the data. The results are shown in Table 9.

Table 9. Comparison of sensor fusion algorithm effectiveness using simulated sensory data (sensor noise $\sigma_1 = \sigma_2 = \sigma_3 = 20°$)

sensor fusion		Drift I		Drift II		Drift III	
		#1	#2	#1	#2	#1	#2
parameters	sensor S_1 only	76.8%	77.2%	75.8%	75.4%	77.5%	72.8%
	sensor S_2 only	71.9%	71.9%	70.4%	70.3%	72.1%	68.6%
	sensor S_3 only	72.2%	71.6%	70.1%	70.5%	70.0%	68.0%
results	linear sum	78.9%	79.1%	77.6%	77.8%	79.0%	75.4%
	DS	80.3%	80.6%	78.9%	79.4%	80.3%	76.9%
	weighted DS	80.4%	80.8%	79.0%	79.4%	80.7%	76.8%
	dyna. Weighted DS	80.5%	80.6%	79.4%	79.6%	82.1%	78.2%

The numbers in percentage format in Table 9 are the fractions of the events in which the meeting-participant's focus of attention is correctly estimated, the columns denote the simulated sensors' drift property and experiment data sets, and the rows denote individual sensors' (Sensor S_1, S_2, and S_3) performance and the effectiveness of sensor fusion methods. The "linear sum" means the probability linear combination with $\alpha = 0.5$; the "DS" means standard Dempster-Shafer method; the "weighted DS" means weighted Dempster-Shafer method with the weights derived from sensors' overall estimation correctness ratio; and the "dynamically weighted DS" means Dempster-Shafer method with adaptively dynamic weighting schemes with remnance factor set to 0.9.

The experimental results in Table 9 largely confirm the conclusions derived from those experiments with prerecorded data as described in the Chapter 5. Firstly, for the four sensor fusion algorithms, with the parameters being set to typical values (linear combination relative weight $\alpha = 0.5$ and the remnance factor $\rho = 0.9$ in dynamically weighted Dempster-Shafer method), there is not much difference regarding the effectiveness of estimation correctness rate. Secondly, roughly speaking, the linear combination method, the standard Dempster-Shafer method (which has the same result from Bayesian method), the weighted Dempster-Shafer method, and the dynamically weighted Dempster-Shafer method progressively have marginally better performances.

Because the sensor S_1 statistically outperforms the other two sensors in focus-of-attention estimation accuracy and reliability, it would be intuition-building to examine at what percentage each of the four sensor fusion methods could statistically compensate for its errors.

Let n_0 stand for the total number of valid cases in an experiment, n_1 stand for the number of cases that the best sensor S_1 has correctly estimated focus-of-attention, and n_f stand for the number of cases that the fused result has correctly estimated the focus-of-attention. Then, out of the mistakes that the best sensor made, statistically, the percentage p_{impv} that the sensor fusion method can correct is:

EQ. 37

$$p_{impv} = \frac{n_f - n_1}{n_0 - n_1}$$

The measure of improvement p_{impv} afforded by the sensor fusion method statistically out of the mistakes made by the best sensor S_1 is illustrated in Figure 16.

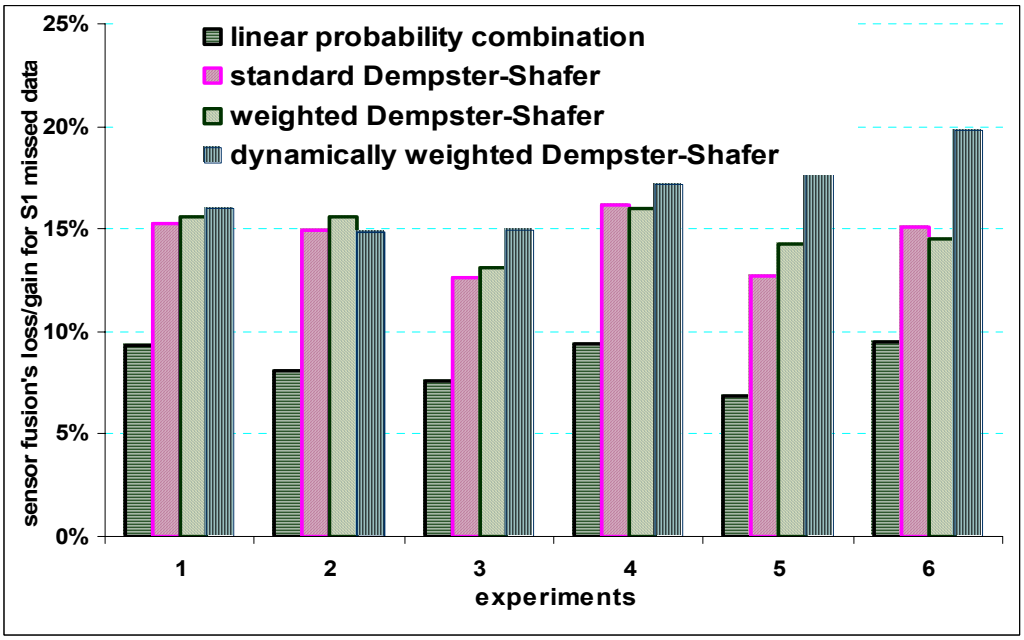

Figure 16. Sensor fusion effects in simulation with sensors being of the same precision but having different drift effects

5.2.3.2 Case II: sensors are conspicuously of different precision

($\sigma_1 = 5°$, $\sigma_2 = 10°$, and $\sigma_3 = 20°$)

Table 10 shows the experimental results with the sensors having the same drift properties as the previous subsection (Subsection 5.2.3.1) described but having conspicuously different precision characteristics.

If one sensor is consistently doing better (of higher precision, more accurate, more reliable) than any other sensors in the system, and there is no way for their lower-level sensory data to be merged, then the traditional sensor fusion method (the Bayesian method) can not improve on the result from the best sensor in a statistical sense. This situation is illustrated in Table 10, where with a large number of repeated experiments, because sensor S_1 statistically produces more accurate predictions, and no other information clue is available to infer when it may fail, using traditional sensor fusion method to combine sensors' outcomes cannot generate results that are better than those from the best sensor.

Table 10. Comparison of sensor fusion algorithm effectiveness using simulated sensory data (sensor noise: $\sigma_1 = 5°, \sigma_2 = 10°$, and $\sigma_3 = 20°$)

sensor fusion		Drift I		Drift II		Drift III	
		#1	#2	#1	#2	#1	#2
parameters	sensor S_1 only	85.7%	87.0%	86.7%	85.0%	83.7%	82.5%
	sensor S_2 only	81.4%	82.3%	82.4%	81.1%	80.1%	77.0%
	sensor S_3 only	71.9%	72.7%	72.1%	70.7%	70.5%	69.2%
results	linear sum	84.8%	86.3%	85.9%	84.3%	84.3%	81.6%
	DS	84.6%	86.1%	85.7%	84.1%	84.4%	80.9%
	weighted DS	84.9%	86.4%	86.0%	84.5%	84.6%	81.6%
	dyna. Weighted DS	86.0%	87.3%	87.4%	85.9%	87.7%	84.8%

Intuitively, a wise human gambler would rationally bet in a way that heavily relies on the best sensor's observation reports, meanwhile seeking clues regarding which sensors should be trusted when, to make judgments about confidence. The beneficial property of

the Dempster-Shafer method is that its behavior resembles a rational human agent's reasoning process, so when the ground truth is available shortly afterwards, using the dynamically weighted Dempster-Shafer sensor fusion method produces the best sensor fusion results, as shown in Table 10.

As in the sensor set parameter situation Case I in previous sub section, the sensor S_1 statistically outperforms the other two sensors in more accurately and reliably estimating the meeting-participant's focus-of-attention. The intuitive statistical improvement p_{impv} is ratio of (correct result fraction obtained using sensor fusion – correct result fraction obtained from the best sensor alone) to (1 – correct result fraction obtained from the best sensor alone). The improvement p_{impv} can be also calculated using formula EQ. 37, its graphic representation is shown in Figure 17.

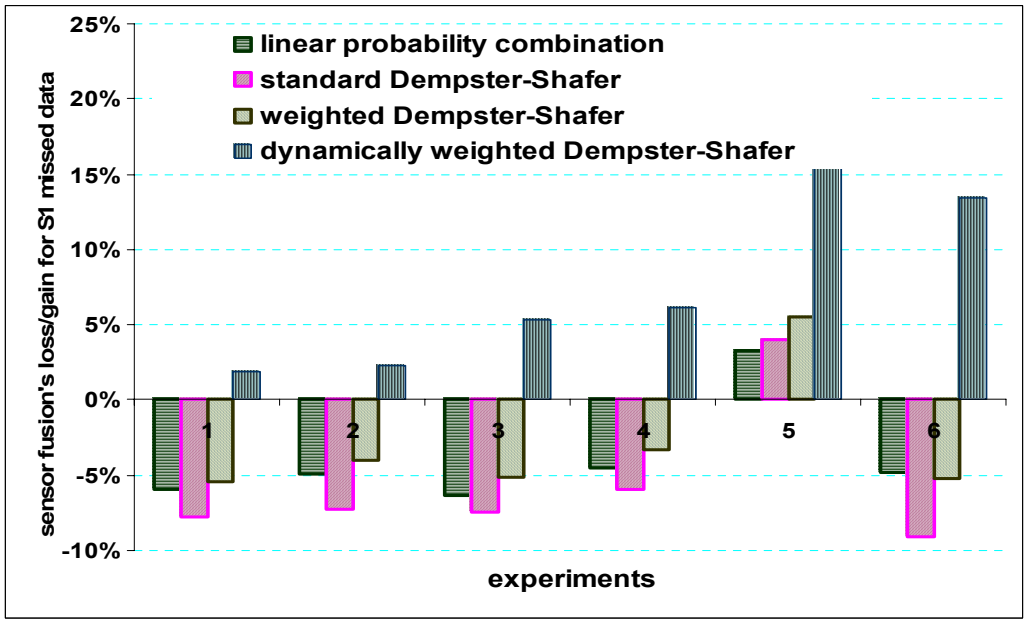

Figure 17. Sensor fusion effects in simulation with sensors being of significant different precision with different drift effects

The negative values in Figure 17 for the linear probability combination method, the classical Dempster-Shafer method, and the weighted Dempster-Shafer method indicate that they yield in all but one case worse accuracy than the best sensor alone, whereas the dynamically weighted Dempster-Shafer method always yields an improvement.

In this case (as well as in case I as described in previous subsection), the Bayesian inference method would produce exactly the same results as the classic Dempster-Shafer sensor fusion method, so its effectiveness is not separately shown from the standard Dempster-Shafer method.

The graphical presentation makes it clearer that, when the joint distribution is not available, and especially when one sensor is much more accurate and reliable than any other, adding a few poor-quality sensors and using linear probability combination, standard Dempster-Shafer, and weighted Dempster-Shafer sensor fusion methods cannot statistically improve overall system performances. The performance can however be improved by using the dynamically weighted Dempster-Shafer method, because this method incorporates useful new information — obtained from comparing with the available ground truth — regarding the temporal evolution of the sensors characteristics.

Chapter 6.

Conclusion and Future Work

6.1. Methodology and Implementation Summary

6.1.1 Context sensing

Traditional sensor fusion in measurement and control systems deals with how to evaluate targeted specific parameters more accurately and reliably. The most cost-effective way to improve such parameter estimation accuracy or reliability is via adding measurement redundancy. There are two ways to make redundant measurements: taking multiple measurements with one sensor or using multiple sensors to measure the same parameters[24]. Therefore, sensor fusion can be realized either along the time dimension to fuse multiple measurements from the same sensor or to fuse measurements from multiple sensors. In practice, sensor fusion is often implemented simultaneously in both dimensions. Typical practices are to filter out noise along the time dimension, and then only the "cleaned" data over multiple sensors are processed [15]. The most commonly used sensor fusion technology is the statistically weighted averaging with the smaller standard deviation measurements receiving proportionally heavier weights.

Sensor fusion for context-aware computing challenges the sensor fusion domain with two difficulties. First, in applications the sensor set's configuration and the sensors'

[24] Some researchers would call fusion of multiple measurement made by the same sensor "sensor characterization", restricting the term "sensor fusion" to the combining of multiple measurements of the same variable by different sensors.

working environment are typically changing constantly, so the sensors' joint observation distribution is typically unavailable. Second, because "context" includes high-level conclusions versus straightforward measurements, the sensor fusion process is no longer just traditional statistically weighted averaging to improve the estimate of some parameters. Instead, many artificial intelligence-based inference algorithms are used to combine the conclusions of different sensors. Furthermore, the conclusions to be fused are often at different levels of abstraction [147].

This dissertation advocates a top-down methodology to meet these challenges in two steps.

First is to specify, for given application requirements, the necessary context information and possible ways to implement the sensing. To actually implement this may require a spiral process involving many forward and backward analyses. The result is an information architecture scheme that specifies all context information pieces' formats and the possible values each information piece can take.

Second, all sensors' outputs are wrapped up by their corresponding software agents, called "Context Widgets", so that only the information pieces defined by the information architecture are reported. These information pieces are collected by other software agent called the "Sensor Fusion Mediators", where the information pieces regarding the same objects or events are combined or consolidated.

The term "sensor fusion" is generally referred to in a broader sense as the process that intelligently fulfills the mapping from multiple raw sensory data sets into some structured information, usually of higher level of abstraction. Sensor fusion techniques can be roughly classified into "cooperative", "competitive", and "complementary". This dissertation promotes a sensing methodology that automatically enables the competitive type of sensor fusion as illustrated in Figure 18.

Usually multiple measurements are fused so that a context Widget reports its observation to a corresponding Sensor Fusion Mediator at a lower update rate and at a higher level of semantic abstraction. At the higher level, observation reports regarding the

same object or event gathered from all the context Widgets are fused by the Sensor Fusion Mediators using an evidence combination algorithm, for example, the Dempster-Shafer Theory of Evidence algorithm.

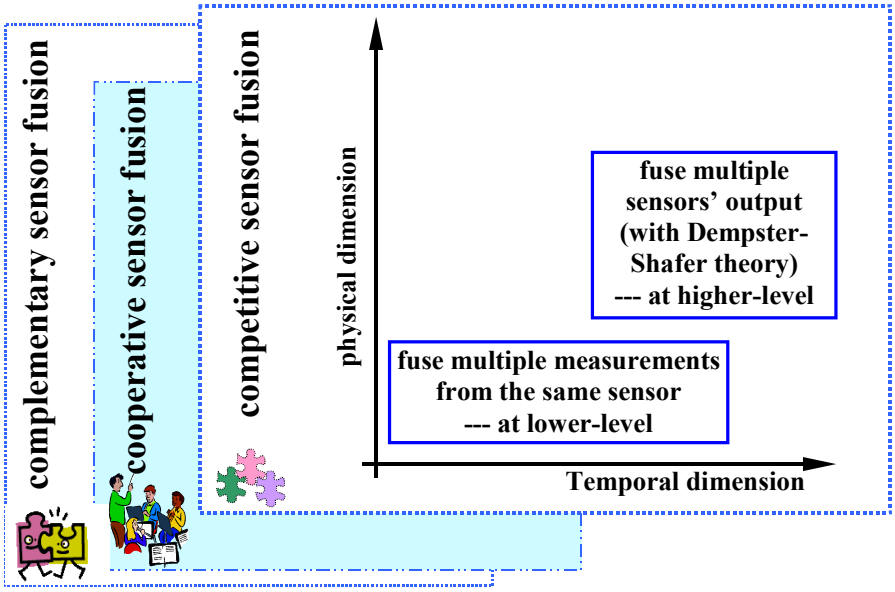

Figure 18. Sensor fusion techniques applicable to context-sensing

6.1.2 Context sensing implementation

The key to provide a generalizable solution of sensor fusion support in context-aware computing systems is to realize the idea of layered architecture in the sense of information abstraction. Using modularized system components is the only practical way to realize the system architecture, given the complexity of real applications.

This dissertation advocates a context information architecture concept that defines "context" as an ensemble of discrete information pieces regarding human computer-users, their events, related objects or facilities. The context information architecture mainly comprises two kinds of semantic entities: the "stage" entity describes environmental

properties (facility functionality, available equipment, current states, etc.) and the "user" entity describes specific users and user-related properties (personal information, preferences, activities, etc.). When the information architecture is further defined in detail using the two entity models as references, a context database can then be built around the targeted users' frequently used facilities (the "stages") using tools of any standard SQL database management system.

In the thus-defined context information architecture — implemented in context database format — a user-group's activity or event is indexed by "stage-ID", "time" and "group-activity-ID" as its joint key, and a specific user's activity or event is indexed by "user-ID", "time", and "activity-ID" as its joint key if one needs to query the relational database system.

This context sensing methodology does not specify how the context database should be implemented; rather it only emphasizes that the context information collection and distribution should be insulated from the concerns regarding how the information is sensed. The database is connected to the context-aware computing system via JDBC interfaces in the dissertation concept-demonstration system, which is built on top of Java platforms to ensure compatibility for future development.

The concept demonstration system is built using the Georgia Tech Context Toolkit building blocks. The Context Toolkit system usage is modified in this dissertation to facilitate sensor fusion and further promote the separation of context information usage from the concerns of context sensing technical implementation. The key modification is the introduction of Sensor Fusion Mediator components, which are special kinds of "Context Aggregators" in the original Context Toolkit systems, but with newly-added functionalities for sensor management and realization of sensor fusion algorithms.

What sensor fusion algorithms could be used in the sensor fusion mediator largely depends on the input information characteristics, and the pursued modular design scheme has made it easy to incorporate new sensor fusion methods. For the demonstration system,

the Dempster-Shafer Theory of Evidence algorithm is chosen and implemented as the main sensor fusion engine.

Choosing Dempster-Shafer theory as the primary sensor fusion algorithm is based on a careful analysis of the special challenges faced in context sensing tasks in typical context-aware computing systems. The choice is the result of comparison of all identified commonly used sensor fusion methods. The standard Dempster-Shafer method is further improved in this dissertation via the novel idea of using dynamically adjusted weights to adaptively incorporate knowledge of the sensor's recent performance.

Theoretical analysis and experiments with prerecorded data and artificially generated data have proved the feasibility of the proposed top-down methodology and the effectiveness of the extended Dempster-Shafer sensor fusion method.

6.2. Dissertation contributions

This dissertation addresses context-sensing challenges in context-aware computing. Specifically, a *generalizable* sensor fusion solution is suggested, and a new sensor fusion algorithm is implemented within its framework. The framework advocates a top-down methodology, and promises to greatly reduce the sensor fusion implementation difficulties. The algorithm is an extension of the Dempster-Shafer Theory of Evidence method. Introducing the Dempster-Shafer theory into, and then further extending it for, context-aware computing are the key contributions of this dissertation.

The Dempster-Shafer theory is very successfully used in multiple source data fusion (MSDF) military applications because of its ability to compute with human subjectivity as well as mathematical probability estimates from the sensors, and to easily combine the two groups of information to obtain results consistent with human intuition. It is believed that this property is especially relevant to the context-aware computing domain, where blending "objective" observations, e.g., from sensors, with "subjective" human estimates, e.g., opinions reflecting human feelings, is also desirable. Based on this observation, the Dempster-Shafer reasoning framework is promising to succeed in the sensor fusion for

context-aware computing domain. Table 11 shows the highlights of an applicability comparison of the extended Dempster-Shafer method versus the classical Bayesian method for sensor fusion in context-aware computing.

Table 11. Comparison of sensor fusion options for context-aware computing

Requirement	Sensor fusion method		Applicability to context-aware HCI applications
	Classic (Bayesian)	Extended Dempster-Shafer	
Modeling uncertainty	Hypotheses are mutually exclusive; subjectivity-probability relationship can be easily established; method variations are well understood and widely used	Hypotheses can be nested; human-like ambiguity and partial ignorance can be easily included and processed; but it is difficult to convert evidence to belief mass function	Dempster-Shafer framework is closer to human intuition, so it can easily combine objective observations with human opinions
Managing sensors suite and drift	It is suitable for static sensor set configuration	It can easily realize differential trust scheme; it is adaptive to dynamic sensor configuration	Context-sensing required to work in dynamic situations
Benefiting from sensor fusion	Joint probability distribution can be used to achieve better estimation for non-independent observations	Only independent observations can be accommodated by current formulation[25]	Joint probability distribution information is usually not available

Using the Dempster-Shafer Theory of Evidence as the primary uncertainty management framework for context-aware computing, the bottom line is that, when all the hypotheses (objective evidence, or subjective propositions) are independent and

[25] It is unknown whether this apparent inability of the superset (the Dempster-Shafer framework) to include the subset (the classical Bayesian framework) demonstrates only the incomplete state of the present formulation or, as some – including sometimes Shafer himself – claim, it demonstrates the classic Bayesian framework does not always emerge as simply a subset of the Dempster-Shafer framework.

mutually exclusive, the Dempster-Shafer method is identical to the Bayesian method, thus it performs as well as the Bayesian-based classical sensor fusion methods. In addition, this suggested solution is expected to gain the following advantages for context-aware computing:

- The reasoning framework works on "evidence", regardless of whether its source is "objective" or "subjective". The hypothesis estimation is expressed as a range between "belief" and "plausibility", thus it is easy to combine human opinions with sensors' observations
- Context-aware computing applications can easily incorporate ambiguous information (e.g., either "A" or "B") and freely admit partial ignorance; thus the human-intuitive-like reasoning process appears to be more suitable than crisper logical schemes for context-aware computing
- It can reason based on all pieces of evidence as an ensemble; this includes nested hypotheses, which cannot be easily handled by the classical Bayesian method

If the ground truth becomes available with some delay, which is often the case in context-aware computing applications, the dynamically weighted Dempster-Shafer method developed in this dissertation can use this information to improve sensor fusion accuracy. Experiments with prerecorded and simulated sensory data streams showed that, among the methods examined, the dynamically weighted Dempster-Shafer method — as its adaptive nature matches the dynamic behavior of humans — is the most suitable sensor fusion method for a general context sensing process. The dynamically weighted Dempster-Shafer sensor fusion method provides the following benefits:

- This method easily realizes a differential trust on sensors scheme, which is very hard to realize by other sensor fusion methods, and it easily allows humans to intervene in the reasoning process

- To some people, the conclusions of the Dempster-Shafer Evidence Combination Rule are counter intuitive when serious conflicts exist; this extended Dempster-Shafer method effectively mitigates the conflicts among the sources of evidence by lowering their assertiveness, thus making the Dempster-Shafer method more easily acceptable to various applications and users

- Using the information regarding the sensors' recent performance makes it adaptive to sensor drift, which is otherwise very hard to handle

The proposed top-down methodology to build context-aware systems is facilitated with the proposed system architecture. The goal of the context-aware computing system architecture is to simplify context usage and its sensing process, such that applications are insulated from the context extraction, and in parallel, the context-sensing implementations are transparent to context extraction and interpretation. Toward this goal, the concept-demonstration system has improved architectural support for context-sensing in the original widget-style Context Toolkit system in the following ways:

- The sensed context information is automatically classified and consolidated. In other words, the system has a more clearly layered structure, the sensed context is further separated from the sensing implementations, so that the user applications can concentrate on better using context instead of on where and how the context is obtained.

- It adds a sensor fusion module, and by using the Dempster-Shafer theory as its primary sensor fusion method; since the Dempster-Shafer method does not require the joint distribution of sensors' observation, it meets the challenge that the joint distribution is usually unavailable in context-aware computing applications

- Because the joint distribution is not required and the evidence combination is cumulative and commutative in the Dempster-Shafer sensor fusion process, the system is very robust to sensor set configuration change

- The system is an infrastructure-style architecture that is easily scalable, as the system is modularized, adding new sensor fusion algorithms to fuse context, or applying new artificial intelligence algorithms to extract context, will not affect the existing ones.

6.3. Future research suggestions

This research project thus far concerns only an individual sensor fusion process, which is only a very small part of the context-aware computing system. Since "context" has a very broad definition, some extra information that can enhance the sensor fusion processes may already exist; how to find and use such extra information is a very interesting topic for further research.

Generally speaking, when two context information pieces are correlated, the conclusion of one information piece can be used to verify the correctness of the other information piece. For example, in a user-identification applications, many sensors' observation can contribute to strengthening or weakening a hypothesis like "this is user A". The "sensor" set may include: fingerprint reader, image pattern from a camera, infrared feature from an infrared camera, the user's speaking sound, the user's height estimation from a motion detector or a stereo camera pair etc. Once the fused context information piece, e.g., "this is user A", has reached a certain confidence level, it ought to be included in the related sensor fusion inference processes such as "this document has been viewed by all the users on site".

Another issue that needs to be further carefully considered, regarding information pieces to populate in a system, is how to avoid a piece of information being abused because an information piece may reach a node via different paths. The commonly suggested practice (in military sensor fusion application domain) is to attach its source (from the original sensor) information to the being processed information pieces. But

how this scheme works in a context-aware computing system is a new topic that needs more research.

Different sensors will have different performances and will behave differently according to environmental changes. When the related information regarding sensors' working environment is included in a context information pool, it should be maximally used to tune up the related sensor fusion processes: when the conditions are changing in favor of one particular sensor set, information from this sensor set should be trusted more. For example, when the environment gets dark around a car, a laser scanner sensor should get more trust than a video camera in detecting nearby objects. Another way to use context information to tune up a sensor fusion process is through intelligently specifying constraints to affect sensor fusion algorithms' behavior, or to appropriately choose one of the many sensor fusion algorithm candidates [25]. For example, from vehicle location and velocity context information how fast objects can possibly move between two successive frames in object tracking is implied. Therefore, in a crowded downtown environment where the vehicle moves slowly, the tracking system can run more advanced object recognition algorithms to get more detailed information about the environment. In contrast, in a highway environment where the vehicle runs at high speed, then the tracking system will have to make do with some simpler algorithms to satisfy the requirement that everything must be done very fast.

This dissertation's advocated top-down context sensing methodology is the first approximation towards building a context information model based on the assumption that the context information can be described as an ensemble of trivial information pieces of discrete factors and events. While this approximation is a very effective way to simplify the context information model, there are many aspects left for future exploration. Perhaps the most important one is to model human feelings, where the status boundaries are very hard to define, hence advanced sensor fusion methods (e.g., the fuzzy logical method) need to be researched.

Appendix[26]

Appendix A

System Software Development

A.1. Tools and Environments Setup

In the Java virtual operating system environment, the basic Context Toolkit system was set up, and then based on it, the context sensing support system — a sensor fusion mediator module and a Dempster-Shafer algorithm implementation module were developed and tested. The test was on two sorts of computer platforms: a Windows 2000 (SP3) PC with Java 1.4.0 JRE/JDK (later upgraded to 1.4.1_02) Standard Edition installed, and two SGI Indigo2 Unix workstations (Irix 6.5.11m operating system) with Java 1.3.01 and 1.3.1 JRE/JDK Standard Edition installed respectively.

The PC hardware includes an AMD 450MHz CPU (later after the machine was replaced with a Pentium 450MHz) with 448M RAM, and the two SGI Indigo2 Unix workstations have 195MHz R10k CPU's with 512M and 256M RAM respectively. Running the developed system applications on these platforms shows no sign of difficulty in terms computation power. However, the Java program debugging, especially in IDE (Integrated Development Environment), is much more demanding on computation power, so the process is tedious. Most of the programming code was developed on a Dell laptop PC with Pentium 1.9GHz CPU and 1G RAM, and then the code was tested (with some minor adjustments) on the slower PC and two Unix workstations. From the experience of using the Context-Aware Toolkit system and further developing it, the recommended minimum hardware requirement is something-like a Pentium 1.5GHz 512M RAM PC.

[26] This part mainly serves as the Motorola UPR final technical report. It may be revised independent of this dissertation to meet Motorola's needs for archival documentation.

The Java programming tools used in the Unix workstations were JDE 2.1.9 (Java Development Environment for Emacs). Two IDE software development tools were tried in Windows platform: the Borland JBuilder 7 (Commercial Licensed Edition) and the Sun One Studio 4 Community Edition (previously called Sun Forte for Java 4.0 Community Edition). It is discovered that although the Sun One Studio is much slower, it does a much better job in tracing source line execution, thus it is much better for program debugging. So, most of the work was done in Sun One Studio 4 CE IDE.

To facilitate the software development process, a Samba file server system was also set up on one of the SGI machines (DEMPSTER.SENSOR.RI.CMU.EDU, IP address 128.2.196.30) so that the two kinds of machines could easily transfer files through mapping a (Unix) server's file directory to a hard disc drive in Windows system.

To implement a dynamic context database, a MySQL DBMS database server was setup in the PC windows (ODOR.SENSOR.RI.CMU.EDU, IP address 128.2.213.132).

A web server (Apache 1.3.23) was also installed on the SGI Unix workstation DEMPSTER for experimenting an additional context information exchange interface. The URL address was http://128.2.196.30:8080/. It hosted only static context information as web pages for development convenience, but the plan (beyond the thesis work) was to integrate it with the system to provide a means of monitoring what was going on inside the system and serving applications with required context information.

A.2. Software architecture background

To develop complex systems to incorporate ever-changing devices and to fulfill tasks that need to evolve from time to time, using a layered and modularized architecture is the only way to achieve the flexibility and scalability requirement. Based on experiences accumulated through extensive research and engineering practices, many system architecture styles have been developed. They are believed by their developers to have achieved optimal, or close to optimal, performances for various specific application situations.

Because context-aware computing is a relatively new research area, the characteristics of its applications have not been fully understood yet [46], so it would be beneficial to examine what has already been learned in related areas, and to use the experience.

A.2.1. Middleware approach to build context-aware computing systems

In computer software engineering, the most commonly used methodology of developing layered and modularized complex systems is to use the "middleware" (also called "component-based") approach ([96], [97], [116]). Middleware sits as a layer above the computer operating system and networking software and below the applications to provide communication and other services that facilitate system integration.

Middleware simplifies the system development by hiding and isolating the complexity of the communication between them [102]. A middleware service is defined by the APIs (the application programming interfaces) and the protocols (the data format) it supports. The middleware work has developed significantly over the years, perhaps the most influential existing specifications or infrastructures are: the Object Management Group's (http://www.omg.org/, the largest software industry consortium) CORBA (Common Object Request Broker Architecture), Sun Microsystems' J2EE (Java 2 Enterprise Edition), and the Microsoft's DCOM (Distributed Common Object Model) and .NET ([117], [118], [127]).

Most of the context-aware computing systems developed thus far use this middleware approach in form of building blocks or templates to facilitate building information services or agents ([79], [101], [103], [125]), although some systems do not emphasize this methodology explicitly.

A.2.2. System architecture for network-based computing

Since all middleware infrastructures strive to support a large spectrum of applications, they often provide many ways to interact and communicate within a system, thus there can be many system configurations to fulfill a required functionality [102]. System

engineering research aims to create an optimally functional integrated system out of independently engineered piece-parts.

System engineering has gained increasing attention, where software architecture is widely researched ([104], [105], [112]). For the last ten years, system architecture research and application have advanced rapidly [119] in conjunction with software reusable pattern research [120] and formal system conceptual modeling research ([118], [121], [122]).

Software architecture specifies the overall structure, which includes: gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; physical distribution; composition of design elements; scaling and performance; and selection among design alternatives [99]. This level of design goes beyond the algorithms and data structures in that its explicit focus is on connectors and configurations [119].

Software architecture research uses an architecture description language (ADL). Current ADLs have different syntax and conceptual frameworks, but they all use three building blocks for architectural description: (1) components – units of localized and independent computation or data store with interface; (2) connectors – architectural building blocks used to model interactions among components and rules that govern those interactions; and (3) architectural configurations – connected graphs of components and connectors that describe architectural structure.

Since system software architectures can have so many possible variations, the term “architecture style” is used for recognizing a shared repertoire of methods, techniques, patterns and idioms in structuring complex software systems. System architectural style description makes complex systems more tangible as it provides significant semantic contents about the kinds of properties of concern, the expected paths of evolution, the overall computational paradigm, and the relationship between this system and other systems [123].

However, there is no simple taxonomy of system architecture styles; the system architecture domain is often closely related to concerns of application domains ([99], [117]). Shaw et al. [98] group the relatively well-known styles into two families: dataflow networks (with 3 variants) and the cooperative message-passing processes (with 8 variants). The dataflow networks family, also called "pipe and filter", has components asynchronously transforming input into output with minimal retained state. The cooperative message-passing processes family has components, called processes, communicating with each other via message passing.

Fielding [93] summarizes commonly used architecture styles for network-based systems and specifies five groups: dataflow, replication, hierarchical, mobile code, and peer-to-peer styles.

- The dataflow group has two typical styles in use: the pipe and filter, and the uniform pipe and filter style.
- The replication group also has two typical styles in use: the replicated repository, and the cache style.
- The hierarchical group has seven typical styles in use: the client-server, the layered client-server, the client-stateless-server, the client-cache-stateless-server, the layered client-cache-stateless-server, the remote session, and the remote data access style.
- The mobile code group has five typical styles in use: the virtual machine, the remote evaluation, the code on demand, the layered code-on-demand client-cache-stateless-server, and the mobile agent style.
- Finally, the peer-to-peer group has four typical styles in use: the event-based integration, the C2 [124], the distributed objects, and the brokered distributed objects style.

Because the ultimate goal of context-aware computing is to make computer controlled systems understand human users' environment and provide services

pervasively, context-aware computing systems have to be network-based, thus the experience gained in this area is important.

A.2.3. Event-based/agent architecture versus context-aware computing

For the network-based systems, Carzaniga et al ([101], [103]) suggested using an "event-based architecture style" to describe the structure and interaction patterns in large-scale distributed systems where the components communicate by generating and receiving event notifications. Carzaniga et al's event-based architecture roughly coincides with Fielding's event-based integration style in the peer-to-peer styles group, but may include variations of other styles, e.g., the client-server style, the mobile agent style, etc.

In event-based architecture descriptions, the connectors are event services in charge of dispatching event notifications, whereas components can be either "recipients" or "objects of interest". The recipients declare their interest in receiving event notifications by subscribing to the event services; the objects of interest notify the occurrence of an event to the event services (alternatively the event services can poll objects of interest to know whether some event has been produced). A component can behave both as a recipient of an event and an object of interest. The event-based architecture style can be realized with a number of previous mentioned middleware infrastructures [100].

The performance of a context-aware computing system largely depends on its system architecture. Most of the context-aware systems developed thus far use the event-based architecture or its variations in the peer-to-peer style group and in the hierarchical style group, and almost all the system architectures use software agents [114].

Autonomous software agent programming is widely studied recent years ([38], [45]). Franklin et al analyzed the agent programming techniques and gave it a formal definition as: "an autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future." [95] The characteristics of software agents (such as

being reactive, autonomous, goal-oriented, temporally continuous, communicative, "personality", etc.) make it very suitable for context-aware computing.

A.3. Software package development

As described before, the context sensing system software development is based on the Georgia Tech's Context Toolkit system. The original Context Toolkit system software package is shown in Figure 19, where the "arch" block stands for the main context processing system architecture, which has many sub-packages to realize the required basic functionalities, and the "apps" block stands for system applications. It can be seen from Figure 19 that the original system design separates the context collecting and context distributing functional blocks from the context-aware applications.

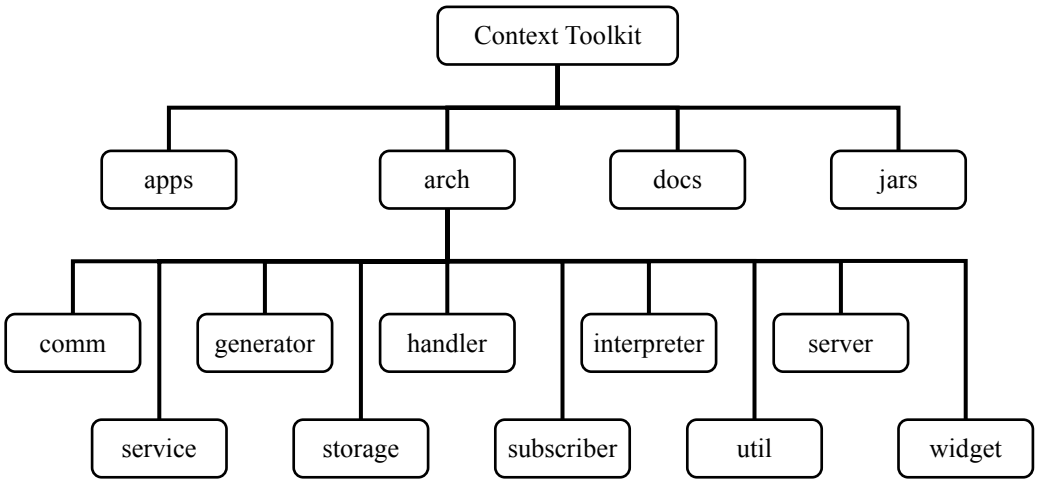

Figure 19. Original Context Toolkit System Software Package

As pointed out in Section 1.3.4, it can also be easily seen in Figure 19, that the Context Toolkit system does not provide system architectural support for sensor fusion. Notice that from software implementation perspective, the "widget" component is

regarded as a part of the Context Toolkit system architecture, which is perhaps not a good design from context-sensing implementation point of view. Because the available sensors are generally considered as a dynamic set for a typical context-aware system, placing the sensors' highly coupled widgets inside the system architecture would possibly make system software maintenance more difficult than necessary.

To improve sensor fusion system architectural support meanwhile keeping compatibility with the original Context Toolkit, the developed context-sensing software package structure is shown in Figure 20.

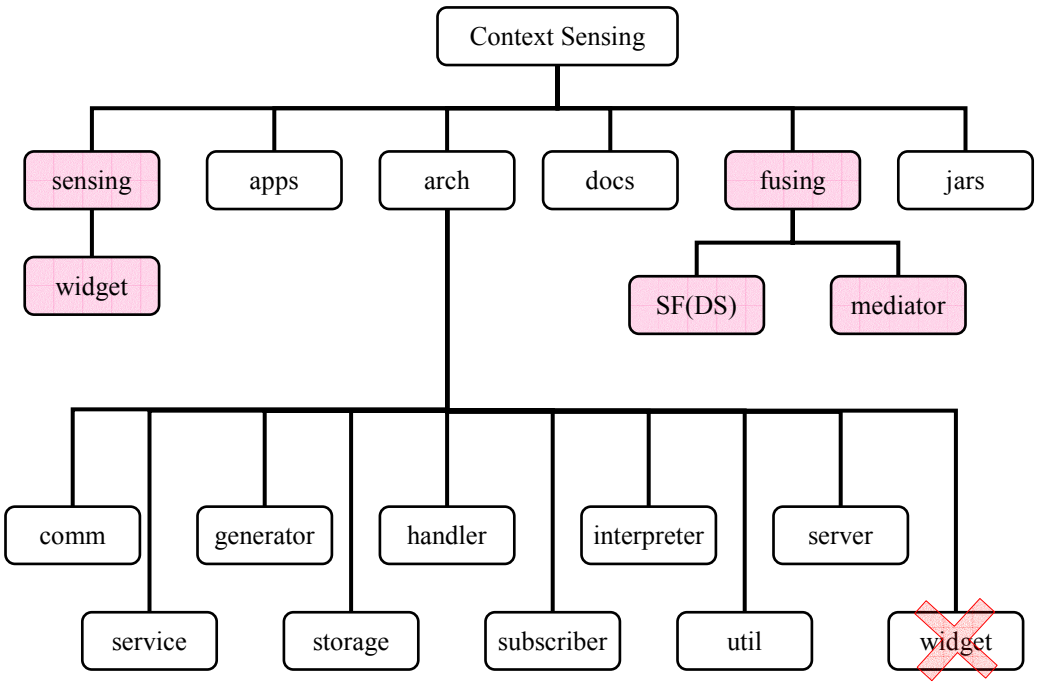

Figure 20. Context sensing software package structure based on the Context Toolkit system

From Figure 20 it can seen that two software packages, “sensing” and “fusing” boxes, each loosely coupled with the original system architecture, are added to the system. Placing the sensing package, which includes all sensor widgets, outside the context system basic architecture makes the system software maintenance easier when adding or changing sensors’ sensing implementation. The “fusing” package contains two kinds of components: the sensor fusion algorithm module to fuse competitive context information pieces (Dempster-Shafer algorithm as the first module implementation) and the sensor fusion mediator module to manage sensors’ widgets and call the sensor fusion engine module to fulfill sensor fusion tasks.

Not shown in the software structure Figure 20, this dissertation gives a new interpretation of the Context Toolkit system components usage: using the Toolkit system to build infrastructure-style system what delivers part of blackboard-style system advantages. The details are describe in Section 3.1.

A.4. Dempster-Shafer algorithm implementation

The system design emphasizes modular structure so that the sensor fusion algorithm implementation modules (for now only the Dempster-Shafer module is implemented) can be used by all sensor fusion mediator instances, each one dealing with a specific information item. More sensor fusion algorithms can be added in future without changing the already existed ones.

Using the meeting-participants’ focus-of-attention analysis as an example, Figure 21 illustrates the programming class object and data structure implementation. Referring to Figure 21, the `hypolist` Hash-table keeps a list of all possible evidence that can be observed, called “frame of discernment” in the Dempster-Shafer reasoning system. All evidence objects, or hypothesis set information, are represented in the `HypothesisSet` data structure, and every sensor widget keeps its “probability mass function” in an `evidenceList` data structure.

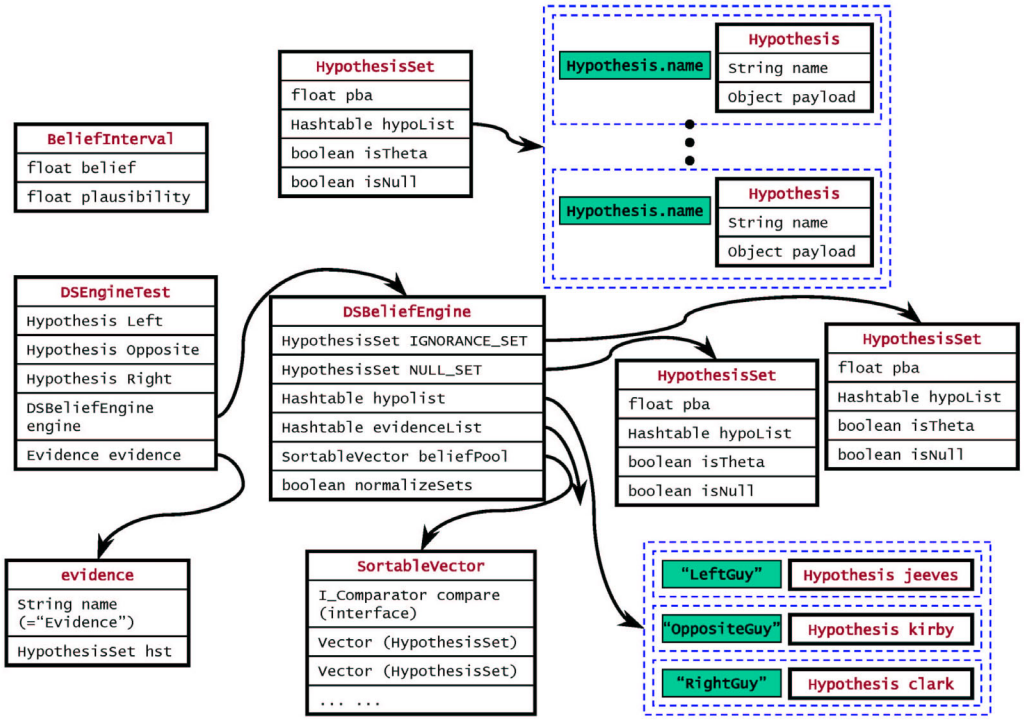

Figure 21. Dempster-Shafer Theory of Evidence programming implementation

The reasoning system maintains its combined best believes in the `beliefPool` data structure, which is updated whenever new evidence is provided by the sensor widgets.

The Dempster-Shafer sensor fusion algorithm with weighting is implemented outside the core Dempster-Shafer program structure shown in Figure 21. There are two sets of Dempster-Shafer reasoning API's, one is for the standard Dempster-Shafer sensor fusion method and the other is for the weighted Dempster-Shafer sensor fusion method. The usage of Dempster-Shafer API's is described in details in Appendix C.

With the standard Dempster-Shafer method, each evidence set's "basic probability assignments" will be normalized and a proper ignorance value will be assigned if necessary. With the weighted Dempster-Shafer method, each evidence set's "basic probability assignments" are further adjusted to accommodate the sensor reliability judgment with the given weight and increase the corresponding ignorance accordingly.

The dynamically weighted Dempster-Shafer sensor fusion method is realized by calling the Dempster-Shafer module with the weights constantly being updated with each new evidence data set according to the correctness of the last performance.

Appendix B
Concept-Proving Demonstration Experiments

To illustrate the feasibility of the proposed top-down context sensing methodology, a prototypal distributed system was built. Prerecorded and simulated sensory data were used for the demonstration, because the research focus is on the sensor fusion aspect of the context information processing — dealing with higher-level information and mainly concerning with the whole system behavior in terms of information flow.

As described in dissertation Section 4.1, the prerecorded "meeting-participant's focus-of-attention analysis" experimental data were used in the concept-proving demonstration. The original experiments and the preprocessing of the sensory data were done by Rainer Stiefelhagen in his PhD dissertation research. Using Stiefelhagen's preprocessed experimental data is reasonable solution given the workload/resource limitation and the complementary relationship between the two research projects — his research focus was to deal with visual and audio signals to estimate focus of attention.

As described in Chapter 2, this dissertation proposed a context-sensing methodology involving two major steps: first is to specify context information architecture, and second is to implement the sensor fusion process with the help of system architectural support. The details of the two-step realization is described in the following sections.

B.1. Specifying context information architecture

The general model of the context description begins with two kinds of basic semantic entities: the STAGE[27] entity to describe environmental information and the USER entity to record the service-targeted human computer-user information. For the case of meeting-participant's focus-of-attention application, we can imagine the following scenario as an example to build context database:

- The STAGE entity class has a subtype class SITE, which has an instance object named SmallConferenceRoom
- There are 2 pieces of EQUIPMENT object in the STAGE instance: a ConferenceTable and a Camcorder
- The SENSOR entity type has two entries: OmniCamera, and Microphones, which together correspondingly have an audio Widget and visual Widget pair, one for each USER in meeting at the site of STAGE.
- As STAGE's subtype the SITE object instance SmallConferenceRoom has extra properties of "Brightness" and "NoiseLevel"
- The USER entity class has five instance objects (users): User-A, User-B, User-C, User-D, and User-E
- The USER's PREFERENCE object records his/her preference properties as of: "MeetingTime", and "ContactMethod"
- The USER's SCHEDULE entity is also maintained by the system
- The USER's following ACTIVITY object types are recorded: GroupMeeting, and FOA (focus-of-attention)
- For the event we were monitoring, there were 4 USER's at the SITE SmallConferenceRoom; and at the SITE entity, a GroupMeeting object instance of USERSACT class (users' group activity) was observed

[27] To make it easier to follow, whenever it needs to be clarified, the following items are shown in special font (Lucida Console font): entity class names, object instances of entity class, table names, and the key of tables.

- For each meeting-participant USER, the ACTIVITY entity class has a FOA object instance, which can take value of the left-side USER, the straight-across-table USER, or right-side USER

Using this application scenario example, the semantic object specifications are listed in Table 12.

Table 12. Semantic object specifications for context information modeling

Object Name	Property Name	min#	max#	ID	Domain Meaning
STAGE	*StageID*	1	1	ID	Stage ID
	StageName	1	1		Stage name
	Functionality UsageRules	1 1	N N		Utility functionality Usage regulations
	EQUIPMENT	0	N		Equipment in STAGE
	SENSOR	0	N		Sensors in STAGE
	WIDGET	0	N		Sensor Widgets in STAGE
	USER	0	N		Users at the STAGE
	USERSACT	0	N		User group-activity STAGE
SITE	*StageID*	1	1	ID	Subtype of STAGE
	Brightness	0	1		Brightness of the site
	NoiseLevel	0	1		Background noise
EQUIPMENT	*EquipID*	1	1	ID	Equipment ID
	EquipName	1	N		Equipment name
	Functionality UsageRules	1 1	N N		Utility functionality Usage regulations

Object Name	Property Name	min#	max#	ID	Domain Meaning
SENSOR	*SensorID*	1	1	ID	Sensor object
	SensorName	1	1		Sensor's name
	ContextEntry	1	N		Context info generated
	WIDGET	1	N		Sensor's WIDGET
WIDGET	*WidgetID*	1	1	ID	WIDGET to report context
	WidgetName	1	1		Widget descriptive name
	SENSOR	1	N		Sensors to WIDGET
USERSACT	*StageID*	1	1	ID	Group activity at STAGE
	StartTime	1	1	ID	Start time of group activity
	ActivityName	1	1	ID	Group activity name
	USER Probability	1 1	N N		Participating users Confidence of user' ID
USER	*UserID*	1	1	ID	Registered user's ID
	UserName FirstName LastName	1 1 1	1 1 1		User name First Name Last Name
	Title Affiliation	1 1	N N		Job title Affiliation of the job
	Sex	1	1		Male or female
	BirthDate Year Date	0 1 0	1 1 1		User's birth date Year for age-group purpose Birth date for event etc.
	Email	1	1		User's Email address
	CampusAddr Room Building	0 0 1	N 1 1		User's campus address Room number Building name

Object Name	Property Name	min#	max#	ID	Domain Meaning
	Phone	0	3		Phone number
	PREFERENCE	0	N		User's preferences
	ACTIVITY	0	N		User's activity description
	SCHEDULE	0	N		User's schedule table
PREFER	USER	1	1	ID	User's preferences
	PrefName	1	1	ID	Items can be chosen
	Preference	0	1		User's preference description
ACTIVITY	USER	1	1	ID	User's activity
	StartTime	1	1	ID	User's activity time
	ActivityName	1	1	ID	User's activity description
SCHEDULE	USER	1	1	ID	User's schedule
	StartTime	1	1	ID	Event starting time
	EndTime	0	1		Event ending time
	EventName	1	1	ID	Event Name
	Importance	0	1		Importance of the event

For our meeting-participant's focus-of-attention analysis application scenario, we built a context database to describe the context information. The database tables with column properties are described in the following list, and the context database table entry and the constraint descriptions are listed in Table 13. Here, the italic format indicates foreign key entries, and the underlined italic format denotes the key of the context database tables.

STAGE(*StageID*, StageName);

StageFunc(*StageID*, *Functionality*, UsageRules);

SITE(*StageID*, Brightness, NoiseLevel);

EQUIPMENT(*EquipID*, EquipName, *StageID*);

EquipFunc(*EquipID*, Functionality, UsageRules);

SENSOR(*SensorID*, SensorName);

Sensing(*SensorID*, InfoPiece);

WIDGET(*WidgetID*, WidgetName);

SensorWidget(*SensorID*, *WidgetID*);

USERSACT(*StageID*, *StartTime*, *ActivityName*);

UserGroup(*StageID*, *StartTime*, *ActivityName*, *UserID*, Probability);

USER(*UserID*, FirstName, LastName, Sex, Email);

UserTitle(*UserID*, Title, Affiliation);

BirthDate(*UserID*, Year, MonthDate)

CampusAddr(*UserID*, *Room*, *Building*, Phone1, Phone2, Phone3);

PREFER(*UesrID*, *PrefName*, Preference);

SCHEDULE(*UserID*, *StartTime*, EndTime, *EventName*, Importance);

ACTIVITY(*UserID*, *StartTime*, *ActivityName*);

Sitting(*StageID*, *StartTime*, *ActivityName*="Meeting", *UserID*, Probability, *UserID*, Probability, *UserID*, Probability);

FOA(*UserID*, *InstantTime*, *Activity*="FocusOfAttention", *UserID*, Probability);

Sitting4(*StageID*, *StartTime*, *ActivityName*="Meeting", *UserID*, Probability);

FOA4(*UserID*, *InstantTime*, ProbabilityLeft,
ProbabilityStraight, ProbabilityRight);

Generally speaking, the context database design procedure can follow the standard "database design with semantic object model" guidelines [34], i.e., a new table is created for each kind of item that has a dynamic instance-repeatability number. For example, the USERSACT table lists all USER group activity names that are known to the system and a new table, the UserGroup table, is created to further describe which USER is participating those group activities — since the UserGroup belongs to the USERSACT entity class, they use the same key.

However, since context information can be extremely complex to model, some restrictions need be taken care of diligently. For example, the STAGE's user group activity USERSACT table has a "Meeting" entry, and for each user the USER's ACTIVITY table has a few entries such as "Sitting", "Speaking", "Meeting", and FOA (Focus-Of-Attention), etc.; notice that some constraints apply regarding how these items can coexist.

Notice that since the Sitting table describes the settings regarding who is sitting beside whom for a special case of user group activity USERSACT, where the ActivityName property value equals "Meeting" and some USER's ACTIVITY table has "Meeting" and "Sitting" entries, and since the Sitting4 table describes the settings regarding who is sitting straight across the conference table for a further specified special case of the Meeting events whereas it has 4 participants, so from USERSACT to Sitting, and from Sitting to Sitting4, an inheritance relationship exists; this relationship is reflected in table design as the same key (joint key here) is used in the three tables.

A USER's focus-of-attention object FOA in general is described in the format of: the subjective *UserID* plus the *current-time* as joint key, and the *UserID* of those other USER's that might have this USER's focus-of-attention, plus a corresponding probability number indicating the confidence of this judgment. However, suppose each original sensor WIDGET only knows how to report a USER's focus-of-attention in FOA4

format (the USER's FOA is on the left, straight-across-table, or right-side USER), then a further context query operation is needed to bind the relative position information to the USER's identification information.

Table 13. Context database table entries description and constraints

Table Name	Property Name	Physical description	Semantic constraints
STAGE	*StageID*	Integer	
	StageName	Text 50	
StageFunc	*StageID*	Integer	*StageID* must already exist in STAGE table
	Functionality	Text 30	
	UsageRules	Text 300	
SITE	*StageID*	Integer	*StageID* must already exist in STAGE table
	Brightness	Double	
	NoiseLevel	Double (db)	
EQUIPMENT	*EquipID*	Integer	
	EquipName	Text 50	
	StageID	Integer	*StageID* must already exist in STAGE table
EquipFunc	*EquipID*	Integer	*EquipID* must already exist in EQUIPMENT table
	Functionality	Text 30	
	UsageRules	Text 300	

Table Name	Property Name	Physical description	Semantic constraints
SENSOR	*SensorID*	Integer	
	SensorName	Text 50	
Sensing	*SensorID*	Integer	*SensorID* must already exist in SENSOR table
	InfoPiece	Text 30	
WIDGET	*WidgetID*	Integer	
	WidgetName	Text 50	
SensorWidget	*SensorID*	Integer	*SensorID* must already exist in SENSOR table
	WidgetID	Integer	*WidgetID* must already exist in WIDGET table
USERSACT	*StageID*	Integer	*StageID* must already exist in STAGE table
	StartTime	Time	
	ActivityName	Text 50	
UserGroup	*StageID*	Integer	The key must already exist in USERSACT table
	StartTime	Time	The key must already exist in USERSACT table
	ActivityName	Text 50	The key must already exist in USERSACT table
	UserID	Integer	*UserID* must already exist in USER table
	Probability	Double [0.0, 1.0]	
USER	*UserID*	Integer	

Table Name	Property Name	Physical description	Semantic constraints
	FirstName	Text 30	
	LastName	Text 30	
	Sex	Text 10	
	Email	Email address	
UserTitle	*UserID*	Integer	*UserID* must already exist in USER table
	Title	Text 30	
	Affiliation	Text 50	
BirthDate	*UserID*	Integer	*UserID* must already exist in USER table
	Year	Integer (1900, 2003)	
	MonthDate	Date	
CampusAddr	*UserID*	Integer	*UserID* must already exist in USER table
	Room	Text 10	
	Building	Text 30	
	Phone1	Text 20	
	Phone2	Text 20	
	Phone	Text 20	
PREFER	*UserID*	Integer	*UserID* must already exist in USER table
	PrefName	Text 30	
	Preference	Text 50	

Table Name	Property Name	Physical description	Semantic constraints
SCHEDULE	*UserID*	Integer	*UserID* must already exist in USER table
	StartTime	Time	
	EndTime	Time	
	EventName	Text 50	
	Importance	Integer [0, 100]	The relative importance of the task
ACTIVITY	*UserID*	Integer	*UserID* must already exist in USER table
	StartTime	Time	
	ActivityName	Text 50	
Sitting	*StageID*	Integer	The key must already exist in the USERSACT table *StageID* must already exist in UserGroup table
	StartTime	Time	The key must already exist in the USERSACT table
	ActivityName	Text 50	The key must already exist in the USERSACT table *ActivityName* must be equal to "Meeting"
	UserID	Integer	The USER we are currently concerned with *UserID* must already exist in UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal "Meeting"
	UserID	Integer	The user at left-side *UserID* must already exist in

Table Name	Property Name	Physical description	Semantic constraints
			UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal “Meeting”
	Probability	Double (0.5, 1.0]	To make the next focus-of-attention estimation meaningful, this *UserID* judgment must have a relative high confidence
	UserID	Integer	The user at right-side *UserID* must already exist in UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal “Meeting”
	Probability	Double (0.5, 1.0]	To make the next focus-of-attention estimation meaningful, this *UserID* judgment must have a relative high confidence
Sitting4	*StageID*	Integer	The key must already exist in Sitting table
	StartTime	Time	The key must already exist in Sitting table
	ActivityName	Text 50	key must already exist in Sitting table *ActivitName* must be equal to “Meeting”
	UserID	Integer	The USER we are currently concerned with *UserID* must already exist in Sitting table
	UserID	Integer	The user at straight-across *UserID* must already exist in

Table Name	Property Name	Physical description	Semantic constraints
			UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal “Meeting
	Probability	Double (0.5, 1.0]	To make the next focus-of-attention estimation meaningful, this *UserID* judgment must have a relative high confidence
FOA	*UserID*	Integer	The USER whose focus-of-attention is of our concern *UserID* key must already exist in UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal “Meeting”
	InstantTime	Time	The key must already exist in ACTIVITY table
	ActivityName	Text 50	The key must already exist in ACTIVITY table *ActivityName* must be equal to “Meeting””
	UserID	Integer	The USER that might be at the focus-of-attention *UserID* key must already exist in UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal “Meeting”
	Probability	Double (0.0, 1.0]	Confidence of the judgment
FOA4	*UserID*	Integer	The USER whose focus-of-attention is of our concern The key must already exist in FOA table

Table Name	Property Name	Physical description	Semantic constraints
			UserID key must already exist in UserGroup table *UserID* must already exist in ACTIVITY table associated with the *ActivityName* equal “Meeting”
	InstantTime	Time	The key must already exist in FOA table
	ActivityName	Text 50	The key must already exist in FOA table *ActivityName* must be equal to “Meeting””
	ProbabilityLeft	Double (0.0, 1.0)	Confidence of the judgment that FOA is on the left-side user
	ProbabilityStraight	Double (0.0, 1.0)	Confidence of the judgment that FOA is on the straight-across-table user
	ProbabilityRight	Double (0.0, 1.0)	Confidence of the judgment that FOA is on the right-side user

B.2. Implementing and demonstrating the focus-of-attention fusion case-study application

For system development programming, the user must copy the contents of the Context Sensing directory (including all its subdirectories and contents as shown in Figure 20) to a desired destination directory on the host computer. For the concept-demonstration experiments, the destination directory is set as “fusion”.

It is suggested that all context-sensing sensor widgets be placed in the “sensing” subdirectory, and that each application has its own sub-subdirectory. In the concept-

demonstration experiments, the sub-subdirectory is named “appsFocusOfAttention”, and there are two widgets `WfoaAudio` and `WfoaVisual` inside it, standing for the audio-sensor widget and the visual-sensor widget respectively.

As it is also shown in Figure 20, the Dempster-Shafer reasoning engine (algorithm implementation software package) is already placed inside the “fusing” subdirectory in a sub-subdirectory named “DempsterShafer”, ready to be called by sensor fusion mediators. More sensor fusion engines, or algorithm implementation packages, are to be developed and placed in this “fusing” subdirectory parallel to the Dempster-Shafer engine. It is suggested that all sensor fusion mediator programs be placed in the “fusing” subdirectory also, each application has its own sub-subdirectory parallel to sensor fusion engines’ directory. In the concept-demonstration experiments, the focus-of-attention application has its directory named “appsFocusOfAttention”, which has a sensor fusion mediator `foaFusing` in it.

For system deployment, the same directory structure as of development system should be preserved, but only the compiled java “.class” files need to be present. To demonstrate how it works, a text file named “person0.text”, which has the prerecorded focus-of-attention experimental data as the source to simulate real sensors, is placed in the root directory, i.e., the “fusion” directory.

In the demonstration, there are two simulated sensors (audio and visual sensor widgets) and one sensor fusion mediator (focus-of-attention mediator). They can run from three different platforms.

Using the same prerecorded experimental data source file, to be run at a specific time start point, the two simulated sensor widgets can be approximately synchronized to report their observations to the sensor fusion mediator regarding a specific meeting-participant’s instantaneous focus-of-attention status.

To simulate the uncertainty in data communication over a network, each simulated sensor widget does not send out its observation reports if a randomly generated number is smaller than a preset threshold; and in the case that the random number is larger than the

threshold so the reports are to be sent out, the transmission time is randomly delayed from the synchronized point within certain period.

As described in Section 4.1, the audio and visual sensor widgets will each report a user's focus-of-attention context in the format of 3 probability numbers corresponding to this meeting-participant's focus-of-attention is on the left-side person, the person straight across the table, or the right-side person.

Assuming Java platform and its class path etc. environments are already properly set on the host computer, to invoke the simulated sensor widget program, in a console or terminal windows change to the "fusion" directory first then type:

```
>java context.sensing.appsFocusOfAttention.WfoaAudio
meeting PORT# false
```

to simulate the audio sensor widget, or type:

```
>java context.sensing.appsFocusOfAttention.WfoaVisual
meeting PORT# flase
```

to simulate visual sensor widget.

Here, the "meeting" specifies that this widget is going to report to a sensor fusion mediator that accepts focus-of-attention analysis data in a 4-person meeting discussion situation; the parameter `PORT#` should be substituted by an integer number that specifies which communication port number this sensor widget uses; and the parameter `false` specifies that the raw sensory data will not be recorded to the context database.

The invoked sensor widget simulation program will pop up a small window as shown in the lower part of Figure 13, where the default simulation source file name can be changed, the maximum random delay time can be specified (the default number is 2 seconds), and a future time should be specified to start the simulation process. Then, click the "start simulation" button; the widget simulation processing will start at the given time.

To invoke the sensor fusion mediator program, change to the "fusion" root directory and type:

```
>java context.fusing.appsFocusOfAttention.foaFusing meeting
PORT#0 S#1Host S#1Port# S#2Host S#2Port#
```

Here, the parameter `meeting` indicates that this sensor fusion mediator is in charge of a meeting-participant's focus-of-attention analysis. The parameter `PORT#0` should be substituted by an integer number that specifies which communication port number this sensor fusion mediator uses. The parameters `S#1Host` and `S#2Host` should be substituted by character strings specifying computer hostnames that the two sensor widgets are hosted on respectively, and the parameters `S#1Port` and `S#2Port` should be substituted by integer numbers specifying which communication port number the two sensor widgets use respectively.

Once invoked, the sensor fusion mediator waits for incoming reports from sensor widgets. Regardless which report comes first, the focus-of-attention estimation reports of the same time stamp will be fused together, which is done at the time that a report of next time stamp has come in.

The fused result is shown graphically in the sensor fusion mediator program windows, and for comparison, both audio and visual sensor widgets' focus-of-attention reports are shown also. For users' to conveniently use the information, a confidence measurement is defined as the biggest probability number over the next biggest probability number as described in **EQ.17**. Thus, the context-sensing system provides a sensible explanation regarding the effectiveness of the sensor fusion method: the confidence indication of sensor fusion result will increase when the two sensor widgets' outputs agree with each other, otherwise the confidence measurement will decrease.

Appendix C

Sensor fusion API description

C.1. Class BeliefInterval

This helper class maintains the belief and plausibility information of an element of evidences or hypotheses in float numbers. It has a private format method formatFloat() to round the float numbers to the precision to the first three digits after the decimal point. The format function is mainly for the probability numbers to be printed out via the toString() method.

`public float belief`

This parameter keeps the basic probability assignment value, i.e., the lower boundary, of the current BeliefInterval object.

`public float plausibility`

This parameter keeps the plausibility value, i.e., the higher boundary, of the current BeliefInterval object.

`public BeliefInterval(float b, float p)`

Constructor to build new instance of BeliefInterval class

Parameters:

`b` — basic probability (i.e., belief, lower-bound probability) number

`p` — plausibility (i.e., higher-bound probability) number

`public String toString()`

To print out the belief interval, the result is formatted with precision of 3 digits after the decimal point, e.g. [0.213 0.965].

Overrides:

`toString` in class `java.lang.Object`

C.2. Class DSfusing

This is the core module class that implements the Dempster-Shafer reasoning process. The reasoning system uses a HypothesisSet class instance to register its frame of discernment information as a private object called IGNORANCE. The system always maintains its current basic probability mass functions in a vector variable, and newly incoming evidence set is kept in a buffer vector, thus it can control when the evidence will be combined into the reasoning system.

`private final HypothesisSet IGNORANCE_SET`

This private parameter is for keeping a HypothesisSet object that encloses all possible hypothesis occurrences, i.e., all the registered Hypothesis objects that collectively comprises the frame of discernment.

`private final HypothesisSet NULL_SET`

This private parameter is for temporarily keeping “impossible” combinations of HypothesisSet objects, i.e., this place is to keep conflict information generated in combining the system belief mass function with the new evidences in the system evidence pool.

`private Hashtable hypoList`

This private parameter is to register all hypotheses that the current reasoning system will be able to recognize. The Hashtable is supposed to store all the Hypothesis objects via the class’ RegisterHypothesis function during system initialization process.

`private SortableVector beliefPool`

This private inner-class SortableVector object is used for the system’s belief mass function storage, i.e., it is the system belief pool. Such information is stored in format of non-trivial HypothesisSet objects.

`private SortableVector evidencePool`

This private inner-class SortableVector object is used as the system’s evidence pool. It buffers new evidence objects in format of HypothesisSet objects.

`public DSfusing()`

This is the default constructor of DSfusing class. All it does is to set the flag of the inside HypothesisSets objects IGNORANCE_SET and NULL_SET, and to prepare two empty SortableVector objects for beliefPool and evidencePool.

`public void addEvidencePiece(HypothesisSet hypset)`

This method only adds the given HypothesisSet object into the SortableVector evidencePool object maintained by the reasoning system. It does not fulfill any evidence or belief reasoning updating tasks.

`public void addEvidence(Evidence e)`

This method adds a new Evidence object to the Dempster-Shafer reasoning system in fulfilling an update process. If the system's current belief mass function is empty, the incoming Evidence object's HypothesisSet, along with a newly created IGNORANCE_SET object, is simply added into it; otherwise, it calculates and updates the system's belief mass function.

Parameters:

`e` – The new Evidence object being added to the reasoning system.

`private void calcIntersectionTableau(Evidence newE)`

This method updates the belief mass function with the given Evidence object. It first completes the new Evidence object with its corresponding ignorance HypothesisSet, whose Hypothesis list encompasses the frame of discernment with probability assignment equal to (1 - [newEvidence's bpa]). The intersection between current beliefs and the completed new evidences is then calculated and the result is normalized. Finally, the belief system's mass belief function is updated with the normalized result.

Parameters:

`newE` – the Evidence object to be combined into the Dempster-Shafer reasoning system's belief mass function.

`private SortableVector`
`combinePasses(Vector p1, Vector p2)`

This method combines two sets of HypothesisSet objects stored in format of two Vectors.

Parameters:

`p1` – Vector of HypothesisSet object set to be combined.

`p2` – Vector of HypothesisSet object set to be combined.

Returns:

It returns the combined set of HypothesisSet objects in a format of SortableVector object.

`public float completeEvidencePool()`

This method completes the evidence pool's HypothesisSet objects with an ignorance HypothesisSet object. It first checks whether the sum of all HypothesisSet objects' probability assignment (bpa) is in the range of [0.0f, 1.0f). It will then assign a value of (1 - sum-of-bpa's) to a newly created IGNORANCE_SET object if this is true. In the case that the sum is not in the range of 0.0f and 1.0f, that is, an exception occurs, it will print out an error message.

Returns:

It returns the sum of basic probability assignment vlaues of the HypothesisSet objects in the system's original belief pool

`public void completeEvidencePool(double weight)`

This method first uses the given weight factor value to adjust each HypothesisSet object's basic probability assignment in the system's evidence pool and then complementarily complete the evidence collection by adding an ignorance HypothesisSet object. Under normal conditions, the sum of basic probability assignment value of the HypothesisSet objects in the system's evidence pool should be within the range of [0.0f, 1.0f), and adding the ignorance HypothesisSet object should then normalize the sum of the basic probability assignments of all the HypothesisSet objects to the value of 1.0. An exception will be generated if the sum is out of the [0.0f, 1.0f] boundary.

Parameters:

`weight` – the weight factor for deciding the voting effectiveness of the evidence collection in system's evidence pool.

`public void updateBeliefPool()`

This method updates the system's belief mass function with the evidences stored in the system evidence pool. It first checks whether the system's belief mass function (i.e., the belief pool) is empty or not, if so, it will move all the HypothesisSet objects in evidence pool into it and clear the evidence pool. It then scans the system evidence pool, and for each HypothesisSet object in it, it finds out the intersections with the objects in the system belief pool. The associated intersection HypothesisSet objects are combined, and the system belief mass function is thus updated with the results. Finally, the system's evidence pool is set to empty, indicating that there is no longer any unanalyzed evidence left.

`public void resetBeliefPool()`

This method clears the system's belief pool so that the system's belief mass function is empty. Any HypothesisSet objects in the system's belief pool are simply discarded.

`public BeliefInterval getBeliefInterval(HypothesisSet hset)`

This method calculates belief interval of the given HypothesisSet object. The lower bound of the interval, i.e., the "belief" value is the sum of the basic probability assignment value of all the HypothesisSet objects, whose elements are contained in the given HypothesisSet object, in the system belief pool. The higher bound of the interval, i.e., the "plausibility" value is one minus the sum of all the basic probability assignment value of the HypothesisSet objects, whose elements are not a subset or superset of the given HypothesisSet object, in the system's belief pool.

Parameters:

`hset` – the given HypothesisSet object whose belief interval is of interest.

Returns:

It returns a BeliefInterval object whose basic probability assignment and plausibility value are properly set.

`public float[] getBeliefPoolBpaArray()`

This method returns an array of the basic probability assignment values of the HypothesisSet objects in the system's current belief pool, i.e., it returns the system's current belief mass function values.

Returns:

float number array of the basic probability assignment values, i.e., the system's belief mass function.

`public HypothesisSet getHighestBeliefSet()`

This method first sorts the HypothesisSet objects in the system's belief pool by their basic probability assignment value and then returns the HypothesisSet object that has the highest basic probability assignment value.

Returns:

It returns the HypothesisSet object whose basic probability assignment value is highest in system's current belief pool.

`public Hypothesis getBestHypothesis()`

This method goes through the system's current belief mass function to find out the HypothesisSet object that consists of only a single Hypothesis object and has the highest probability assignment value.

Returns:

It returns the best Hypothesis in the system's belief mass function.

`public String getBestHypothesisName()`

This method returns the name of the best Hypothesis. The best Hypothesis is the HypothesisSet object that has only a single Hypothesis object inside it and has the highest basic probability assignment value in the system's current belief pool.

Returns:

It returns a character string of the best Hypothesis' name.

`public Hypothesis getBestEvidence()`

This method first sorts the HypothesisSet objects in the system's evidence pool by their basic probability assignment value and then returns the HypothesisSet

object that consists of single Hypothesis object and has the highest basic probability assignment value.

Returns:

It returns the Hypothesis object whose basic probability assignment value is highest in the system's current evidence pool.

`public String getBestEvidenceName()`

This method returns the name of the best evidence object in the system's current evidence pool. The best evidence piece is the HypothesisSet object that has only a single Hypothesis object inside it and has the highest basic probability assignment value.

Returns:

It returns a character string of the name of the best evidence object.

`public Hypothesis getHypothesis(String name)`

This method checks to see if the Hypothesis object with the given name has been already registered in this Dempster-Shafer reasoning system.

Parameters:

`name` – the name of the Hypothesis being check for existence

Returns:

It returns a Hypothesis object with given name, null if not already exists in the system

`public HypothesisSet getHypothesisSet(Hypothesis h)`

This method wraps the given Hypothesis object in a HypothesisSet object.

Parameters:

`h` - the Hypothesis object to be in a newly created HypothesisSet object

Returns:

It returns a newly created HypothesisSet object whose only element is the given Hypothesis object.

`private HypothesisSet getIntersection( HypothesisSet hset1, HypothesisSet hset2 )`

This method finds the intersection of the two given HypothesisSet objects. It scans the two HypothesisSet objects to find all the Hypothesis objects that both sides contain. Such found common Hypothesis objects are added together to create a new HypothesisSet object, whose basic probability assignment value is set equal to the product of the two original HypothesisSet objects' probabilities.

Parameters:

`Hset1` – the first HypothesisSet object for the intersection calculation.

`Hset2` – the second HypothesisSet object for the intersection calculation.

Returns:

It returns a HypothesisSet object whose Hypothesis elements are contained in both of the original HypothesisSet objects, and the basic probability assignment value equals the product of the two original HypothesisSet objects' probabilities.

`private Vector intersectWithBeliefPool(HypothesisSet testSet)`

This method finds the intersection of the given HypothesisSet object with the HypothesisSet objects maintained in the reasoning system's belief pool. The intersections of given HypothesisSet object with every HypothesisSet object inside the system's belief pool are added up and the belief pool is updated with the results. The system's normalizeSet flag is set to "true" for future normalization operations if any null HypothesisSet object is generated from the intersection calculation.

Parameters:

`TestSet` – the HypothesisSet object to intersect with the HypothesisSet objects in the system's belief pool.

Returns:

It returns a Vector object that contains the intersection HypothesisSet objects.

`private SortableVector normalizeSets(Vector sets)`

This method takes a Vector of HypothesisSet objects and returns the non-null part of the HypothesisSet objects with normalized basic probability assignment value. The normalization operation divides the basic probability assignment value of

each non-null HypothesisSet object with one minus the sum of all the basic probability assignment values of the null HypothesisSet objects, and uses the quotient as the HypothesisSet object's new basic probability assignment value.

Parameters:

`sets` – the Vector of HypothesisSet objects to be normalized.

Returns:

It returns a SortableVector object that contains the non-null HypothesisSet objects whose basic probability assignment value is normalized.

`public String printBeliefIntervals()`

This method helps to print out the belief interval of the system's current belief mass function. It prints out all the HypothesisSet objects in the system maintained belief pool.

Returns:

It returns a printable character stream, each line contains a HypothesisSet name followed by its belief interval values.

`public String printBeliefPool()`

This method helps to print out the HypothesisSet objects maintained in the system belief pool. Each printed line contains the name and their corresponding basic probability assignment value of a HypothesisSet object.

Returns:

It returns a printable character stream that contains a HypothesisSet object in each line.

`public void registerHypothesis(Hypothesis h)`

This method helps to keep a record of all the possible hypotheses, the so-called "frame of discernment", in this instance of a Dempster-Shafer reasoning system.

`public void reset()`

This method resets the Dempster-Shafer reasoning system. The reset process includes three steps: first, the system's frame of discernment information is set to empty; second, the system's belief pool is set to empty; and third, the system's evidence pool is set to empty.

public void restart()

This method partially resets the Dempster-Shafer reasoning system. It resets the system's belief pool and evidence pool but does not empty the system's frame of discernment information.

public void sortBeliefPool()

This method helps rearrange the HypothesisSet objects in the system's belief pool. The objects are stored in a SortVector object and the rearrangement sorts the HypothesisSet objects in a decreasing order by their basic probability assignment value.

public String toString()

This method prints out all the registered hypotheses in the Demspter-Shafer reasoning system. These are the Hypothesis objects that consist of the frame of discernment of the system.

Overrides:

`toString` in class `java.lang.Object`

C.2.1. private interface I_Comparator

This interface defines two object comparison formats to be used in SortableVector inner class.

C.2.2. Class SortableVector

Extends: `java.util.Vector`

This inner class extends the java Vector class with a specifically defined object comparison operation so that it can easily sort its contained objects. It is intended for handling the Dempster-Shafer reasoning system's belief pool and evidence pool.

private I_Comparator compare

This private parameter contains an instance of the Comparator inner-class that implements the I_comparator interface. The behavior of the Comparator object defines how the current SortableVector objects sort their contents.

`public SortableVector(I_Comparator comp)`

This constructor constructs an inner-class SortableVector object with an object that implements I_Comparator interface.

Parameters:

`comp` – the (Comparator inner-class) object that defines how the current SortableVector object will sort its contained objects.

`public void Sort()`

This method sorts all objects contained in the SortableVector class instance.

`private void quickSort(int left, int right)`

This method sorts the stored objects in the current SortableVector so that the between the given index `left` and index `right` they are in ascending order, defined by the `compare` object.

Parameters:

`left` – the left-end index, objects with index to its right will be sorted.

`right` – the right-end index, objects with index to its left will be sorted.

`private void swap(int loc1, int loc2)`

This method swaps the two objects with the given index `loc1` and index `loc2` respectively in current SortableVector.

Parameters:

`loc1` – the index of one of the two objects that will be swapped.

`loc2` – the index of the other of the two objects that will be swapped.

C.2.3. Class Comparator implements I_Comparator

This inner class implements the I_Comparator interface to define comparison relationship between HypothesisSet objects.

`public boolean lessThan(Object lhs, Object rhs)`

This method defines the "less than" relationship between two given HypothesisSet objects according their basic probability assignment value.

Parameters:

`lhs` – the first HypothesisSet object to be compared.

`rhs` – the second HypothesisSet object to be compared.

Returns:

It returns a boolean "true" if the `lhs` HypothesisSet object's basic probability assignment value is less that of `rhs`, a boolean "false" if otherwise.

`public lessThanOrEqual(Object lhs, Object rhs)`

This method defines the "less than or equal" relationship between two given HypothesisSet objects according their basic probability assignment value.

Parameters:

`lhs` – the first HypothesisSet (left-hand side) object to be compared.

`rhs` – the second HypothesisSet (right-hand side) object to be compared

Returns:

It returns a boolean "true" if the `lhs` HypothesisSet object's basic probability assignment value is less than or equal to that of `rhs`, a boolean "false" if otherwise.

C.3. Class Evidence

This helper class comprises of a HypothesisSet class object and a character string name so that it is easier to be referenced. For convenience, this class provides a constructor that can pass a float value to the parameter of basic probability assignment of the inside HypothesisSet class object.

`private String name`

This private String parameter provides a descriptive name or sentence as an explanation of this Evidence object.

private HypothesisSet hset

This private HypothesisSet object is the core of the current evidence object.

public Evidence(String n)

This constructor constructs an Evidence object with only the reference name, or a short explanation, being specified, its parameter is originally set as an "EvidencePiece". The constructor does not specify the inside HypothesisSet object yet.

Parameters:

n – the name or an explanation character string to describe the evidence object.

public Evidence(String n, HypothesisSet h, float bpa)

This constructor constructs a new Evidence object with all the needed information including a name or short explanation string, a HypothesisSet object, and a float number to be used as the basic probability assignment value for the inside HypothesisSet object.

Parameters:

n – name or an explanation string to describe the Evidence object.

h – the HypothesisSet to be included inside the newly created Evidence object.

bpa – the basic probability assignment for the inside HypothesisSet object.

public Evidence(String n, HypothesisSet h)

This constructor constructs a new evidence object with a given name or explanation string and a HypothesisSet object, but without providing the basic probability assignment value of the HypothesisSet object, the default value of 0.0f is used.

Parameters:

n – the name or an explanatory string to describe the Evidence object.

h – to be used as the inside HypothesisSet object.

public void addHypothesis(Hypothesis h)

This method adds the given Hypothesis object into the calling object's inside HypothesisSet object.

Parameters:
`h` – the Hypothesis object to be added into the Evidence object's HypothesisSet object.

`public float getBelief()`

This method gets the basic probability assignment value of the HypothesisSet object inside the calling Evidence object.

Returns:
It returns the basic probability assignment value the current Evidence object.

`public String getName()`

This method returns the name or an explanation stored inside and regarding the current calling Evidence object.

`public HypothesisSet getSet()`

This method returns the HypothesisSet object inside the calling Evidence object.

`public void setBelief(float bpa)`

This method uses the given float number to set the basic probability assignment value of the HypothesisSet object inside the current calling Evidence object.

`bpa` – the float number to be used as the basic probability assignment value.

`public String toString()`

This method prints out the current calling Evidence object whose content includes its name or explanation, the inside HypothesisSet object, and the basic probability assignment value.

Overrides:

toString in class java.lang.Object

C.4. Class Hypothesis

This class is the basic data structure for Dempster-Shafer sensor fusion as it constructs the element hypothesis set of frame of discernment. It contains a "name" for its identification.

public Hypothesis(String name)

The constructor of the Hypothesis class with only a string parameter for its instance name.

Parameters:

`name` — the name for the newly built Hypothesis class instance.

public Hypothesis(String name, String description)

The constructor of the Hypothesis class with a string parameter for its name and a description regarding this Hypothesis object instance.

Parameters:

`name` — The name of the Hypothesis object to be constructed, it is used as identifier of the object.

description — A description of Hypothesis object to be constructed. It should explain this and related elements in the frame of discernment in Dempster-Shafer reasoning system.

public String getName()

This function returns the name of the Hypothesis object instance.

public String getDescription()

This function returns the description of the Hypothesis object instance as a text string to be printed out.

public String toString()

This function returns a text string that is formatted for html style print. The text string contains first the name of the Hypothesis object instance, formatted in bold fonts, then the description of the Hypothesis object instance starting in a new line.

Overrides:

`toString` in class `java.lang.Object`

C.5. Class HypothesisSet

This class constructs an evidence object that comprises of the basic hypothesis elements in a Dempster-Shafer reasoning system. It may contain any number of the basic hypothesis elements from zero to all the hypotheses in the frame of discernment. When it does not actually contain any hypothesis, it represents the null set, whereas when it contains all the elements of the basic hypotheses in the frame of discernment, it represents the ignorance set in the Dempster-Shafer reasoning system.

`public HypothesisSet()`

The constructor without any parameter initializes the HypothesisSet object being built to the null set (i.e., the conflict set) with its basic probability assignment being set to zero.

`public HypothesisSet(Hypothesis h)`

This constructor builds an evidence object with the included Hypothesis object as its only element of hypotheses. Notice that the newly built object has been labeled as non-null set (isNull value is set to false), but the basic probability assignment value is still zero as set by the being called empty-parameter constructor inside this constructor.

Parameters:

`h` — the Hypothesis object to be used as the first hypothesis element of the newly built HypothesisSet object.

`public HypothesisSet(Hypothesis h, float bpaValue)`

This constructor builds a HypothesisSet object with the given Hypothesis object its basic hypothesis element and the given float number as its basic probability assignment value.

Parameters:

`h` — the Hypothesis object to be used as the first hypothesis element of the newly built HypothesisSet object.

`bpaValue` — this number to be used as the basic probability assignment value of the newly built HypothesisSet object.

public HypothesisSet(HypothesisSet hs)

This constructor builds a new HypothesisSet object using the given HypothesisSet object as template, i.e., the newly built object will comprise of the same hypothesis elements, and will take the same properties or values, of the given HypothesisSet object.

Parameters:

`hs` — the HypothesisSet object to be used as template for building the new hypothesisSet object.

public void absorbBpaValue(HypothesisSet hpst)
throws java.lang.IllegalArgumentException

This function first checks to see whether the current HypothesisSet object has the same hypothesis elements as the given HypothesisSet object, if so, it then adds the basic probability assignment value of the given HypothesisSet object onto this current HypothesisSet object.

Parameters:

`hpst` — the HypothesisSet object whose basic probability assignment value might be absorbed to this current HypothesisSet object.

Exceptions:

`java.lang.IllegalArgumentException` — the basic probability assignment value cannot be absorbed if it is of a hypothesis set with different content.

public void absorbHypotheses(HypothesisSet hpst)

This function absorbs the basic hypothesis elements of the given HypothesisSet object but does not change the basic probability assignment value of either object.

Parameters:

`hpst` — the HypothesisSet object whose hypothesis elements are to be extracted and absorbed into the calling object.

public void addHypothesis(Hypothesis h)

This function adds the given Hypothesis object onto its hypothesis element set but does not change the basic probability assignment value.

Parameters:

`h` — the hypothesis object to be added onto the calling object's hypothesis element set.

`public void addToBpa(float f)`

This function adds the given float number to the calling object's basic probability assignment value.

Parameters:

`f` — the number to be added to the current calling object's basic probability assignment value.

`public boolean contains(HypothesisSet hs)`

This function checks to see whether this current calling HypothesisSet object is a superset of the hypothesis elements that the given HypothesisSet object contains. It returns a "true" value if this calling HypothesisSet object has all the hypothesis elements that the given HypothesisSet object contains.

Parameters:

`hs` — the HypothesisSet object whose hypothesis elements are checked to see whether they are included in the current calling HypothesisSet object

`public boolean equals(Object o)`

This method tests whether the given HypothesisSet object equals to the current calling HypothesisSet object, in terms of whether the two contain the same hypothesis elements. All other conditions, such as basic probability assignment value, plausibility value, are not checked by this method.

Parameters:

`o` — the object to be checked for whether it is a HypothesisSet instance that has the same hypothesis elements as the current calling object.

Overrides:

It overrides the "equals" function in class java.lang.Object.

`public float getBpa()`

The function returns the basic probability assignment value of the calling HypothesisSet object in a float number format.

`public Hypothesis getHypothesis(String name)`

This function returns the Hypothesis object with the given name in the current calling HypothesisSet object.

Parameters:

name — the name of the Hypothesis object to be found in the current calling HypothesisSet object.

`public java.util.Enumeration getHypotheses()`

This method gets all the hypothesis elements in the current HypothesisSet object in format of a Java Hashtable object.

`public java.util.Enumeration getHypothesesNames()`

This method gets the name of hypothesis element objects in the current calling HypothesisSet object in the format of a Java Hashtable's keys.

`public int getSize()`

This method reports the number of hypothesis elements in the current calling HypothesisSet object.

`public boolean isEmpty()`

This method checks whether there is any hypothesis element object in the current calling HypothesisSet object. It returns a "true" value if there is not any hypothesis object already included.

`public boolean isTheta()`

This method reports whether the current calling HypothesisSet object thinks that it encompasses all the registered hypothesis elements in the frame of discernment. In other words, it returns a "true" value when it thinks so, thus indicating itself as actually the ignorance set in the Dempster-Shafer reasoning system.

`public boolean isNull()`

This method reports whether the current calling HypothesisSet object regards itself as an impossible evidence combination, i.e., a conflict in the Dempster-Shafer reasoning system.

`public void setBpa(float f)`

This method sets the basic probability assignment value, using the given float number to replace its original value.

Parameters:

`f` — the float number to be assigned as the new basic probability assignment value of the current calling HypothesisSet object.

`public void setTheta(Boolean tf)`

This method sets the calling HypothesisSet object's flag of ignorance (Theta) acknowledgement according to the given Boolean value. The method does not check whether the calling object actually contains all the elements of the frame of discernment.

Parameters:

`tf` — the Boolean value to be set for the acknowledgement of ignorance set.

`public void setNull(Boolean tf)`

This method sets the calling HypothesisSet object's flag regarding whether it acknowledges itself as a null set (an invalid set, an empty set, or a conflict).

Parameters:

`tf` — the Boolean value to be set for the acknowledgement of null set.

`public String toString()`

This method prints out all the Hypothesis elements contained in the calling HypothesisSet object.

Overrides:

It overrides the "toString" function in class java.lang.Object.

`private String formatFloat(float f)`

This method rounds the given number to the first three digits after decimal point, it then returns the number in string format. The method is intended for printing out basic probability mass function value.

Parameters:

`f` — the float number to be print out up to 3 digits after decimal point.

`public float getPlausibility()`

This method returns the plausibility value stored in the calling HypothesisSet object.

`public void setPlausibility(float f)`
`  throws java.lang.IllegalArgumentException`

This function first checks to see whether the current HypothesisSet object has the same hypothesis elements as the given HypothesisSet object, if so, it then adds the basic probability assignment value of the given HypothesisSet object onto this current HypothesisSet object.

Parameters:

hpst — the HypothesisSet object whose basic probability assignment value might be absorbed to this current HypothesisSet object.

Exceptions:

java.lang.IllegalArgumentException — the basic probability assignment value cannot be absorbed if it is of a hypothesis set with different content.

References

[1]. Anind K. Dey and Gregory D. Abowd, "Towards a Better Understanding of Context and Context-Awareness", Proceedings of the CHI 2000 Workshop on "The What, Who, Where, When, and How of Context-Awareness", The Hague, Netherlands, April 1-6, 2000, ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

[2]. Bill N. Schilit and Marvin M. Theimer, "Disseminating Active Map Information to Mobile Hosts", IEEE Network, 8(5) 1994, 22-32, http://citeseer.nj.nec.com/schilit94disseminating.html

[3]. Bill N. Schilit, N.L Adams, and R. Want, "ContextAware Computing Applications", Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. IEEE Computer Society. http://citeseer.nj.nec.com/schilit94contextaware.html

[4]. William Noah Schilit, "A System Architecture for Context-Aware Mobile Computing", Ph.D. thesis, Columbia University, 1995, http://seattleweb.intel-research.net/people/schilit/schilit-thesis.pdf

[5]. Mari Korkea-aho, "Context-Aware Applications Survey", http://www.hut.fi/~mkorkeaa/doc/context-aware.html

[6]. Roy Want, Andy Hopper, Veronica Falcao, and Jonathon Gibbons, "The Active Badge Location System" (ftp.uk.research.att.com:/pub/docs/att/tr.92.1.pdf), http://www.cam-orl.co.uk/ab.html

[7]. R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, and M. Weiser, "The ParcTab Ubiquitous Computing Experiment", Xerox Parc technical report, http://citeseer.nj.nec.com/want-parctab.html

[8]. Sue Long, Rob Kooper, Gregory D. Abowd, and Christopher G. Atkeson, "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study", Proc. 2nd ACM International Conference on Mobile Computing (MOBICOM), Rye, New York, U.S. http://www.cc.gatech.edu/fce/cyberguide/pubs/mobicom96-cyberguide.ps

[9]. J. Pascoe, "Adding Generic Contextual Capabilities to Wearable Computers", 2nd International Symposium on Wearable Computers, Pittsburgh, Pennsylvania USA, October 19-20, 1998, page 92-99. http://www.cs.ukc.ac.uk/pubs/1998/676/index.html

[10]. Nick Ryan, Jason Pascoe, and David R. Morse, "FieldNote : a Handheld Information System for the Field", First International Workshop on

TeloGeoProcessing (Telegeo'99), Lyon, 1999, http://citeseer.nj.nec.com/nick99fieldnote.html

[11]. Barry Brumitt, Brian Meyers, John Krumm, Amanda Kern, and Steven Shafer, "EasyLiving: Technologies for Intelligent Environments", Handheld and Ubiquitous Computing, September 2000. http://www.research.microsoft.com/easyliving/Documents/2000%2009%20Barry%20HUC.pdf

[12]. Steve Shafer, "Ten Dimensions of Ubiquitous Computing", Keynote presentation at Conference on Managing Interactions in Smart Environments, December 1999. http://www.research.microsoft.com/easyliving/Documents/"1999 12 Ten Dimensions.doc"

[13]. Cory D. Kidd, Robert J. Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth Mynatt, Thad E. Starner, and Wendy Newstetter, "The Aware Home: A Living Laboratory for Ubiquitous Computing Research", Proceedings of the Second International Workshop on Cooperative Buildings - CoBuild'99. Position paper, October 1999. http://www.cc.gatech.edu/fce/house/cobuild99_final.doc

[14]. Robert N. Johnson, "Building Plug-and-Play Networked Smart Transducer", Sensors Magazine, October 1997, http://www.smartsensor.com/doc/sensors.pdf

[15]. Stan P. Woods, "The IEEE-P1451.2 Draft Standard for Smart Transducer Interface Modules", Hewlett-Packard Company presented at Sensor Expo Boston, May 1997, http://www.sensorsmag.com/articles/1097/ieee1097/main.shtml

[16]. Anind K. Dey, Jennifer Mankoff, and Gregory D. Abowd, "Distributed Mediation of Imperfectly Sensed Context in Aware Environments", GVU Technical Report GIT-GVU-00-14. September 2000. ftp://ftp.cc.gatech.edu/pub/gvu/tr/2000/00-14.pdf

[17]. Anind K. Key, Daniel Salber, Gregory D. Abowd, and Masayasu Futakawa, "An Architecture To Support Context-Aware Applications", GVU Technical Report GIT-GVU-99-23. June 1999, ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-23.pdf.

[18]. Glenn Shafer, "A Mathematical Theory of Evidence", Princeton University Press, 1976

[19]. Mark Stefik, "Introduction to Knowledge Systems", Morgan Kaufman Publishers, Inc., 1995, ISBN 1-55860-166-X

[20]. Mongi A. Abidi and Rafael C. Gonzalez (editors), "Data Fusion in Robotics and Machine Intelligence", Academic Press, Inc. 1992, (ISBN 0-12-042120-8, CMU E&S 629.892 D232)

[21]. Gregory D. Abowd and Elizabeth D. Mynatt, “Charting Past, Present, and Future Research in Ubiquitous Computing”, ACM Transactions on Computer-Human Interaction, Vol. 7, No. 1, March 2000, Page 29-58. http://www.cc.gatech.edu/fce/pubs/tochi-millenium.pdf

[22]. Anind K. Dey, “Providing Architecture Support for Building Context-Aware Applications”, PhD thesis, November 2000, http://www.cc.gatech.edu/fce/ctk/pubs/dey-thesis.pdf

[23]. Lawrence A. Klein, “Sensor and Data Fusion Concepts and Applications” (second edition), SPIE Optical Engineering Press, 1999, ISBN 0-8194-3231-8

[24]. Frans Groen, “Sensor Data Fusion” presentation, 11/3/99, http://www.science.uva.nl/~arnoud/OOAS/Presentation9fus/

[25]. Jay Gowdy, “Emergent Architecture: A Case Study for Outdoor Mobile Robots”, PhD thesis (CMU-RI-TR-00-27), November 1, 2000, http://www.ri.cmu.edu/pub_files/pub2/gowdy_jay_2000_2/gowdy_jay_2000_2.pdf

[26]. Paul Castro, Patrick Chiu, Ted Kremenek, and Richard Muntz, “A Probabilistic Room Location Service for Wireless Networked Environments”, Proceedings of Ubicomp 2001, pp.18-34, Springer-Verlag LNCS 2201, Atlanta, GA, 2001 http://link.springer.de/link/service/series/0558/bibs/2201/22010018.htm

[27]. Gregory D. Abowd, Agathe Battestini, and Thomas O’Connell, “The Location Service: A Framework for Handling Multiple Location Sensing Technologies”, CHI 2002, Minneapolis, MN, April 20-25, 2002 http://www.cc.gatech.edu/fce/ahri/publications/location_service.pdf

[28]. Paul Castro and Richard Muntz, “Managing Context Data for Smart Spaces”, IEEE Personal Communications, Vol.7, No.5, October 2000 http://mmsl.cs.ucla.edu/publications/pdf/ieeePCOct2000.pdf

[29]. Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen, “There is more to Context than Location”, Proceedings of the International Workshop on Interactive Applications of Mobile Computing (IMC98), November 1998, Rostock, Germany, http://www.teco.uni-karlsruhe.de/~albrecht/publication/imc98.ps

[30]. Rainer Stiefelhagen, Jie Yang, and Alex Waibel, “Estimating Focus of Attention Based on Gaze and Sound”, Proceedings of Workshop on Perceptive User Interfaces PUI 2001, Orlando, Florida, USA http://www.is.cs.cmu.edu/papers/multimodal/PUI01/PUI2001_rainer.pdf

[31]. Huadong Wu, Mel Siegel, Rainer Stiefelhagen, and Jie Yang, "Sensor Fusion Using Dempster-Shafer Theory," presented at IEEE International Measurement Technology Conference (IMTC) 2002, Anchorage AK USA, 2002, http://www-2.cs.cmu.edu/~whd/publications/1076-Siegel.pdf

[32]. Jie Yang and Alex Waibel, "A Real-Time Face Tracker", Proceedings of WACV, Page 142-147, 1996. http://www.is.cs.cmu.edu/papers/multimodal/96.wacv.jie.ps.gz

[33]. Rainer Stiefelhagen, "Tacking Focus of Attention in Meetings", Proceedings of IEEE International Conference on Multimodal Interfaces 2002, Pittsburgh PA USA October 14-16, 2002. http://www.is.cs.cmu.edu/papers/multimodal/ICMI2002/icmi2002_stiefelhagen.pdf

[34]. David M. Kroenke, "Database Processing: Fundamentals, Design & Implementation", 8th Edition, Prentice Hall, 2002

[35]. Sebastien Populaire, Joelle Blanc, Thierry Denoeux, Philippe Ginestet, and Albert Mpe A Guilikeng, "Fusion of Expert Knowledge with Data using Belief Functions: a case study in wastewater treatment", 5th International Conference on Information Fusion, Annapolis, Maryland, USA, 7-11 July 2002

[36]. Ren C. Luo and M. G. Kay, "Multisensor Integration and Fusion in Intelligent Systems," IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-19(5) Page 901-931, http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=44007

[37]. R. C. Luo and M. G. Kay, "Data Fusion and Sensor Integration: State-of-the-art 1990s", Chapter 2 of "Data Fusion in Robotics and Machine Intelligence", Academic Press, Inc., 1992

[38]. H. Qi, X. Wang, S. S. Iyengar, and K. Chakrabarty, "Multisensor Data Fusion In Distributed Sensor Networks Using Mobile Agents", Proceedings of International Conference on Information Fusion, pp. 11-16, August 2001

[39]. H. Qi, S. S. Iyengar and K. Chakrabarty, "Distributed Sensor Fusion - A Review Of Recent Research", Journal of the Franklin Institute, vol. 338, pp. 655-668, 2001.

[40]. H. Qi, S. S. Iyengar and K. Chakrabarty, "Distributed Multi-Resolution Data Integration Using Mobile Agents", Proc. IEEE Aerospace Conference, vol. 3, pp. 1133-1141, 2001.

[41]. H. Qi, S. S. Iyengar and K. Chakrabarty, "Multi-Resolution Data Integration Using Mobile Agents In Distributed Sensor Networks", IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews, vol. 31, no. 3, pp383-391, August, 2001.

[42]. Gregory D. Abowd, Elizabeth D. Mynatt, and Tom Rodden, "The Human Experience", IEEE Pervasive Computing, Vol.1 No.1; January-March, pp. 48-57, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/01/b1048.pdf

[43]. David Garlan, Dan Siewiorek, Asim Smailagic, and Peter Steenkiste, "Project Aura: Toward Distraction-Free Pervasive Computing", IEEE Pervasive Computing, Vol.1 No.2; April-June, pp. 22-31, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/02/b2022.pdf

[44]. Hani Naguib and George Coulouris, “Location Information Management”, Proceedings of Annual Conference on Ubiquitous Computing 2001, Atlanta, USA,October 2001, http://www-lce.eng.cam.ac.uk/qosdream/Publications/ubicomp.pdf

[45]. Angela Pawlowski and Craig Stoneking, “Army Aviation Fusion of Sensor-Pushed and Agent-Pulled Information”, American Helicopter Society 57th Annual Forum, Washington, DC, May 9-11, 2001, http://www.atl.external.lmco.com/overview/papers/1107.pdf

[46]. Steve Jameson, “Architectures for Distributed Information Fusion To Support Situation Awareness on the Digital Battlefield”, 4th International Conference on Data Fusion, August 2001, http://www.atl.external.lmco.com/overview/papers/1030.pdf

[47]. Charles Dean Haynie, “Development of a Novel Zero-Turn Radius Autonomous Vehicle”, M.S. Thesis, Mechanical Engineering Department, Virginia Polytechnic Institute and State University, August 1998, http://scholar.lib.vt.edu/theses/available/etd-7698-132910/

[48]. James Llinas, and David Hall, “An Introduction to Multi-Sensor Data Fusion”, Proceedings of the IEEE, Vol. 85, No.1, January, 1997, pp.6-20, http://www.infofusion.buffalo.edu/reports/Dr. Llinas' stuff/papers/intro_mltisnsr_data_fusion.pdf

[49]. Edward L. Waltz, “Information Understanding: Integrating Data Fusion and Data Mining Processes”, Circuits and Systems, 1998. ISCAS '98. Proceedings of the 1998 IEEE International Symposium on , Volume: 6 , 31 May-3 June 1998 Page(s): 553 -556 vol.6, http://ingengineering.com/Documents/Docs-DataFusion/MPA11_6.PDF

[50]. Kristof Van Laerhoven, “On-line Adaptive Context Awareness Starting from Low-level Sensors”, Masters Thesis, the Free University of Brussels (VUB). Brussels, Belgium, 1999, http://www.comp.lancs.ac.uk/~kristof/old/papers/thesis99.pdf

[51]. M. Kokar, J. A. Tomasik, and J. Weyman, “A formal approach to information fusion”, Proceedings of 2nd Intern. Conf. On Information Fusion, vol. I, pp. 133--140, 1999, http://www.coe.neu.edu/~kokar/publications/f99-mjj.ps

[52]. Dave McDaniel, “An Information Fusion Framework for Data Integration”, Proceedings of the 13th Software Technology Conference, 2001

[53]. David L. Hall and James Llinas (editors), “Handbook of Multisensor Data Fusion”, CRC Press, 2001

[54]. R.S. Doyle and C.J. Harris, “Multi-Sensor Data Fusion for Obstacle Tracking Using Neuro-Fuzzy Estimation Algorithms”, SPIE (The International Society for Optical

Engineering) Proceedings Vol.2233, Sensor Fusion and Aerospace Applications II, 6-7 April 1994, Orlando, Florida, Page 112-123

[55]. Shulin Yang and Kuo-Chu Chang, "Modular Neural Net Architecture for Automatic Target Recognition", SPIE Proceedings Vol.2755, Signal Processing, Sensor Fusion, and Target Recognition V, 8-10 April 1996, Orlando, Florida, Page 166-177

[56]. Ronald P.S. Mahler, "Unified Data Fusion: Fuzzy Logic, Evidence, and Rules", SPIE Proceedings Vol.2755, Signal Processing, Sensor Fusion, and Target Recognition V, 8-10 April 1996, Orlando, Florida, Page 226-237

[57]. Ronald P.S. Mahler, "Combining Ambiguous Evidence With Respect To Ambiguous A Priori Knowledge, I: Boolean Logic", IEEE Transactions on Systems, Man and Cybernetics, Part A, Volume: 26 Issue: 1 , Jan. 1996, Page 27-41

[58]. Ronald P.S. Mahler, "The modified Dempster-Shafer approach to classification", IEEE Transactions on Systems, Man and Cybernetics, Part A, Volume: 27 Issue: 1, Jan. 1997, Page 96 -104

[59]. Firooz A. Sadjadi (editor), "Selected Papers on Sensor and Data Fusion", SPIE Milestore Series Volume MS 124, SPIE Press, 1996, ISBN 0-8194-2265-7

[60]. Richard T. Antony, "Principles of Data Fusion Automation", Artech House Inc., 1995, ISBN 0-89006-760-0

[61]. Alan N. Steinberg, Christopher L. Bowman, and F. E. White, "Revisions to the JDL Data Fusion Model", Proceedings of SPIE AeroSense (Sensor Fusion: Architectures, Algorithms and Applications III), page 430-441, Orlando, Florida, 1999.

[62]. Erik. P. Blasch and Susan Plano, "JDL Level 5 fusion model: user refinement issues and applications in group tracking," SPIE Vol. 4729, Aerosense, 2002, pp. 270 – 279

[63]. David Lee Hall, "Mathematical Techniques in Multisensor Data Fusion", Artech House Inc., 1992 ISBN 0-89006-558-6

[64]. I. R. Goodman, Ronald P. S. Mahler, and Hung T. Hguyen, "Mathematics of Data Fusion", Kluwer Academic Publishers, 1997, ISBN 0-7923-4674-2

[65]. Todor Tagarev and Petya Ivanova, "Computational Intelligence in Multi-Source Data and Information Fusion", Inform & Security, Vol. 2, 1999, ISSN 1311-1493, http://www.isn.ethz.ch/researchpub/publihouse/infosecurity/volume_2/f4/f4_index.htm

[66]. Albrecht Schmidt, "Implicit Human Computer Interaction Through Context", Personal Technologies, Vol. 4(2), June 2000 http://citeseer.nj.nec.com/schmidt00implicit.html

[67]. Martin Jansson, "Context Shadow: An Infrastructure for Context Aware Computing", Proceedings of Artificial Intelligence in Mobile System 2002, Lyon, France, http://www.dsv.su.se/feel/DSV/ContextShadow.pdf

[68]. Albrecht Schmidt and Jessica Forbess, "What GPS Doesn't Tell You: Determining One's Context with Low-level Sensors", The 6th IEEE International Conference on Electronics, Circuits and Systems, September 5 - 8, 1999, Paphos, Cyprus, http://www.hum.auc.dk/~slau01/inf8/artikler/Barkhuus_Artikel_Ref/what-gps-doesn-t.pdf

[69]. Hans-W. Gellersen, Michael Beigl and Albrecht Schmidt, "Sensor-based Context-Awareness for Situated Computing", Workshop on Software Engineering for Wearable and Pervasive Computing SEWPC00 at the 22nd Int. Conference on Software Engineering ICSE 2000, Limerick, Ireland, 6.June 2000, http://www.teco.uni-karlsruhe.de/~albrecht/publication/sewpc00/sensor-based-context.pdf

[70]. Hans-W. Gellersen, Albrech Schmidt, and Michael Beigl, "Multi-Sensor Context-Awareness in Mobile Devices and Smart Artifacts", Mobile Networks and Applications (MONET), Oct 2002, http://www.comp.lancs.ac.uk/~albrecht/pubs/pdf/gellersen_monet_2002.pdf

[71]. Datong Chen, Albrecht Schmidt, and Hans-W. Gellesen, "An Architecture for Multi-Sensor Fusion in Mobile Environments", Proceedings International Conference on Information Fusion, Sunnyvale, CA, USA, July 1999. Vol. II, pp 861-868, http://www.comp.lancs.ac.uk/~albrecht/pubs/pdf/chen_fusion1999.PDF

[72]. Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Heikki Keranen, and Esko-Juhani Malm, "Managing Context Information in Mobile Devices", IEEE Pervasive Computing, Volume 2 Number 3, July-September 2003, pp. 42-51, ISSN 1536-1268

[73]. Stephen S. Yau, Fariaz Karim, Yu Wang, Bin Wang, and Sandeep K.S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing", IEEE Pervasive Computing, Vol.1 No.3; July-September 2002, pp. 33-40, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/03/b3033.pdf

[74]. Scott F. Midkiff, "Mobile Computing 'Killer App' Competition", IEEE Pervasive Computing, Vol.1 No.3; July-September 2002, pp. 101-104, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/03/b3101.pdf

[75]. Vince Stanford, "Using Pervasive Computing to Deliver Elder Care", IEEE Pervasive Computing, Vol.1 No.1; January-March 2002, pp. 10-13, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/01/b1010.pdf

[76]. Nigel Davies and Hans-Werner Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems", IEEE Pervasive Computing, Vol.1 No.1; January-

March 2002, pp. 26-35, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/01/b1026.pdf

[77]. Roy Want, Gaetano Borriello, Trevor Pering, and Keith I. Farkas, "Disappearing Hardware", IEEE Pervasive Computing, Vol.1 No.1; January-March 2002, pp. 36-47, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/01/b1036.pdf

[78]. Deborah Estrin, David Culler, Kris Pister, and Gaurav Sukhatme, "Connecting the Physical World with Pervasive Networks", IEEE Pervasive Computing, Vol.1 No.1; January-March 2002, pp. 59-69, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/01/b1059.pdf

[79]. Tim Kindberg and Armando Fox, "System Software for Ubiquitous Computing", IEEE Pervasive Computing, Vol.1 No.1; January-March 2002, pp. 70-81, ISSN 1536-1268, http://csdl.computer.org/dl/mags/pc/2002/01/b1070.pdf

[80]. Steve Shafer, Barry Brumitt, and Brian Meyers, "The EasyLiving Intelligent Environment System", CHI Workshop on Research Directions in Situated Computing, April 2000, http://research.microsoft.com/easyliving/Documents/"2000 04 Steve Shafer CHI.doc"

[81]. Mark Weiser, "The Computer of the 21st Century," Scientific American, Vol.265 No.3, September 1991, pp.19-25, http://www.cc.gatech.edu/fac/Gregory.Abowd/hci-resources/area-bok/papers/p94-weiser.pdf

[82]. Stelios C.A. Thomopoulos, "Sensor Integration and Data Fusion", Journal of Robotics Systems, Volume 7 Number 3 Page 337 – 372, June 1990, ISSN 0741-2223

[83]. S.C.A. Thomopoulos, R. Viswanathan, and D.K. Bougoulias, "Optimal Distributed Decision Fusion", IEEE Transactions on Aerospace and Electronic Systems, Volume 25 Issue 5, September 1989, Page 761 – 765

[84]. Miguel A. Munoz, Marcela Rodriguez, Jesus Favela, Ana I. Martinez-Garcia, and Victor M. Gonzalez, "Context-Aware Mobile Communication in Hospitals", IEEE Computer (ISSN 0018-9162) September 2003, Volume 36 Number 9, Page 38 – 46

[85]. Brad A. Myers, Michael Beigl, "Handheld Computing", IEEE Computer (ISSN 0018-9162), September 2003, Volume 36 Number 9, Page 27 – 29

[86]. Irfan A. Essa, "Ubiquitous Sensing for Smart and Aware Environments: Technologies towards the Building of an Aware Home", Position Paper for the DARPA/NSF/NIST Workshop on Smart Environments, July 1999, http://www.cc.gatech.edu/fce/ahri/publications/pp.pdf

[87]. Jeffrey Hightower, and Gaetano Borriello, "Location Systems for Ubiquitous Computing", IEEE Computer (ISSN 0018-9162), August 2001, Page 57 – 66

[88]. Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, Andy Ward, and Andy Hopper, "Implement a Sentient Computing System", IEEE Computer (ISSN 0018-9162), August 2001, Page 50 – 56

[89]. Robert M. Mosee, "A Discipline Independent Definition of Information", Journal of the American Society for Information Science, 48 (3) 1997, 254 – 269 http://www.ils.unc.edu/~losee/book5.pdf

[90]. Salil Pradhan, Cyril Brignone, Jun-Hong Cui, Alan McReynolds, and Mark T. Smith, "Websigns: Hyperlinking Physical Locations to the Web", IEEE Computer (ISSN 0018-9162) August 2001, Page 42 – 48

[91]. Nigel Davies, Keith Cheverst, Keith Mitchell, and Alon Efrat, "Using and Determining Location in a Context-Sensitive Tour Guide", IEEE Computer (0018-9162) August 2001, Page 35 – 41

[92]. Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, Edmondo Ferretti, and Massimiliano Margarone, "VeGame: Exploring Art and History in Venice", IEEE Computer (ISSN 0018-9162) September 2003 Volume 36 Number 9, Page 48 – 55

[93]. Roy Thomas Fielding, "Architectural Styles and the Design of Network-based Software Architecture", doctoral dissertation, School of Information and Computer Science, University of California, Irvine, 2000, http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[94]. Nenad Medvidovic, Marija Mikic-Rakic, Nikunj R. Mehta, and Sam Malek, "Software Architectural Support for Handheld Computing", IEEE Computer (ISSN 0018-9162) September 2003 Volume 36 Number 9, Page 66 – 73

[95]. Stan Franklin, and Art Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer Verlag, 1996, http://www.cs.memphis.edu/~franklin/AgentProg.html

[96]. Eric M. Dashofy, Nenad Medvidovic, and Richard N. Taylor, "Using Off-The-Shelf Middleware to Implement Connectors in Distributed Software Architectures", Proceeding of 1999 International Conference on Software Engineering (ICSE'99), ACM, New York, 1999, 3-12.

[97]. Philip A. Bernstein, "Middleware: A Model for Distributed System Services", Communications of the ACM February 1996 Vol. 39 No. 2, pp.86-98

[98]. Mary Shaw, and Paul Clements, "A Field Guide to Boxology: Preliminary Classification of Architectural Styles for Software Systems", Proceeding of the 2nd International Software Architecture Workshop (ISW-2), San Francisco, CA, USA, October 1996, pp. 50 – 54

[99]. Nenad Medvidovic, and David S. Rosenblum, “Domains of Concern in Software Architectures and Architecture Description Languages”, Proceedings of the 1997 USENIX Conference on Domain-Specific Languages, October 15-17, Santa Barbara, California,

[100]. Antonio Carzaniga, Elisabetta Di Nitto, David S. Rosenblum, and Alexander L. Wolf, “Issues in Supporting Event-based Architectural Styles”, Proceedings of the 3rd International Software Architecture Workshop (ISAW-3), Orlando, Florida, USA, November 1998 http://www.cs.colorado.edu/users/carzanig/papers/isaw3.pdf

[101]. Antonio Carzaniga, David S. Rosenblum, and Alexander L. Wolf, “Design of a Scalable Event Notification Service: Interface and Architecture”, Technical Report CU-CS-863-98, Department of Computer Science, University of Colorado, August, 1998, http://www.cs.colorado.edu/users/carzanig/papers/CU-CS-863-98.pdf

[102]. Marco Castaldi, Antonio Carzaniga, Paola Inverardi, and Alexander L. Wolf, “A Lightweight Infrastructure for Reconfiguring Applications”, in B. Westfechtel, A. van der Hoek (Editors): SCM 2001/2003, LNCS 2649, pp. 231-244, 2003, Springer-Verlag. in conjunction with ICSE 03, International Conference on Software Engineering. Portland, Oregon. May, 2003, http://www.cs.colorado.edu/users/carzanig/papers/cciw_scm11.pdf

[103]. Antonio Carzaniga, “Architectures for an Event Notification Service Scalable to Wide-area Networks”, PhD Thesis, Politecnico di Milano, December 1998, http://www.cs.colorado.edu/users/carzanig/papers/phd_thesis.pdf

[104]. Robert J. Allen, “A Formal Approach to Software Architecture”, Ph.D. Thesis, CMU-CS-97-144, SCS CMU, May `1997, http://reports-archive.adm.cs.cmu.edu/anon/1997/CMU-CS-97-144.pdf

[105]. David Garlan, “Software architecture: a Roadmap”, Proceedings of the Conference on The Future of Software, ELimerick, Ireland, May 2000, Page 91 – 101, ISBN 1-58113-253-0

[106]. William G. Griswold, Robert Boyer, Steven W. Brown, and Tan Minh Truong, “A Component Architecture for an Extensible, Highly Integrated Context-Aware Computing Infrastructure”, 2003 International Conference on Software Engineering (ICSE 2003), May 2003, http://www.cs.ucsd.edu/~wgg/Abstracts/ac-arch.pdf

[107]. Asim Smailagic, Daniel P. Siewiorek, Joshua Anhalt, and Francine Gemperle, Daniel Salber, Sam Weber, Jim Beck, Jim Jennings, “Towards Context Aware Computing: Experiences and Lessons”, IEEE Journal on Intelligent Systems, Vol. 16, No. 3, June 2001, pp 38-46 http://www.cs.cmu.edu/~asim/DistractionFreeComputing.pdf

[108]. Andreas Braun, Allen H. Dutoit, and Bernd Brügge, “A Software Architecture for Knowledge Acquisition and Retrieval for Global Distributed Teams”, International Workshop on Global Software Development, International Conference on Software

Engineering. Portland, Oregon, May 9, 2003, http://wwwbruegge.in.tum.de/publications/includes/pub/braun2003dcba/braun2003dcba.pdf

[109]. Gaëtan Rey, Joëlle Coutaz, and James L. Crowley, "The Contextor: A Computational Model for Context-Aware Computing", Position paper of Workshop of Ubiquitous Computing 2002 (UBICOMP 2002) September 29 – October 1, Goteborg, Sweden, http://iihm.imag.fr/rey/papier/UbicompWorkshop.pdf

[110]. J. Coutaz and G. Rey, "Foundation for a Theory of Contextors", in the Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces (CADUI02), ACM Publ., Valenciennes, France, May 2002, pp. 283 – 302, http://iihm.imag.fr/rey/papier/cadui02.pdf

[111]. James L. Crowley, Jolle Coutaz, Gaeten Rey, and Patrick Reignier, "Perceptual Components for Context Aware Computing", Ubiquitous Computing: Proceedings of 4th International Conference (UbiComp 2002), Göteborg, Sweden, September 29 - October 1, 2002, http://www-prima.inrialpes.fr/publi/CrowleyCoutaz-Ubicomp2002.pdf

[112]. Nenad Medvidovic, "Modeling Software Architecture in the Unified Modeling Language", ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 11 Issue 1, January 2002, Page 2 – 57 (ISSN:1049-331X)

[113]. Jason I. Hong, and James A. Landay, "An Infrastructure Approach to Context-Aware Computing", Human-Computer Interaction Vol. 16 No.2-4, December 2001, pp. 287 – 303, http://guir.berkeley.edu/projects/cfabric/pubs/context-essay-final.pdf

[114]. Gruia-Catalin Roman, Christine Julien, and Amy L. Murphy, "A Declarative Approach to Agent Centered Context-Aware Computing in Ad Hoc Wireless Environments", Software Engineering for Large-Scale Multi-Agent Systems, Garcia, A., Lucena, C., Zambonelli, F., Omicini, A., and Castro, J., (editors), Springer LNCS 2603, 2003, http://www.cs.wustl.edu/mobilab/pubs/SELMAS-02.pdf

[115]. Terry Winograd, "Architecture for Context", Human-Computer Interaction, Vol. 16 No. 2-4, 2001, pp. 401-419, http://www1.ics.uci.edu/~jpd/NonTradUI/SpecialIssue/winograd.pdf

[116]. Ivica Crnkovic, and Magnus Larsson (editors), "Building Reliable Component-Based Software Systems", Artech House Inc., 2002, ISBN 1-58053-327-2

[117]. Douglas K. Barry, "Web Services and Service-Oriented Architecture: The Savvy Manager's Guide", Morgan Kaufmann Publisher 2003, ISBN 1558609067

[118]. David S. Frankel, "Model Driven Architecture: Applying MDA to Enterprise Computing", Wiley Publishing Inc. 2003, ISBN 0-471-31920-1

[119]. Christine Hofmeister, Robert Nord, and Dilip Soni, "Applied Software Architecture", Addison Wesley Longman, Inc. 2000, ISBN 0-201-32571-3

[120]. Douglas Schmidt, Michael Stal, Hans Rohnert, and Frank Buschmann, "Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects", John Wiley & Son 2000, ISBN 0471606952

[121]. Dov Dori, "Conceptual Modeling and System Architecting, Introduction", Communications of ACM (Association for Computing Machinery), October 2003, Volume 46 Number 10, pp. 63 – 65, ISSN 0001-0782

[122]. Nathan R. Soderborg, Edward F. Crawley, and Dov Dori, "System Function and Architecture: OPM-based Definitions and Operational Templates", Communications of ACM, October 2003 Vol. 46 No.10, pp. 67 – 72, ISSN 0001-0782

[123]. Gregory Abowd, Robert Allen, and David Garlan, "Formalizing Style to Understand Descriptions of Software Architecture", ACM Transactions on Software Engineering and Methodology, Vol. 4, No. 4, October 1995, page 319 – 364 file://reports.adm.cs.cmu.edu/usr/anon/1995/CMU-CS-95-111.ps

[124]. Richard N. Taylor, Nenad Medvidovic, Kenneth M. Anderson, E. James Whitehead Jr., Jason E. Robbins, Kari A. Nies, Peyman Oreizy, and Deborah L. Dubrow, "A Component- and Message-Based Architectural Style for GUI Software", IEEE Transactions on Software Engineering, Vol. 22, No. 6, June 1996, pp. 390 – 406, http://sunset.usc.edu/~neno/teaching/s99/C2-TSE.pdf

[125]. Mohan Kumar, Behrooz A. Shirazi, Sajal K. Das, Byung Y. Sung, David Levine, and Mukesh Singhal, "PICO: A Middleware Framework for Pervasive Computing", IEEE Pervasive Computing, Volume 2 Number 3, July-September 2003, pp. 72 – 79, ISSN 1536-1268

[126]. Daniel D. Corkill, "Blackboard Systems", Journal of AI Expert, Vol. 6, No. 9, September 1991, pp. 40 – 47, http://bbtech.com/papers/ai-expert.pdf

[127]. Frantisek Plasil, and Michael Stal, "An Architectural View of Distributed Objects and Components in CORBA, Java RMI, and COM/DCOM", (Journal of) Software – Concepts and Tools, Vol. 19, No. 1, 1998, pp. 14 – 28, http://www.stal.de/Downloads/springer98.pdf

[128]. Kari Sentz and Scott Ferson, "Combination of Evidence in Dempster-Shafer Theory", Technical Report, April 2002 Los Alamos National Laboratory, Los Alamos, NM, http://www.sandia.gov/epistemic/Reports/SAND2002-0835.pdf

[129]. Robert A. Hummel and Michael S. Landy, "A Statistical Viewpoint on the Theory of Evidence", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10, No. 2, March 1988, pp. 235 – 247

[130]. Jurg Kohlas and Paul-Andre Monney, "A Mathematical Theory of Hints, An Approach to the Dempster-Shafer Theory of Evidence", Springer-Verlag Berlin Heidelberg, 1995, ISBN 3-540-59176-1

[131]. Ronald R. Yager, Janusz Kacprzyk, and Mario Fedrizzi (editors), "Advances in the Dempster-Shafer Theory of Evidence", John Wiley & Sons, February 1994, ISBN 0471552488 (CMU# 006.3.A2448)

[132]. James W. Hall, David I. Blockley, and John P. Davis, "Uncertainty Inference using Interval Probability Theory", Internationial Journal of Approximate Reasoning, Vol.19, No.3-4 (October 1998) pp. 247 – 264, http://www.cen.bris.ac.uk/civil/staff/jwh/Publications/IJAR19.pdf

[133]. James W. Hall, and Jonathan Lawry, "Generation, Combination and Extension of Random Set Approximations to Coherent Lower and Upper Probabilities", to in Journal of Reliability & System Safety, ISSN 0951-8320, http://www.cen.bris.ac.uk/civil/staff/jwh/Publications/hall_lawry_RESS_revised.pdf

[134]. George J. Klir, "Uncertainty and Information Measures for Imprecise Probability: An Overview", the Proceedings of the first International Probabilities and Their Applications", Ghent, Belgium, June 29 – July 2, 1999, ftp://decsai.ugr.es/pub/utai/other/smc/isipta99/050.pdf

[135]. David Harmanec, "Measures of Uncertainty and Information", Imprecise Probability Project (http://ippserv.rug.ac.be/home/ipp.html), http://www.sipta.org/documentation/summary_measures/summary_measures.html

[136]. Luis Mateus Rocha, "Relative Uncertainty and Evidence Sets: A Constructivist Framework", International Journal of General Systems, Vol. 26 (1-2), pp. 35 – 61, http://www.c3.lanl.gov/~rocha/ijgs_unc.html

[137]. Igor Douven, "Inference to Best Explanation is Coherent", in the "PSA(98): Proceedings of the Biennial Meeting of the Philosophy of Science Association, Part 1: Contributed Papers", University of Chicago Press, Don A. Howard (Editor), 1998, pp. 424 – 435, http://www.phil.uu.nl/preprints/preprints/PREPRINTS/preprint182.ps.Z

[138]. Luis Raul Pericchi, "Sets of Prior Probabilities and Bayesian Robustness", Imprecise Probability Project (http://ippserv.rug.ac.be/home/ipp.html), http://ippserv.rug.ac.be/documentation/robust/robust.html

[139]. Ronald P. S. Mahler, "Combing Ambiguous Evidence with respect to Ambiguous a priori Knowledge, I: Boolean Logic", IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, Vol. 26, No. 1, January 1996, pp. 27 – 41

[140]. S. James Press, "The Subjectivity of Scientists and the Bayesian Approach", John Wiley and Sons, Inc., 2001, ISBN 0-471-39685-0 (CMU 507.2.P93S)

[141]. S. James Press, "Subjective and Objective Bayesian Statistics: Principles, Models, and Applications, Second Edition", John Wiley and Sons, Inc., 2003, ISBN 0-471-34843-0 (CMU 507.2.P93S2)

[142]. David A. Freedman, "Notes on the Dutch Book Argument", Lecture Notes, Department of Statistics, University of Berkley at Berkley, http://www.stat.berkeley.edu/~census/dutchdef.pdf

[143]. Glenn Shafer and Judea Pearl (editors), "Readings in Uncertainty Reasoning", Morgan Kaufmann Publihser Inc., 1990, ISBN 1-55860-125-2

[144]. Philippe Semts and Robert Kennes, "The Transferable Belief Model", Artificial Intelligence Journal, Vol. 66 (1994), pp. 191 – 234, http://iridia.ulb.ac.be/~psmets/TBM-AIJ.pdf

[145]. Philippe Smets, "No Dutch Book Can Be Built Against TBM Even Though Update Is Not Obtained by Bayes Rule of Conditioning", SIS, Workshop on Probabilistic Expert Systems, R. Scozzafava(ed.), Roma, Italy, 1993 pp. 181 – 204, http://iridia.ulb.ac.be/~psmets/Dynamic_Dutch_Books.pdf

[146]. Joseph Y. Halpern and Riccardo Pucella, "A Logic for Reasoning about Upper Probabilities", Journal of Artificial Intelligence Research, 17, pp. 57 – 81, 2002, http://www.cs.cornell.edu/home/halpern/upjair.pdf

[147]. Bing Ma, "Parametric and Nonparametric Approaches for Multisensor Data Fusion", Ph.D. dissertation, Electrical Engineering: Systems, the University of Michigan, 2001, http://ww.eecs.umich.edu/~hero/Preprints/thesis_ma.pdf

[148]. Andrew Martin Robert Ward, "Sensor-driven Computing", Ph.D. dissertation, Corpus Christi College, University of Cambridge, August, 1998, http://www.sigmobile.org/phd/1999/theses/ward.pdf

[149]. Jennifer A. Healey, "Wearable and Automotive Systems for Affect Recognition from Physiology", Ph.D. dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, May 2000, ftp://whitechapel.media.mit.edu/pub/tech-reports/TR-526.ps.Z

[150]. Huadong Wu, Mel Siegel, and Sevim Ablay, "Sensor Fusion using Dempster-Shafer Theory II: Static Weighting and Kalman Filter-like Dynamic Weighting", IMTC (IEEE annual Instrumentation and Measurement Technology Conference) 2003 proceedings, Vail, CO USA, May 20-22, 2003, http://www-2.cs.cmu.edu/~whd/publications/7179.pdf

[151]. Ted Faison, "Interaction Patterns for Communicating Processes", The Proceedings of Pattern Languages of Program Conference, August 11-14, 1998,

Allerton Park, Monticello, Illinois, USA, http://jerry.cs.uiuc.edu/~plop/plop98/final_submissions/P02.pdf

[152]. John William McManus, "Design and Analysis Techniques for Concurrent Blackboard Systems", Ph.D. Dissertation, Department of Computer Science, College of William and Mary in Virginia, April 1992, http://techreports.larc.nasa.gov/ltrs/PDF/phd-92-mcmanus.pdf

[153]. John W. McManus, "Design and Analysis Tools for Concurrent Blackboard Systems", 10th AIAA/IEEE Digital Avionics Systems Conference, Los Angeles, California, October 14 - 17, 1991, pp. 432-439, http://techreports.larc.nasa.gov/ltrs/PDF/conf-10-dasc.pdf

[154]. David Howie, "Interpreting Probability: Controversies and Development in the Twentieth Century", Cambridge University Press, August 2002, ISBN 0521812518

[155]. Albert Tebo, "Sensor Fusion Employs A Variety of Architecture, Algorithm, and Applications", OE Reports (the International Society for Optical Engineering) 164, August 1997, http://www.spie.org/web/oer/august/aug97/sensor.html

[156]. Wilfried Elmenreich, "Sensor Fusion in Time-Triggered Systems", Ph.D. dissertation, Vienna University of Technology, October 2002, http://www.vmars.tuwien.ac.at/~wilfried/papers/elmenreich_Dissertation_sensorFusionInTimeTriggeredSystems.pdf

[157]. Scott Stillman and Irfan Essa, "Toward Reliable Mulitmodal Sensing in Aware Environments", Perceptual User Interfaces (PUI 2001) Workshop (held in conjunction with ACM UIST 2001 Conference), Orlando, Florida, November 15-16, 2001, http://www.cs.ucsb.edu/conferences/PUI/PUIWorkshop/PUI-2001/a18.pdf

[158]. Michael Hahn and Emmanuel Baltsavias, "Cooperative Algorithms and Techniques of Image Analysis and GIS", D. Fritsch, M. Englich, and M. Sester (editors), The International Archives of Photogrammetry and Remote Sensing Volume XXXII Part 4, ISPRS Commission IV Symposium on GIS – GIS Between Visions and Applications, http://www.ifp.uni-stuttgart.de/publications/commIV/hahn13neu.pdf